

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_CMAPI_CMAPI_INT_H_
#define CORE_FPDFAPI_CMAPI_CMAPI_INT_H_

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "third_party/base/span.h"

struct FXCMAP_DWordCIDMap {
  uint16_t m_HiWord;
  uint16_t m_LoWordLow;
  uint16_t m_LoWordHigh;
  uint16_t m_CID;
};

struct FXCMAP_CMap {
  enum MapType : uint8_t { Single, Range };

  const char* m_Name; // Raw, POD struct.
  const uint16_t* m_pWordMap; // Raw, POD struct.
  const FXCMAP_DWordCIDMap* m_pDWordMap; // Raw, POD struct.
  uint16_t m_WordCount;
  uint16_t m_DWordCount;
  MapType m_WordMapType;
  int8_t m_UseOffset;
};

const FXCMAP_CMap* FindEmbeddedCMap(pdfium::span<const FXCMAP_CMap> pCMaps,
                                     const ByteString& name);

uint16_t CIDFromCharCode(const FXCMAP_CMap* pMap, uint32_t charcode);
uint32_t CharCodeFromCID(const FXCMAP_CMap* pMap, uint16_t cid);

#endif // CORE_FPDFAPI_CMAPI_CMAPI_INT_H_
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_CMAPS_CNS1_CMAPS_CNS1_H_
#define CORE_FPDFAPI_CMAPS_CNS1_CMAPS_CNS1_H_

#include "core/fpdfapi/cmmaps/cmap_int.h"

extern const uint16_t g_FXCMAP_B5pc_H_0[];
extern const uint16_t g_FXCMAP_B5pc_V_0[];
extern const uint16_t g_FXCMAP_HKscs_B5_H_5[];
extern const uint16_t g_FXCMAP_HKscs_B5_V_5[];
extern const uint16_t g_FXCMAP_ETen_B5_H_0[];
extern const uint16_t g_FXCMAP_ETen_B5_V_0[];
extern const uint16_t g_FXCMAP_ETenms_B5_H_0[];
extern const uint16_t g_FXCMAP_ETenms_B5_V_0[];
extern const uint16_t g_FXCMAP_CNS_EUC_H_0[];
extern const FXCMAP_DWordCIDMap g_FXCMAP_CNS_EUC_H_0_DWord[];
extern const uint16_t g_FXCMAP_CNS_EUC_V_0[];
extern const FXCMAP_DWordCIDMap g_FXCMAP_CNS_EUC_V_0_DWord[];
extern const uint16_t g_FXCMAP_UniCNS_UCS2_H_3[];
extern const uint16_t g_FXCMAP_UniCNS_UCS2_V_3[];
extern const uint16_t g_FXCMAP_UniCNS_UTF16_H_0[];
extern const uint16_t g_FXCMAP_CNS1CID2Unicode_5[19088];
extern const FXCMAP_CMap g_FXCMAP_CNS1_cmmaps[];
extern const size_t g_FXCMAP_CNS1_cmmaps_size;

#endif // CORE_FPDFAPI_CMAPS_CNS1_CMAPS_CNS1_H_
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_CMAPI_GB1_CMAPI_GB1_H_
#define CORE_FPDFAPI_CMAPI_GB1_CMAPI_GB1_H_

#include "core/fpdfapi/cmapi/cmap_int.h"

extern const uint16_t g_FXCMap_GB_EUC_H_0[];
extern const uint16_t g_FXCMap_GB_EUC_V_0[];
extern const uint16_t g_FXCMap_GBpc_EUC_H_0[];
extern const uint16_t g_FXCMap_GBpc_EUC_V_0[];
extern const uint16_t g_FXCMap_GBK_EUC_H_2[];
extern const uint16_t g_FXCMap_GBK_EUC_V_2[];
extern const uint16_t g_FXCMap_GBKp_EUC_H_2[];
extern const uint16_t g_FXCMap_GBKp_EUC_V_2[];
extern const uint16_t g_FXCMap_GBK2K_H_5[];
extern const FXCMap_DWordCIDMap g_FXCMap_GBK2K_H_5_DWord[];
extern const uint16_t g_FXCMap_GBK2K_V_5[];
extern const uint16_t g_FXCMap_UniGB_UCS2_H_4[];
extern const uint16_t g_FXCMap_UniGB_UCS2_V_4[];
extern const uint16_t g_FXCMap_GB1CID2Unicode_5[30284];
extern const FXCMap_CMap g_FXCMap_GB1_cmapi[];
extern const size_t g_FXCMap_GB1_cmapi_size;

#endif // CORE_FPDFAPI_CMAPI_GB1_CMAPI_GB1_H_
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_CMAPS_JAPAN1_CMAPS_JAPAN1_H  
#define CORE_FPDFAPI_CMAPS_JAPAN1_CMAPS_JAPAN1_H
```

```
#include "core/fpdfapi/cmmaps/cmap_int.h"
```

```
extern const uint16_t g_FXCMAP_83pv_RKSJ_H_1[];  
extern const uint16_t g_FXCMAP_90ms_RKSJ_H_2[];  
extern const uint16_t g_FXCMAP_90ms_RKSJ_V_2[];  
extern const uint16_t g_FXCMAP_90msp_RKSJ_H_2[];  
extern const uint16_t g_FXCMAP_90msp_RKSJ_V_2[];  
extern const uint16_t g_FXCMAP_90pv_RKSJ_H_1[];  
extern const uint16_t g_FXCMAP_Add_RKSJ_H_1[];  
extern const uint16_t g_FXCMAP_Add_RKSJ_V_1[];  
extern const uint16_t g_FXCMAP_EUC_H_1[];  
extern const uint16_t g_FXCMAP_EUC_V_1[];  
extern const uint16_t g_FXCMAP_Ext_RKSJ_H_2[];  
extern const uint16_t g_FXCMAP_Ext_RKSJ_V_2[];  
extern const uint16_t g_FXCMAP_H_1[];  
extern const uint16_t g_FXCMAP_V_1[];  
extern const uint16_t g_FXCMAP_UniJIS_UCS2_H_4[];  
extern const uint16_t g_FXCMAP_UniJIS_UCS2_V_4[];  
extern const uint16_t g_FXCMAP_UniJIS_UCS2_HW_H_4[];  
extern const uint16_t g_FXCMAP_UniJIS_UCS2_HW_V_4[];  
extern const uint16_t g_FXCMAP_UniJIS_UTF16_H_0[];  
extern const uint16_t g_FXCMAP_UniJIS_UTF16_H_0_DWord[];  
extern const uint16_t g_FXCMAP_UniJIS_UTF16_V_0[];  
extern const uint16_t g_FXCMAP_Japan1CID2Unicode_4[15444];  
extern const FXCMAP_CMap g_FXCMAP_Japan1_cmmaps[];  
extern const size_t g_FXCMAP_Japan1_cmmaps_size;
```

```
#endif // CORE_FPDFAPI_CMAPS_JAPAN1_CMAPS_JAPAN1_H
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_CMAPS_KOREA1_CMAPS_KOREA1_H  
#define CORE_FPDFAPI_CMAPS_KOREA1_CMAPS_KOREA1_H
```

```
#include "core/fpdfapi/cmmaps/cmap_int.h"
```

```
extern const uint16_t g_FXCMAP_KSC_EUC_H_0[];  
extern const uint16_t g_FXCMAP_KSC_EUC_V_0[];  
extern const uint16_t g_FXCMAP_KSCms_UHC_H_1[];  
extern const uint16_t g_FXCMAP_KSCms_UHC_V_1[];  
extern const uint16_t g_FXCMAP_KSCms_UHC_HW_H_1[];  
extern const uint16_t g_FXCMAP_KSCms_UHC_HW_V_1[];  
extern const uint16_t g_FXCMAP_KSCpc_EUC_H_0[];  
extern const uint16_t g_FXCMAP_UniKS_UCS2_H_1[];  
extern const uint16_t g_FXCMAP_UniKS_UCS2_V_1[];  
extern const uint16_t g_FXCMAP_UniKS_UTF16_H_0[];  
extern const uint16_t g_FXCMAP_Korea1CID2Unicode_2[18352];  
extern const FXCMAP_CMap g_FXCMAP_Korea1_cmmaps[];  
extern const size_t g_FXCMAP_Korea1_cmmaps_size;
```

```
#endif // CORE_FPDFAPI_CMAPS_KOREA1_CMAPS_KOREA1_H
```

third\_party/pdfium/core/fpdfapi/edit/cpdf\_contentstream\_write\_utils.h

Tue Nov 12 15:18:17 20

```
// Copyright 2019 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FPDFAPI_EDIT_CPDF_CONTENTSTREAM_WRITE_UTILS_H_  
#define CORE_FPDFAPI_EDIT_CPDF_CONTENTSTREAM_WRITE_UTILS_H_
```

```
#include <ostream>
```

```
#include "core/fxcrt/fx_coordinates.h"
```

```
std::ostream& WriteFloat(std::ostream& stream, float value);  
std::ostream& operator<<(std::ostream& ar, const CFX_Matrix& matrix);  
std::ostream& operator<<(std::ostream& ar, const CFX_PointF& point);
```

```
#endif // CORE_FPDFAPI_EDIT_CPDF_CONTENTSTREAM_WRITE_UTILS_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.  
  
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_EDIT_CPDF_CREATOR_H  
#define CORE_FPDFAPI_EDIT_CPDF_CREATOR_H
```

```
#include <map>  
#include <memory>  
#include <vector>
```

```
#include "core/fxcrt/fx_stream.h"  
#include "core/fxcrt/retain_ptr.h"  
#include "core/fxcrt/unowned_ptr.h"
```

```
class CPDF_Array;  
class CPDF_CryptoHandler;  
class CPDF_SecurityHandler;  
class CPDF_Dictionary;  
class CPDF_Document;  
class CPDF_Object;  
class CPDF_Parser;
```

```
#define FPDFCREATE_INCREMENTAL 1  
#define FPDFCREATE_NO_ORIGINAL 2
```

```
class CPDF_Creator {  
public:  
    CPDF_Creator(CPDF_Document* pDoc,  
                const RetainPtr<IFX_RetainableWriteStream>& archive);  
    ~CPDF_Creator();  
  
    void RemoveSecurity();  
    bool Create(uint32_t flags);  
    bool SetFileVersion(int32_t fileVersion);
```

```
private:
```

```
enum class Stage {  
    kInvalid = -1,  
    kInit0 = 0,  
    kWriteHeader10 = 10,  
    kWriteIncremental15 = 15,  
    kInitWriteObjs20 = 20,  
    kWriteOldObjs21 = 21,  
    kInitWriteNewObjs25 = 25,  
    kWriteNewObjs26 = 26,  
    kWriteEncryptDict27 = 27,  
    kInitWriteXRefs80 = 80,  
    kWriteXrefsNotIncremental81 = 81,  
    kWriteXrefsIncremental82 = 82,  
    kWriteTrailerAndFinish90 = 90,  
    kComplete100 = 100,  
};
```

```
bool Continue();  
void Clear();
```

```
void InitNewObjNumOffsets();  
void InitID();
```

```
CPDF_Creator::Stage WriteDoc_Stage1();
```

```
CPDF_Creator::Stage WriteDoc_Stage2();
CPDF_Creator::Stage WriteDoc_Stage3();
CPDF_Creator::Stage WriteDoc_Stage4();

bool WriteOldIndirectObject(uint32_t objnum);
bool WriteOldObjs();
bool WriteNewObjs();
bool WriteIndirectObj(uint32_t objnum, const CPDF_Object* pObj);

CPDF_CryptoHandler* GetCryptoHandler();

UnownedPtr<CPDF_Document> const m_pDocument;
UnownedPtr<const CPDF_Parser> const m_pParser;
RetainPtr<const CPDF_Dictionary> m_pEncryptDict;
RetainPtr<CPDF_Dictionary> m_pNewEncryptDict;
RetainPtr<CPDF_SecurityHandler> m_pSecurityHandler;
RetainPtr<const CPDF_Object> m_pMetadata;
uint32_t m_dwLastObjNum;
std::unique_ptr<IFX_ArchiveStream> m_Archive;
FX_FILESIZE m_SavedOffset = 0;
Stage m_iStage = Stage::kInvalid;
uint32_t m_CurObjNum = 0;
FX_FILESIZE m_XrefStart = 0;
std::map<uint32_t, FX_FILESIZE> m_ObjectOffsets;
std::vector<uint32_t> m_NewObjNumArray; // Sorted, ascending.
RetainPtr<CPDF_Array> m_pIDArray;
int32_t m_FileVersion = 0;
bool m_bSecurityChanged = false;
bool m_IsIncremental = false;
bool m_IsOriginal = false;
};

#endif // CORE_FPDFAPI_EDIT_CPDF_CREATOR_H_
```

```

// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_EDIT_CPDF_PAGECONTENTGENERATOR_H_
#define CORE_FPDFAPI_EDIT_CPDF_PAGECONTENTGENERATOR_H_

#include <map>
#include <memory>
#include <sstream>
#include <vector>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_ContentMarks;
class CPDF_Document;
class CPDF_ImageObject;
class CPDF_Object;
class CPDF_PageObject;
class CPDF_PageObjectHolder;
class CPDF_PathObject;
class CPDF_TextObject;

class CPDF_PageContentGenerator {
public:
  explicit CPDF_PageContentGenerator(CPDF_PageObjectHolder* pObjHolder);
  ~CPDF_PageContentGenerator();

  void GenerateContent();
  bool ProcessPageObjects(std::ostringstream* buf);

private:
  friend class CPDF_PageContentGeneratorTest;

  void ProcessPageObject(std::ostringstream* buf, CPDF_PageObject* pPageObj);
  void ProcessPath(std::ostringstream* buf, CPDF_PathObject* pPathObj);
  void ProcessImage(std::ostringstream* buf, CPDF_ImageObject* pImageObj);
  void ProcessGraphics(std::ostringstream* buf, CPDF_PageObject* pPageObj);
  void ProcessDefaultGraphics(std::ostringstream* buf);
  void ProcessText(std::ostringstream* buf, CPDF_TextObject* pTextObj);
  ByteString GetOrCreateDefaultGraphics() const;
  ByteString RealizeResource(const CPDF_Object* pResource,
                             const ByteString& bsType) const;
  const CPDF_ContentMarks* ProcessContentMarks(std::ostringstream* buf,
                                                const CPDF_PageObject* pPageObj,
                                                const CPDF_ContentMarks* pPrev);
  void FinishMarks(std::ostringstream* buf,
                  const CPDF_ContentMarks* pContentMarks);

  // Returns a map from content stream index to new stream data. Unmodified
  // streams are not touched.
  std::map<int32_t, std::unique_ptr<std::ostringstream>>
  GenerateModifiedStreams();

  // Add buffer as a stream in page's 'Contents'
  void UpdateContentStreams(
    std::map<int32_t, std::unique_ptr<std::ostringstream>>* new_stream_data);

  // Set the stream index of all page objects with stream index ==

```

```
// |CPDF_PageObject::kNoContentStream|. These are new objects that had not  
// been parsed from or written to any content stream yet.
```

```
void UpdateStreamlessPageObjects(int new_content_stream_index);
```

```
UnownedPtr<CPDF_PageObjectHolder> const m_pObjHolder;
```

```
UnownedPtr<CPDF_Document> const m_pDocument;
```

```
std::vector<UnownedPtr<CPDF_PageObject>> m_pageObjects;
```

```
};
```

```
#endif // CORE_FPDFAPI_EDIT_CPDF_PAGECONTENTGENERATOR_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

#ifndef CORE_FPDFAPI_EDIT_CPDF_PAGECONTENTMANAGER_H_
#define CORE_FPDFAPI_EDIT_CPDF_PAGECONTENTMANAGER_H_

#include <set>
#include <sstream>

#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Array;
class CPDF_Document;
class CPDF_Object;
class CPDF_Stream;
class CPDF_PageObjectHolder;

class CPDF_PageContentManager {
public:
  explicit CPDF_PageContentManager(const CPDF_PageObjectHolder* obj_holder);
  ~CPDF_PageContentManager();

  // Gets the Content stream at a given index. If Contents is a single stream
  // rather than an array, it is considered to be at index 0.
  CPDF_Stream* GetStreamByIndex(size_t stream_index);

  // Adds a new Content stream. Its index in the array will be returned, or 0
  // if Contents is not an array, but only a single stream.
  size_t AddStream(std::ostream* buf);

  // Schedule the removal of the Content stream at a given index. It will be
  // removed when ExecuteScheduledRemovals() is called.
  void ScheduleRemoveStreamByIndex(size_t stream_index);

  // Remove all Content streams for which ScheduleRemoveStreamByIndex() was
  // called. Update the content stream of all page objects with the shifted
  // indexes.
  void ExecuteScheduledRemovals();

private:
  UnownedPtr<const CPDF_PageObjectHolder> const obj_holder_;
  UnownedPtr<CPDF_Document> const doc_;
  RetainPtr<CPDF_Array> contents_array_;
  RetainPtr<CPDF_Stream> contents_stream_;
  std::set<size_t> streams_to_remove_;
};

#endif // CORE_FPDFAPI_EDIT_CPDF_PAGECONTENTMANAGER_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FPDFAPI_EDIT_CPDF_STRINGARCHIVESTREAM_H_  
#define CORE_FPDFAPI_EDIT_CPDF_STRINGARCHIVESTREAM_H_  
  
#include "core/fxcrt/fx_stream.h"  
  
class CPDF_StringArchiveStream final : public IFX_ArchiveStream {  
public:  
    explicit CPDF_StringArchiveStream(std::ostringstream* stream);  
    ~CPDF_StringArchiveStream() override;  
  
    // IFX_ArchiveStream  
    bool WriteByte(uint8_t byte) override;  
    bool WriteDWord(uint32_t i) override;  
    FX_FILESIZE CurrentOffset() const override;  
    bool WriteBlock(const void* pData, size_t size) override;  
    bool WriteString(ByteStringView str) override;  
  
private:  
    std::ostringstream* stream_;  
};  
  
#endif // CORE_FPDFAPI_EDIT_CPDF_STRINGARCHIVESTREAM_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CFX_CTTGSUBTABLE_H
#define CORE_FPDFAPI_FONT_CFX_CTTGSUBTABLE_H

#include <stdint.h>

#include <memory>
#include <set>
#include <vector>

#include "core/fxge/fx_freetype.h"

class CFX_CTTGSUBTable {
public:
    explicit CFX_CTTGSUBTable(FT_Bytes gsub);
    ~CFX_CTTGSUBTable();

    uint32_t GetVerticalGlyph(uint32_t glyphnum) const;

private:
    struct TLangSysRecord {
        TLangSysRecord();
        ~TLangSysRecord();

        uint32_t LangSysTag;
        uint16_t LookupOrder;
        uint16_t ReqFeatureIndex;
        std::vector<uint16_t> FeatureIndices;
    };

    struct TScriptRecord {
        TScriptRecord();
        ~TScriptRecord();

        uint32_t ScriptTag;
        uint16_t DefaultLangSys;
        std::vector<TLangSysRecord> LangSysRecords;
    };

    struct TFeatureRecord {
        TFeatureRecord();
        ~TFeatureRecord();

        uint32_t FeatureTag;
        uint16_t FeatureParams;
        std::vector<uint16_t> LookupListIndices;
    };

    struct TRangeRecord {
        TRangeRecord();

        uint16_t Start;
        uint16_t End;
        uint16_t StartCoverageIndex;
    };

    struct TCoverageFormatBase {
        virtual ~TCoverageFormatBase() = default;
    };
};
```

```
    uint16_t CoverageFormat;
};

struct TCoverageFormat1 final : public TCoverageFormatBase {
    TCoverageFormat1();
    ~TCoverageFormat1() override;

    std::vector<uint16_t> GlyphArray;
};

struct TCoverageFormat2 final : public TCoverageFormatBase {
    TCoverageFormat2();
    ~TCoverageFormat2() override;

    std::vector<TRangeRecord> RangeRecords;
};

struct TDevice {
    TDevice() : StartSize(0), EndSize(0), DeltaFormat(0) {}

    uint16_t StartSize;
    uint16_t EndSize;
    uint16_t DeltaFormat;
};

struct TSubTableBase {
    TSubTableBase();
    virtual ~TSubTableBase();

    std::unique_ptr<TCoverageFormatBase> Coverage;
    uint16_t SubstFormat;
};

struct TSubTable1 final : public TSubTableBase {
    TSubTable1();
    ~TSubTable1() override;

    int16_t DeltaGlyphID;
};

struct TSubTable2 final : public TSubTableBase {
    TSubTable2();
    ~TSubTable2() override;

    std::vector<uint16_t> Substitutes;
};

struct TLookup {
    TLookup();
    ~TLookup();

    uint16_t LookupType;
    uint16_t LookupFlag;
    std::vector<std::unique_ptr<TSubTableBase>> SubTables;
};

bool LoadGSUBTable(FT_Bytes gsub);
bool Parse(FT_Bytes scriptlist, FT_Bytes featurelist, FT_Bytes lookuplist);
void ParseScriptList(FT_Bytes raw);
void ParseScript(FT_Bytes raw, TScriptRecord* rec);
void ParseLangSys(FT_Bytes raw, TLangSysRecord* rec);
void ParseFeatureList(FT_Bytes raw);
void ParseFeature(FT_Bytes raw, TFeatureRecord* rec);
```

```
void ParseLookupList(FT_Bytes raw);
void ParseLookup(FT_Bytes raw, TLookup* rec);
std::unique_ptr<TCoverageFormatBase> ParseCoverage(FT_Bytes raw);
void ParseCoverageFormat1(FT_Bytes raw, TCoverageFormat1* rec);
void ParseCoverageFormat2(FT_Bytes raw, TCoverageFormat2* rec);
void ParseSingleSubst(FT_Bytes raw, std::unique_ptr<TSubTableBase>* rec);
void ParseSingleSubstFormat1(FT_Bytes raw, TSubTable1* rec);
void ParseSingleSubstFormat2(FT_Bytes raw, TSubTable2* rec);

bool GetVerticalGlyphSub(const TFeatureRecord& feature,
                        uint32_t glyphnum,
                        uint32_t* vglyphnum) const;
bool GetVerticalGlyphSub2(const TLookup& lookup,
                        uint32_t glyphnum,
                        uint32_t* vglyphnum) const;
int GetCoverageIndex(TCoverageFormatBase* Coverage, uint32_t g) const;

uint8_t GetUInt8(FT_Bytes& p) const;
int16_t GetInt16(FT_Bytes& p) const;
uint16_t GetUInt16(FT_Bytes& p) const;
int32_t GetInt32(FT_Bytes& p) const;
uint32_t GetUInt32(FT_Bytes& p) const;

std::set<uint32_t> m_featureSet;
std::vector<TScriptRecord> ScriptList;
std::vector<TFeatureRecord> FeatureList;
std::vector<TLookup> LookupList;
};

#endif // CORE_FPDFAPI_FONT_CFX_CTTGSUBTABLE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CFX_STOCKFONTARRAY_H_
#define CORE_FPDFAPI_FONT_CFX_STOCKFONTARRAY_H_

#include <memory>

#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/cfx_fontmapper.h"

class CPDF_Font;

class CFX_StockFontArray {
public:
  CFX_StockFontArray();
  ~CFX_StockFontArray();

  RetainPtr<CPDF_Font> GetFont(CFX_FontMapper::StandardFont index) const;
  void SetFont(CFX_FontMapper::StandardFont index,
              const RetainPtr<CPDF_Font>& pFont);

private:
  RetainPtr<CPDF_Font> m_StockFonts[14];
};

#endif // CORE_FPDFAPI_FONT_CFX_STOCKFONTARRAY_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CPDF_CID2UNICODEMAP_H_
#define CORE_FPDFAPI_FONT_CPDF_CID2UNICODEMAP_H_

#include "core/fpdfapi/font/cpdf_cidfont.h"
#include "third_party/base/span.h"

class CPDF_CID2UnicodeMap {
public:
  explicit CPDF_CID2UnicodeMap(CIDSet charset);
  ~CPDF_CID2UnicodeMap();

  bool IsLoaded() const;
  wchar_t UnicodeFromCID(uint16_t cid) const;

private:
  const CIDSet m_Charset;
  const pdfium::span<const uint16_t> m_pEmbeddedMap;
};

#endif // CORE_FPDFAPI_FONT_CPDF_CID2UNICODEMAP_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifdef CORE_FPDFAPI_FONT_CPDF_CIDFONT_H
#define CORE_FPDFAPI_FONT_CPDF_CIDFONT_H

#include <memory>
#include <vector>

#include "core/fpdfapi/font/cpdf_font.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

enum CIDSet : uint8_t {
    CIDSET_UNKNOWN,
    CIDSET_GB1,
    CIDSET_CNS1,
    CIDSET_JAPAN1,
    CIDSET_KOREA1,
    CIDSET_UNICODE,
    CIDSET_NUM_SETS
};

class CFX_CTTGSUBTable;
class CPDF_Array;
class CPDF_CID2UnicodeMap;
class CPDF_CMap;
class CPDF_StreamAcc;

class CPDF_CIDFont final : public CPDF_Font {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    ~CPDF_CIDFont() override;

    static float CIDTransformToFloat(uint8_t ch);

    // CPDF_Font:
    bool IsCIDFont() const override;
    const CPDF_CIDFont* AsCIDFont() const override;
    CPDF_CIDFont* AsCIDFont() override;
    int GlyphFromCharCode(uint32_t charcode, bool* pVertGlyph) override;
    uint32_t GetCharWidthF(uint32_t charcode) override;
    FX_RECT GetCharBBox(uint32_t charcode) override;
    uint32_t GetNextChar(ByteStringView pString, size_t* pOffset) const override;
    size_t CountChar(ByteStringView pString) const override;
    int AppendChar(char* str, uint32_t charcode) const override;
    bool IsVertWriting() const override;
    bool IsUnicodeCompatible() const override;
    bool Load() override;
    WideString UnicodeFromCharCode(uint32_t charcode) const override;
    uint32_t CharCodeFromUnicode(wchar_t Unicode) const override;

    uint16_t CIDFromCharCode(uint32_t charcode) const;
    const uint8_t* GetCIDTransform(uint16_t CID) const;
    short GetVertWidth(uint16_t CID) const;
    void GetVertOrigin(uint16_t CID, short& vx, short& vy) const;
};
```

```
    int GetCharSize(uint32_t charcode) const;

private:
    CPDF_CIDFont(CPDF_Document* pDocument, CPDF_Dictionary* pFontDict);

    void LoadGB2312();
    int GetGlyphIndex(uint32_t unicodeb, bool* pVertGlyph);
    int GetVerticalGlyph(int index, bool* pVertGlyph);
    void LoadMetricsArray(const CPDF_Array* pArray,
                          std::vector<uint32_t>* result,
                          int nElements);

    void LoadSubstFont();
    wchar_t GetUnicodeFromCharCode(uint32_t charcode) const;

    RetainPtr<const CPDF_CMap> m_pCMap;
    UnownedPtr<const CPDF_CID2UnicodeMap> m_pCID2UnicodeMap;
    RetainPtr<CPDF_StreamAcc> m_pStreamAcc;
    std::unique_ptr<CFX_CTTGSUBTable> m_pTTGSUBTable;
    bool m_bType1 = false;
    bool m_bCIDIsGID = false;
    bool m_bAnsiWidthsFixed = false;
    bool m_bAdobeCourierStd = false;
    CIDSet m_Charset = CIDSET_UNKNOWN;
    uint16_t m_DefaultWidth = 1000;
    short m_DefaultVY = 880;
    short m_DefaultW1 = -1000;
    std::vector<uint32_t> m_WidthList;
    std::vector<uint32_t> m_VertMetrics;
    FX_RECT m_CharBBox[256];
};

#endif // CORE_FPDFAPI_FONT_CPDF_CIDFONT_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CPDF_CMAP_H_
#define CORE_FPDFAPI_FONT_CPDF_CMAP_H_

#include <vector>

#include "core/fpdfapi/font/cpdf_cidfont.h"
#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/span.h"

struct FXCMAP_CMap;

enum CIDCoding : uint8_t {
  CIDCODING_UNKNOWN = 0,
  CIDCODING_GB,
  CIDCODING_BIG5,
  CIDCODING_JIS,
  CIDCODING_KOREA,
  CIDCODING_UCS2,
  CIDCODING_CID,
  CIDCODING_UTF16,
};

class CPDF_CMap final : public Retainable {
public:
  enum CodingScheme : uint8_t {
    OneByte,
    TwoBytes,
    MixedTwoBytes,
    MixedFourBytes
  };

  struct CodeRange {
    size_t m_CharSize;
    uint8_t m_Lower[4];
    uint8_t m_Upper[4];
  };

  struct CIDRange {
    uint32_t m_StartCode;
    uint32_t m_EndCode;
    uint16_t m_StartCID;
  };

  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  bool IsLoaded() const { return m_bLoaded; }
  bool IsVertWriting() const { return m_bVertical; }

  uint16_t CIDFromCharCode(uint32_t charcode) const;

  int GetCharSize(uint32_t charcode) const;
  uint32_t GetNextChar(ByteStringView pString, size_t* pOffset) const;
  size_t CountChar(ByteStringView pString) const;
  int AppendChar(char* str, uint32_t charcode) const;

  void SetVertical(bool vert) { m_bVertical = vert; }
};
```

```
void SetCodingScheme(CodingScheme scheme) { m_CodingScheme = scheme; }
void SetAdditionalMappings(std::vector<CIDRange> mappings);
void SetMixedFourByteLeadingRanges(std::vector<CodeRange> ranges);

int GetCoding() const { return m_Coding; }
const FXCMAP_CMap* GetEmbedMap() const { return m_pEmbedMap.Get(); }
CIDSet GetCharset() const { return m_Charset; }
void SetCharset(CIDSet set) { m_Charset = set; }

void SetDirectCharcodeToCIDTable(size_t idx, uint16_t val) {
    m_DirectCharcodeToCIDTable[idx] = val;
}
bool IsDirectCharcodeToCIDTableIsEmpty() const {
    return m_DirectCharcodeToCIDTable.empty();
}

private:
explicit CPDF_CMap(const ByteString& bsPredefinedName);
explicit CPDF_CMap(pdfium::span<const uint8_t> spEmbeddedData);
~CPDF_CMap() override;

bool m_bLoaded = false;
bool m_bVertical = false;
CIDSet m_Charset = CIDSET_UNKNOWN;
CodingScheme m_CodingScheme = TwoBytes;
int m_Coding = CIDCODING_UNKNOWN;
std::vector<bool> m_MixedTwoByteLeadingBytes;
std::vector<CodeRange> m_MixedFourByteLeadingRanges;
std::vector<uint16_t> m_DirectCharcodeToCIDTable;
std::vector<CIDRange> m_AdditionalCharcodeToCIDMappings;
UnownedPtr<const FXCMAP_CMap> m_pEmbedMap;
};

#endif // CORE_FPDFAPI_FONT_CPDF_CMAP_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CPDF_CMAPMANAGER_H_
#define CORE_FPDFAPI_FONT_CPDF_CMAPMANAGER_H_

#include <map>
#include <memory>

#include "core/fpdfapi/font/cpdf_cidfont.h"
#include "core/fxcrt/bytestring.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_CMapManager {
public:
  CPDF_CMapManager();
  ~CPDF_CMapManager();

  RetainPtr<const CPDF_CMap> GetPredefinedCMap(const ByteString& name);
  CPDF_CID2UnicodeMap* GetCID2UnicodeMap(CIDSet charset);

private:
  std::map<ByteString, RetainPtr<const CPDF_CMap>> m_CMaps;
  std::unique_ptr<CPDF_CID2UnicodeMap> m_CID2UnicodeMaps[CIDSET_NUM_SETS];
};

#endif // CORE_FPDFAPI_FONT_CPDF_CMAPMANAGER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CPDF_CMAPPARSER_H_
#define CORE_FPDFAPI_FONT_CPDF_CMAPPARSER_H_

#include <utility>
#include <vector>

#include "core/fpdfapi/font/cpdf_cidfont.h"
#include "core/fpdfapi/font/cpdf_cmap.h"
#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/optional.h"

class CPDF_CMapParser {
public:
  explicit CPDF_CMapParser(CPDF_CMap* pCMap);
  ~CPDF_CMapParser();

  void ParseWord(ByteStringView word);

  static CIDSet CharsetFromOrdering(ByteStringView ordering);

private:
  friend class cpdf_cmapparser_GetCode_Test;
  friend class cpdf_cmapparser_GetCodeRange_Test;

  enum Status {
    kStart,
    kProcessingCidChar,
    kProcessingCidRange,
    kProcessingRegistry,
    kProcessingOrdering,
    kProcessingSupplement,
    kProcessingWMode,
    kProcessingCodeSpaceRange,
  };

  void HandleCid(ByteStringView word);
  void HandleCodeSpaceRange(ByteStringView word);

  static uint32_t GetCode(ByteStringView word);
  static Optional<CPDF_CMap::CodeRange> GetCodeRange(ByteStringView first,
                                                       ByteStringView second);

  Status m_Status = kStart;
  int m_CodeSeq = 0;
  UnownedPtr<CPDF_CMap> const m_pCMap;
  std::vector<CPDF_CMap::CodeRange> m_Ranges;
  std::vector<CPDF_CMap::CodeRange> m_PendingRanges;
  std::vector<CPDF_CMap::CIDRange> m_AdditionalCharcodeToCIDMappings;
  ByteString m_LastWord;
  uint32_t m_CodePoints[4];
};

#endif // CORE_FPDFAPI_FONT_CPDF_CMAPPARSER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CPDF_FONTENCODING_H_
#define CORE_FPDFAPI_FONT_CPDF_FONTENCODING_H_

#include <memory>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/string_pool_template.h"
#include "core/fxcrt/weak_ptr.h"

#define PDFFONT_ENCODING_BUILTIN 0
#define PDFFONT_ENCODING_WINANSI 1
#define PDFFONT_ENCODING_MACROMAN 2
#define PDFFONT_ENCODING_MACEXPERT 3
#define PDFFONT_ENCODING_STANDARD 4
#define PDFFONT_ENCODING_ADOBE_SYMBOL 5
#define PDFFONT_ENCODING_ZAPFDINGBATS 6
#define PDFFONT_ENCODING_PDFDOC 7
#define PDFFONT_ENCODING_MS_SYMBOL 8

uint32_t FT_CharCodeFromUnicode(int encoding, wchar_t unicode);
wchar_t FT_UnicodeFromCharCode(int encoding, uint32_t charcode);

const uint16_t* PDF_UnicodesForPredefinedCharSet(int encoding);
const char* PDF_CharNameFromPredefinedCharSet(int encoding, uint8_t charcode);

class CPDF_Object;

class CPDF_FontEncoding {
public:
    static constexpr size_t kEncodingTableSize = 256;

    explicit CPDF_FontEncoding(int PredefinedEncoding);

    bool IsIdentical(const CPDF_FontEncoding* pAnother) const;

    wchar_t UnicodeFromCharCode(uint8_t charcode) const {
        return m_Unicodes[charcode];
    }
    int CharCodeFromUnicode(wchar_t unicode) const;

    void SetUnicode(uint8_t charcode, wchar_t unicode) {
        m_Unicodes[charcode] = unicode;
    }

    RetainPtr<CPDF_Object> Realize(WeakPtr<ByteStringPool> pPool) const;

private:
    wchar_t m_Unicodes[kEncodingTableSize];
};

#endif // CORE_FPDFAPI_FONT_CPDF_FONTENCODING_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_FONT_CPDF_FONTGLOBS_H_  
#define CORE_FPDFAPI_FONT_CPDF_FONTGLOBS_H_
```

```
#include <map>  
#include <memory>
```

```
#include "core/fpdfapi/cmmaps/cmap_int.h"  
#include "core/fpdfapi/font/cpdf_cmapmanager.h"  
#include "core/fxcrt/retain_ptr.h"  
#include "core/fxge/cfx_fontmapper.h"  
#include "third_party/base/span.h"
```

```
class CFX_StockFontArray;
```

```
class CPDF_FontGlobals {  
public:
```

```
    // Per-process singleton which must be managed by callers.
```

```
    static void Create();  
    static void Destroy();  
    static CPDF_FontGlobals* GetInstance();
```

```
    // Caller must load the maps before using font globals.
```

```
    void LoadEmbeddedMaps();
```

```
    void Clear(CPDF_Document* pDoc);  
    RetainPtr<CPDF_Font> Find(CPDF_Document* pDoc,  
                             CFX_FontMapper::StandardFont index);
```

```
    void Set(CPDF_Document* pDoc,  
            CFX_FontMapper::StandardFont index,  
            const RetainPtr<CPDF_Font>& pFont);
```

```
    void SetEmbeddedCharset(size_t idx, pdfium::span<const FXCMAP_CMap> map) {  
        m_EmbeddedCharsets[idx] = map;  
    }
```

```
    pdfium::span<const FXCMAP_CMap> GetEmbeddedCharset(size_t idx) const {  
        return m_EmbeddedCharsets[idx];  
    }
```

```
    void SetEmbeddedToUnicode(size_t idx, pdfium::span<const uint16_t> map) {  
        m_EmbeddedToUnicodes[idx] = map;  
    }
```

```
    pdfium::span<const uint16_t> GetEmbeddedToUnicode(size_t idx) {  
        return m_EmbeddedToUnicodes[idx];  
    }
```

```
    CPDF_CMapManager* GetCMapManager() { return &m_CMapManager; }
```

```
private:
```

```
    CPDF_FontGlobals();  
    ~CPDF_FontGlobals();
```

```
    void LoadEmbeddedGB1CMaps();  
    void LoadEmbeddedCNS1CMaps();  
    void LoadEmbeddedJapan1CMaps();  
    void LoadEmbeddedKorealCMaps();
```

```
    CPDF_CMapManager m_CMapManager;  
    pdfium::span<const FXCMAP_CMap> m_EmbeddedCharsets[CIDSET_NUM_SETS];
```

```
pdfium::span<const uint16_t> m_EmbeddedToUnicodes[CIDSET_NUM_SETS];
std::map<CPDF_Document*, std::unique_ptr<CFX_StockFontArray>> m_StockMap;
};

#endif // CORE_FPDFAPI_FONT_CPDF_FONTGLOBALS_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CPDF_FONT_H_
#define CORE_FPDFAPI_FONT_CPDF_FONT_H_

#include <memory>
#include <utility>
#include <vector>

#include "build/build_config.h"
#include "core/fpdfapi/parser/cpdf_dictionary.h"
#include "core/fpdfapi/parser/cpdf_stream_acc.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/cfx_font.h"

class CFX_DIBitmap;
class CFX_SubstFont;
class CPDF_CIDFont;
class CPDF_Document;
class CPDF_Object;
class CPDF_TrueTypeFont;
class CPDF_Type1Font;
class CPDF_Type3Char;
class CPDF_Type3Font;
class CPDF_ToUnicodeMap;

class CPDF_Font : public Retainable, public Observable {
public:
  // Callback mechanism for Type3 fonts to get pixels from forms.
  class FormIface {
public:
    virtual ~FormIface() {}

    virtual void ParseContentForType3Char(CPDF_Type3Char* pChar) = 0;
    virtual bool HasPageObjects() const = 0;
    virtual CFX_FloatRect CalcBoundingBox() const = 0;
    virtual Optional<std::pair<RetainPtr<CFX_DIBitmap>, CFX_Matrix>>
        GetBitmapAndMatrixFromSoleImageOfForm() const = 0;
  };

  // Callback mechanism for Type3 fonts to get new forms from upper layers.
  class FormFactoryIface {
public:
    virtual ~FormFactoryIface() {}

    virtual std::unique_ptr<FormIface> CreateForm(
        CPDF_Document* pDocument,
        CPDF_Dictionary* pPageResources,
        CPDF_Stream* pFormStream) = 0;
  };

  static const uint32_t kInvalidCharCode = static_cast<uint32_t>(-1);

  // |pFactory| only required for Type3 fonts.
  static RetainPtr<CPDF_Font> Create(CPDF_Document* pDoc,
```

```

        CPDF_Dictionary* pFontDict,
        FormFactoryIface* pFactory);
static RetainPtr<CPDF_Font> GetStockFont(CPDF_Document* pDoc,
        ByteStringView fontname);

~CPDF_Font() override;

virtual bool IsType1Font() const;
virtual bool IsTrueTypeFont() const;
virtual bool IsType3Font() const;
virtual bool IsCIDFont() const;
virtual const CPDF_Type1Font* AsType1Font() const;
virtual CPDF_Type1Font* AsType1Font();
virtual const CPDF_TrueTypeFont* AsTrueTypeFont() const;
virtual CPDF_TrueTypeFont* AsTrueTypeFont();
virtual const CPDF_Type3Font* AsType3Font() const;
virtual CPDF_Type3Font* AsType3Font();
virtual const CPDF_CIDFont* AsCIDFont() const;
virtual CPDF_CIDFont* AsCIDFont();

virtual void WillBeDestroyed();
virtual bool IsVertWriting() const;
virtual bool IsUnicodeCompatible() const;
virtual uint32_t GetNextChar(ByteStringView pString, size_t* pOffset) const;
virtual size_t CountChar(ByteStringView pString) const;
virtual int AppendChar(char* buf, uint32_t charcode) const;
virtual int GlyphFromCharCode(uint32_t charcode, bool* pVertGlyph) = 0;
#if defined(OS_MACOSX)
virtual int GlyphFromCharCodeExt(uint32_t charcode);
#endif
virtual WideString UnicodeFromCharCode(uint32_t charcode) const;
virtual uint32_t CharCodeFromUnicode(wchar_t Unicode) const;
virtual bool HasFontWidths() const;

ByteString GetBaseFontName() const { return m_BaseFontName; }
CFX_SubstFont* GetSubstFont() const { return m_Font.GetSubstFont(); }
bool IsEmbedded() const { return IsType3Font() || m_pFontFile != nullptr; }
CPDF_Dictionary* GetFontDict() const { return m_pFontDict.Get(); }
void ClearFontDict() { m_pFontDict = nullptr; }
bool IsStandardFont() const;
bool HasFace() const { return !!m_Font.GetFaceRec(); }
void AppendChar(ByteString* str, uint32_t charcode) const;

const FX_RECT& GetFontBBox() const { return m_FontBBox; }
int GetTypeAscent() const { return m_Ascent; }
int GetTypeDescent() const { return m_Descent; }
uint32_t GetStringWidth(ByteStringView pString);
uint32_t FallbackFontFromCharCode(uint32_t charcode);
int FallbackGlyphFromCharCode(int fallbackFont, uint32_t charcode);
int GetFontFlags() const { return m_Flags; }
int GetFontWeight() const;

virtual uint32_t GetCharWidthF(uint32_t charcode) = 0;
virtual FX_RECT GetCharBBox(uint32_t charcode) = 0;

// Can return nullptr for stock Type1 fonts. Always returns non-null for other
// font types.
CPDF_Document* GetDocument() const { return m_pDocument.Get(); }

CFX_Font* GetFont() { return &m_Font; }
const CFX_Font* GetFont() const { return &m_Font; }

CFX_Font* GetFontFallback(int position);

```

**protected:**

```
CPDF_Font(CPDF_Document* pDocument, CPDF_Dictionary* pFontDict);
```

```
static int TT2PDF(int m, FXFT_FaceRec* face);
```

```
static bool FT_UseTTCharmap(FXFT_FaceRec* face,  
                             int platform_id,  
                             int encoding_id);
```

```
static const char* GetAdobeCharName(int iBaseEncoding,  
                                       const std::vector<ByteString>& charnames,  
                                       uint32_t charcode);
```

```
virtual bool Load() = 0;
```

```
void LoadUnicodeMap() const; // logically const only.
```

```
void LoadFontDescriptor(const CPDF_Dictionary* pFontDesc);
```

```
void CheckFontMetrics();
```

```
UnownedPtr<CPDF_Document> const m_pDocument;
```

```
CFX_Font m_Font;
```

```
std::vector<std::unique_ptr<CFX_Font>> m_FontFallbacks;
```

```
RetainPtr<CPDF_StreamAcc> m_pFontFile;
```

```
RetainPtr<CPDF_Dictionary> m_pFontDict;
```

```
ByteString m_BaseFontName;
```

```
mutable std::unique_ptr<CPDF_ToUnicodeMap> m_pToUnicodeMap;
```

```
mutable bool m_bToUnicodeLoaded = false;
```

```
int m_Flags = 0;
```

```
int m_StemV = 0;
```

```
int m_Ascent = 0;
```

```
int m_Descent = 0;
```

```
int m_ItalicAngle = 0;
```

```
FX_RECT m_FontBBox;
```

```
};
```

```
#endif // CORE_FPDFAPI_FONT_CPDF_FONT_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_FONT_CPDF_SIMPLEFONT_H_  
#define CORE_FPDFAPI_FONT_CPDF_SIMPLEFONT_H_
```

```
#include <vector>
```

```
#include "core/fpdfapi/font/cpdf_font.h"  
#include "core/fpdfapi/font/cpdf_fontencoding.h"  
#include "core/fxcrt/fx_string.h"  
#include "core/fxcrt/fx_system.h"
```

```
class CPDF_SimpleFont : public CPDF_Font {  
public:  
    ~CPDF_SimpleFont() override;  
  
    // CPDF_Font  
    uint32_t GetCharWidthF(uint32_t charcode) override;  
    FX_RECT GetCharBBox(uint32_t charcode) override;  
    int GlyphFromCharCode(uint32_t charcode, bool* pVertGlyph) override;  
    bool IsUnicodeCompatible() const override;  
    WideString UnicodeFromCharCode(uint32_t charcode) const override;  
    uint32_t CharCodeFromUnicode(wchar_t Unicode) const override;  
  
    const CPDF_FontEncoding* GetEncoding() const { return &m_Encoding; }  
  
    bool HasFontWidths() const override;  
  
protected:  
    CPDF_SimpleFont(CPDF_Document* pDocument, CPDF_Dictionary* pFontDict);  
  
    virtual void LoadGlyphMap() = 0;  
  
    bool LoadCommon();  
    void LoadSubstFont();  
    void LoadCharMetrics(int charcode);  
    void LoadPDFEncoding(bool bEmbedded, bool bTrueType);  
  
    CPDF_FontEncoding m_Encoding{PDFFONT_ENCODING_BUILTIN};  
    int m_BaseEncoding = PDFFONT_ENCODING_BUILTIN;  
    bool m_bUseFontWidth;  
    std::vector<ByteString> m_CharNames;  
    uint16_t m_GlyphIndex[256];  
    uint16_t m_CharWidth[256];  
    FX_RECT m_CharBBox[256];  
};  
  
#endif // CORE_FPDFAPI_FONT_CPDF_SIMPLEFONT_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CPDF_TOUNICODEMAP_H_
#define CORE_FPDFAPI_FONT_CPDF_TOUNICODEMAP_H_

#include <map>

#include "core/fxcrt/cfx_widetextbuf.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_CID2UnicodeMap;
class CPDF_SimpleParser;
class CPDF_Stream;

class CPDF_ToUnicodeMap {
public:
  explicit CPDF_ToUnicodeMap(const CPDF_Stream* pStream);
  ~CPDF_ToUnicodeMap();

  WideString Lookup(uint32_t charcode) const;
  uint32_t ReverseLookup(wchar_t unicode) const;

private:
  friend class cpdf_tounicodemap_StringToCode_Test;
  friend class cpdf_tounicodemap_StringToWideString_Test;

  static uint32_t StringToCode(ByteStringView str);
  static WideString StringToWideString(ByteStringView str);

  void Load(const CPDF_Stream* pStream);
  void HandleBeginBFChar(CPDF_SimpleParser* pParser);
  void HandleBeginBFRange(CPDF_SimpleParser* pParser);
  uint32_t GetUnicode() const;
  void SetCode(uint32_t srccode, WideString destcode);

  std::map<uint32_t, uint32_t> m_Map;
  UnownedPtr<const CPDF_CID2UnicodeMap> m_pBaseMap;
  CFX_WideTextBuf m_MultiCharBuf;
};

#endif // CORE_FPDFAPI_FONT_CPDF_TOUNICODEMAP_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CPDF_TRUETYPEFONT_H_
#define CORE_FPDFAPI_FONT_CPDF_TRUETYPEFONT_H_

#include "core/fpdfapi/font/cpdf_simplefont.h"
#include "core/fxcrt/fx_system.h"

class CPDF_TrueTypeFont final : public CPDF_SimpleFont {
public:
  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  ~CPDF_TrueTypeFont() override;

  // CPDF_Font:
  bool IsTrueTypeFont() const override;
  const CPDF_TrueTypeFont* AsTrueTypeFont() const override;
  CPDF_TrueTypeFont* AsTrueTypeFont() override;

private:
  CPDF_TrueTypeFont(CPDF_Document* pDocument, CPDF_Dictionary* pFontDict);

  // CPDF_Font:
  bool Load() override;

  // CPDF_SimpleFont:
  void LoadGlyphMap() override;
};

#endif // CORE_FPDFAPI_FONT_CPDF_TRUETYPEFONT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CPDF_TYPE1FONT_H_
#define CORE_FPDFAPI_FONT_CPDF_TYPE1FONT_H_

#include "build/build_config.h"
#include "core/fpdfapi/font/cpdf_simplefont.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxge/cfx_fontmapper.h"

class CPDF_Type1Font final : public CPDF_SimpleFont {
public:
  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  ~CPDF_Type1Font() override;

  // CPDF_Font:
  bool IsType1Font() const override;
  const CPDF_Type1Font* AsType1Font() const override;
  CPDF_Type1Font* AsType1Font() override;
#ifdef OS_MACOSX
  int GlyphFromCharCodeExt(uint32_t charcode) override;
#endif

  bool IsBase14Font() const { return m_Base14Font.has_value(); }

private:
  CPDF_Type1Font(CPDF_Document* pDocument, CPDF_Dictionary* pFontDict);

  // CPDF_Font:
  bool Load() override;

  // CPDF_SimpleFont:
  void LoadGlyphMap() override;

  bool IsSymbolicFont() const;
  bool IsFixedFont() const;

#ifdef OS_MACOSX
  void SetExtGID(const char* name, uint32_t charcode);
  void CalcExtGID(uint32_t charcode);

  uint16_t m_ExtGID[256];
#endif

  Optional<CFX_FontMapper::StandardFont> m_Base14Font;
};

#endif // CORE_FPDFAPI_FONT_CPDF_TYPE1FONT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_FONT_CPDF_TYPE3CHAR_H
#define CORE_FPDFAPI_FONT_CPDF_TYPE3CHAR_H

#include <memory>
#include <utility>

#include "core/fpdfapi/font/cpdf_font.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/optional.h"

class CFX_DIBitmap;

class CPDF_Type3Char {
public:
    CPDF_Type3Char();
    ~CPDF_Type3Char();

    static float TextUnitToGlyphUnit(float fTextUnit);
    static void TextUnitRectToGlyphUnitRect(CFX_FloatRect* pRect);

    bool LoadBitmapFromSoleImageOfForm();
    void InitializeFromStreamData(bool bColored, const float* pData);
    void Transform(CPDF_Font::FormIface* pForm, const CFX_Matrix& matrix);
    void WillBeDestroyed();

    RetainPtr<CFX_DIBitmap> GetBitmap();
    const RetainPtr<CFX_DIBitmap>& GetBitmap() const;

    bool colored() const { return m_bColored; }
    uint32_t width() const { return m_Width; }
    const CFX_Matrix& matrix() const { return m_ImageMatrix; }
    const FX_RECT& bbox() const { return m_BBox; }

    const CPDF_Font::FormIface* form() const { return m_pForm.get(); }
    void SetForm(std::unique_ptr<CPDF_Font::FormIface> pForm);

private:
    std::unique_ptr<CPDF_Font::FormIface> m_pForm;
    RetainPtr<CFX_DIBitmap> m_pBitmap;
    bool m_bColored = false;
    uint32_t m_Width = 0;
    CFX_Matrix m_ImageMatrix;
    FX_RECT m_BBox;
};

#endif // CORE_FPDFAPI_FONT_CPDF_TYPE3CHAR_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_FONT_CPDF_TYPE3FONT_H
```

```
#define CORE_FPDFAPI_FONT_CPDF_TYPE3FONT_H
```

```
#include <map>
```

```
#include <memory>
```

```
#include "core/fpdfapi/font/cpdf_simplefont.h"
```

```
#include "core/fxcrt/fx_coordinates.h"
```

```
#include "core/fxcrt/fx_system.h"
```

```
#include "core/fxcrt/unowned_ptr.h"
```

```
class CPDF_Dictionary;
```

```
class CPDF_Document;
```

```
class CPDF_Stream;
```

```
class CPDF_Type3Char;
```

```
class CPDF_Type3Font final : public CPDF_SimpleFont {
```

```
public:
```

```
    template <typename T, typename... Args>
```

```
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);
```

```
    ~CPDF_Type3Font() override;
```

```
    // CPDF_Font:
```

```
    bool IsType3Font() const override;
```

```
    const CPDF_Type3Font* AsType3Font() const override;
```

```
    CPDF_Type3Font* AsType3Font() override;
```

```
    void WillBeDestroyed() override;
```

```
    uint32_t GetCharWidthF(uint32_t charcode) override;
```

```
    FX_RECT GetCharBBox(uint32_t charcode) override;
```

```
    void SetPageResources(CPDF_Dictionary* pResources) {  
        m_pageResources.Reset(pResources);  
    }
```

```
    CPDF_Type3Char* LoadChar(uint32_t charcode);
```

```
    void CheckType3FontMetrics();
```

```
    CFX_Matrix& GetFontMatrix() { return m_FontMatrix; }
```

```
private:
```

```
    CPDF_Type3Font(CPDF_Document* pDocument,  
                  CPDF_Dictionary* pFontDict,  
                  FormFactoryIface* pFormFactory);
```

```
    // CPDF_Font:
```

```
    bool Load() override;
```

```
    // CPDF_SimpleFont:
```

```
    void LoadGlyphMap() override;
```

```
    // The depth char loading is in, to avoid recursive calling LoadChar().
```

```
    int m_CharLoadingDepth = 0;
```

```
    CFX_Matrix m_FontMatrix;
```

```
    UnownedPtr<FormFactoryIface> const m_pFormFactory;
```

```
    RetainPtr<CPDF_Dictionary> m_pCharProcs;
```

```
    RetainPtr<CPDF_Dictionary> m_pPageResources;
```

```
    RetainPtr<CPDF_Dictionary> m_pFontResources;
```

```
    std::map<uint32_t, std::unique_ptr<CPDF_Type3Char>> m_CacheMap;
    uint32_t m_CharWidthL[256];
};

#endif // CORE_FPDFAPI_FONT_CPDF_TYPE3FONT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_ALLSTATES_H_
#define CORE_FPDFAPI_PAGE_CPDF_ALLSTATES_H_

#include "core/fpdfapi/page/cpdf_graphicstates.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"

class CPDF_Array;
class CPDF_Dictionary;
class CPDF_StreamContentParser;

class CPDF_AllStates final : public CPDF_GraphicStates {
public:
  CPDF_AllStates();
  ~CPDF_AllStates() override;

  void Copy(const CPDF_AllStates& src);
  void ProcessExtGS(CPDF_Dictionary* pGS, CPDF_StreamContentParser* pParser);
  void SetLineDash(const CPDF_Array* pArray, float phase, float scale);

  CFX_Matrix m_TextMatrix;
  CFX_Matrix m_CTM;
  CFX_Matrix m_ParentMatrix;
  CFX_PointF m_TextPos;
  CFX_PointF m_TextLinePos;
  float m_TextLeading = 0.0f;
  float m_TextRise = 0.0f;
  float m_TextHorzScale = 1.0f;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_ALLSTATES_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PAGE_CPDF_ANNOTCONTEXT_H_  
#define CORE_FPDFAPI_PAGE_CPDF_ANNOTCONTEXT_H_
```

```
#include <memory>
```

```
#include "core/fxcrt/retain_ptr.h"  
#include "core/fxcrt/unowned_ptr.h"
```

```
class CPDF_Dictionary;  
class CPDF_Form;  
class CPDF_Page;  
class CPDF_Stream;
```

```
class CPDF_AnnotContext {  
  public:  
    CPDF_AnnotContext(CPDF_Dictionary* pAnnotDict, CPDF_Page* pPage);  
    ~CPDF_AnnotContext();  
  
    void SetForm(CPDF_Stream* pStream);  
    bool HasForm() const { return !!m_pAnnotForm; }  
    CPDF_Form* GetForm() const { return m_pAnnotForm.get(); }  
  
    // Never nullptr.  
    CPDF_Dictionary* GetAnnotDict() const { return m_pAnnotDict.get(); }  
  
    // Never nullptr.  
    CPDF_Page* GetPage() const { return m_pPage.get(); }  
  
  private:  
    std::unique_ptr<CPDF_Form> m_pAnnotForm;  
    RetainPtr<CPDF_Dictionary> const m_pAnnotDict;  
    UnownedPtr<CPDF_Page> const m_pPage;  
};  
  
#endif // CORE_FPDFAPI_PAGE_CPDF_ANNOTCONTEXT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PAGE_CPDF_CLIPPATH_H_
#define CORE_FPDFAPI_PAGE_CPDF_CLIPPATH_H_

#include <memory>
#include <utility>
#include <vector>

#include "core/fpdfapi/page/cpdf_path.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/shared_copy_on_write.h"

class CPDF_TextObject;

class CPDF_ClipPath {
public:
    CPDF_ClipPath();
    CPDF_ClipPath(const CPDF_ClipPath& that);
    CPDF_ClipPath& operator=(const CPDF_ClipPath& that);
    ~CPDF_ClipPath();

    void Emplace() { m_Ref.Emplace(); }
    void SetNull() { m_Ref.SetNull(); }

    bool HasRef() const { return !!m_Ref; }
    bool operator==(const CPDF_ClipPath& that) const {
        return m_Ref == that.m_Ref;
    }
    bool operator!=(const CPDF_ClipPath& that) const { return !(*this == that); }

    size_t GetPathCount() const;
    CPDF_Path GetPath(size_t i) const;
    uint8_t GetClipType(size_t i) const;
    size_t GetTextCount() const;
    CPDF_TextObject* GetText(size_t i) const;
    CFX_FloatRect GetClipBox() const;
    void AppendPath(CPDF_Path path, uint8_t type, bool bAutoMerge);
    void AppendTexts(std::vector<std::unique_ptr<CPDF_TextObject>>* pTexts);
    void CopyClipPath(const CPDF_ClipPath& that);
    void Transform(const CFX_Matrix& matrix);

private:
    class PathData final : public Retainable {
public:
        template <typename T, typename... Args>
        friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

        RetainPtr<PathData> Clone() const;

        using PathAndTypeData = std::pair<CPDF_Path, uint8_t>;

        std::vector<PathAndTypeData> m_PathAndTypeList;
        std::vector<std::unique_ptr<CPDF_TextObject>> m_TextList;

private:
        PathData();
        PathData(const PathData& that);
        ~PathData() override;
    };
};
```

```
};  
  
    SharedCopyOnWrite<PathData> m_Ref;  
};  
  
#endif // CORE_FPDFAPI_PAGE_CPDF_CLIPPATH_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_COLOR_H_
#define CORE_FPDFAPI_PAGE_CPDF_COLOR_H_

#include <memory>
#include <vector>

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_ColorSpace;
class CPDF_Pattern;
class PatternValue;

class CPDF_Color {
public:
  CPDF_Color();
  CPDF_Color(const CPDF_Color& that);

  ~CPDF_Color();

  CPDF_Color& operator=(const CPDF_Color& that);

  bool IsNull() const { return m_Buffer.empty() && !m_pValue; }
  bool IsPattern() const;
  void SetColorSpace(const RetainPtr<CPDF_ColorSpace>& pCS);
  void SetValueForNonPattern(const std::vector<float>& values);
  void SetValueForPattern(const RetainPtr<CPDF_Pattern>& pPattern,
                          const std::vector<float>& values);
  uint32_t CountComponents() const;
  bool IsColorSpaceRGB() const;
  bool GetRGB(int* R, int* G, int* B) const;

  // Should only be called if IsPattern() returns true.
  CPDF_Pattern* GetPattern() const;

protected:
  bool IsPatternInternal() const;

  std::vector<float> m_Buffer; // Used for non-pattern colorspace.
  std::unique_ptr<PatternValue> m_pValue; // Used for pattern colorspace.
  RetainPtr<CPDF_ColorSpace> m_pCS;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_COLOR_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_COLORSPACE_H_
#define CORE_FPDFAPI_PAGE_CPDF_COLORSPACE_H_

#include <array>
#include <memory>
#include <set>
#include <vector>

#include "core/fpdfapi/page/cpdf_pattern.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/span.h"

#define PDFCS_DEVICEGRAY 1
#define PDFCS_DEVICERGB 2
#define PDFCS_DEVICECMYK 3
#define PDFCS_CALGRAY 4
#define PDFCS_CALRGB 5
#define PDFCS_LAB 6
#define PDFCS_ICCBASED 7
#define PDFCS_SEPARATION 8
#define PDFCS_DEVICEN 9
#define PDFCS_INDEXED 10
#define PDFCS_PATTERN 11

class CPDF_Array;
class CPDF_Document;
class CPDF_Object;
class CPDF_PatternCS;

constexpr size_t kMaxPatternColorComps = 16;

class PatternValue {
public:
    PatternValue();
    PatternValue(const PatternValue& that);
    ~PatternValue();

    void SetComps(pdfium::span<const float> comps);
    pdfium::span<const float> GetComps() const {
        // TODO(tsepez): update span.h from base for implicit std::array ctor.
        return {m_Comps.data(), m_Comps.size()};
    }

    CPDF_Pattern* GetPattern() const { return m_pRetainedPattern.Get(); }
    void SetPattern(const RetainPtr<CPDF_Pattern>& pPattern) {
        m_pRetainedPattern = pPattern;
    }

private:
    RetainPtr<CPDF_Pattern> m_pRetainedPattern;
    std::array<float, kMaxPatternColorComps> m_Comps;
};
```

```

class CPDF_ColorSpace : public Retainable, public Observable {
public:
    static RetainPtr<CPDF_ColorSpace> GetStockCS(int Family);
    static RetainPtr<CPDF_ColorSpace> ColorspaceFromName(const ByteString& name);
    static RetainPtr<CPDF_ColorSpace> Load(CPDF_Document* pDoc,
                                           CPDF_Object* pObj);
    static RetainPtr<CPDF_ColorSpace> Load(
        CPDF_Document* pDoc,
        const CPDF_Object* pObj,
        std::set<const CPDF_Object*>* pVisited);
    static uint32_t ComponentsForFamily(int family);

    const CPDF_Array* GetArray() const { return m_pArray.Get(); }
    CPDF_Document* GetDocument() const { return m_pDocument.Get(); }

    // Should only be called if this colorspace is not a pattern.
    std::vector<float> CreateBufAndSetDefaultColor() const;

    uint32_t CountComponents() const;
    int GetFamily() const { return m_Family; }
    bool IsSpecial() const {
        return GetFamily() == PDFCS_SEPARATION || GetFamily() == PDFCS_DEVICEN ||
            GetFamily() == PDFCS_INDEXED || GetFamily() == PDFCS_PATTERN;
    }

    virtual bool GetRGB(const float* pBuf,
                      float* R,
                      float* G,
                      float* B) const = 0;

    virtual void GetDefaultValue(int iComponent,
                                float* value,
                                float* min,
                                float* max) const;
    virtual void TranslateImageLine(uint8_t* dest_buf,
                                    const uint8_t* src_buf,
                                    int pixels,
                                    int image_width,
                                    int image_height,
                                    bool bTransMask) const;
    virtual void EnableStdConversion(bool bEnabled);
    virtual bool IsNormal() const;

    // Returns |this| as a CPDF_PatternCS* if |this| is a pattern.
    virtual CPDF_PatternCS* AsPatternCS();
    virtual const CPDF_PatternCS* AsPatternCS() const;

    // Use instead of GetRGB() for patterns.
    virtual bool GetPatternRGB(const PatternValue& value,
                              float* R,
                              float* G,
                              float* B) const;

protected:
    CPDF_ColorSpace(CPDF_Document* pDoc, int family);
    ~CPDF_ColorSpace() override;

    // Returns the number of components, or 0 on failure.
    virtual uint32_t v_Load(CPDF_Document* pDoc,
                          const CPDF_Array* pArray,
                          std::set<const CPDF_Object*>* pVisited) = 0;

    // Stock colorspace are not loaded normally. This initializes their

```

```
// components count.
void SetComponentsForStockCS(uint32_t nComponents);

UnownedPtr<CPDF_Document> const m_pDocument;
RetainPtr<const CPDF_Array> m_pArray;
const int m_Family;
uint32_t m_dwStdConversion = 0;

private:
    uint32_t m_nComponents = 0;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_COLORSPACE_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_COLORSTATE_H_
#define CORE_FPDFAPI_PAGE_CPDF_COLORSTATE_H_

#include <vector>

#include "core/fpdfapi/page/cpdf_color.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/shared_copy_on_write.h"
#include "core/fxge/fx_dib.h"

class CPDF_Color;
class CPDF_ColorSpace;
class CPDF_Pattern;

class CPDF_ColorState {
public:
  CPDF_ColorState();
  CPDF_ColorState(const CPDF_ColorState& that);
  ~CPDF_ColorState();

  void Emplace();
  void SetDefault();

  FX_COLORREF GetFillColorRef() const;
  void SetFillColorRef(FX_COLORREF colorref);

  FX_COLORREF GetStrokeColorRef() const;
  void SetStrokeColorRef(FX_COLORREF colorref);

  const CPDF_Color* GetFillColor() const;
  CPDF_Color* GetMutableFillColor();
  bool HasFillColor() const;

  const CPDF_Color* GetStrokeColor() const;
  CPDF_Color* GetMutableStrokeColor();
  bool HasStrokeColor() const;

  void SetFillColor(const RetainPtr<CPDF_ColorSpace>& pCS,
                   const std::vector<float>& values);
  void SetStrokeColor(const RetainPtr<CPDF_ColorSpace>& pCS,
                     const std::vector<float>& values);
  void SetFillPattern(const RetainPtr<CPDF_Pattern>& pattern,
                    const std::vector<float>& values);
  void SetStrokePattern(const RetainPtr<CPDF_Pattern>& pattern,
                      const std::vector<float>& values);

  bool HasRef() const { return !!m_Ref; }

private:
  class ColorData final : public Retainable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    RetainPtr<ColorData> Clone() const;
  };
};
```

```
void SetDefault();

FX_COLORREF m_FillColorRef = 0;
FX_COLORREF m_StrokeColorRef = 0;
CPDF_Color m_FillColor;
CPDF_Color m_StrokeColor;

private:
ColorData();
ColorData(const ColorData& src);
~ColorData() override;
};

void SetColor(const RetainPtr<CPDF_ColorSpace>& pCS,
              const std::vector<float>& values,
              CPDF_Color* color,
              FX_COLORREF* colorref);
void SetPattern(const RetainPtr<CPDF_Pattern>& pPattern,
                const std::vector<float>& values,
                CPDF_Color* color,
                FX_COLORREF* colorref);

SharedCopyOnWrite<ColorData> m_Ref;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_COLORSTATE_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_CONTENTMARKITEM_H
#define CORE_FPDFAPI_PAGE_CPDF_CONTENTMARKITEM_H

#include <memory>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Dictionary;

class CPDF_ContentMarkItem final : public Retainable {
public:
    enum ParamType { kNone, kPropertiesDict, kDirectDict };

    explicit CPDF_ContentMarkItem(ByteString name);
    ~CPDF_ContentMarkItem() override;

    const ByteString& GetName() const { return m_MarkName; }
    ParamType GetParamType() const { return m_ParamType; }
    const CPDF_Dictionary* GetParam() const;
    CPDF_Dictionary* GetParam();
    const ByteString& GetPropertyName() const { return m_PropertyName; }
    bool HasMCID() const;

    void SetDirectDict(RetainPtr<CPDF_Dictionary> pDict);
    void SetPropertiesHolder(CPDF_Dictionary* pHolder,
                           const ByteString& property_name);

private:
    ParamType m_ParamType = kNone;
    ByteString m_MarkName;
    ByteString m_PropertyName;
    RetainPtr<CPDF_Dictionary> m_pPropertiesHolder;
    RetainPtr<CPDF_Dictionary> m_pDirectDict;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_CONTENTMARKITEM_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.  
  
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PAGE_CPDF_CONTENTMARKS_H_  
#define CORE_FPDFAPI_PAGE_CPDF_CONTENTMARKS_H_  
  
#include <memory>  
#include <vector>  
  
#include "core/fpdfapi/page/cpdf_contentmarkitem.h"  
#include "core/fxcrt/fx_system.h"  
#include "core/fxcrt/retain_ptr.h"  
  
class CPDF_Dictionary;  
  
class CPDF_ContentMarks {  
public:  
    CPDF_ContentMarks();  
    ~CPDF_ContentMarks();  
  
    std::unique_ptr<CPDF_ContentMarks> Clone();  
    int GetMarkedContentID() const;  
    size_t CountItems() const;  
    bool ContainsItem(const CPDF_ContentMarkItem* pItem) const;  
  
    // The returned pointer is never null.  
    CPDF_ContentMarkItem* GetItem(size_t index);  
    const CPDF_ContentMarkItem* GetItem(size_t index) const;  
  
    void AddMark(ByteString name);  
    void AddMarkWithDirectDict(ByteString name, CPDF_Dictionary* pDict);  
    void AddMarkWithPropertiesHolder(const ByteString& name,  
                                     CPDF_Dictionary* pDict,  
                                     const ByteString& property_name);  
    bool RemoveMark(CPDF_ContentMarkItem* pMarkItem);  
    void DeleteLastMark();  
    size_t FindFirstDifference(const CPDF_ContentMarks* other) const;  
  
private:  
    class MarkData final : public Retainable {  
    public:  
        MarkData();  
        MarkData(const MarkData& src);  
        ~MarkData() override;  
  
        size_t CountItems() const;  
        bool ContainsItem(const CPDF_ContentMarkItem* pItem) const;  
        CPDF_ContentMarkItem* GetItem(size_t index);  
        const CPDF_ContentMarkItem* GetItem(size_t index) const;  
  
        int GetMarkedContentID() const;  
        void AddMark(ByteString name);  
        void AddMarkWithDirectDict(ByteString name, CPDF_Dictionary* pDict);  
        void AddMarkWithPropertiesHolder(const ByteString& name,  
                                           CPDF_Dictionary* pDict,  
                                           const ByteString& property_name);  
        bool RemoveMark(CPDF_ContentMarkItem* pMarkItem);  
        void DeleteLastMark();  
  
private:
```

```
    std::vector<RetainPtr<CPDF_ContentMarkItem>> m_Marks;
};

void EnsureMarkDataExists();

RetainPtr<MarkData> m_pMarkData;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_CONTENTMARKS_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_CONTENTPARSER_H
#define CORE_FPDFAPI_PAGE_CPDF_CONTENTPARSER_H

#include <memory>
#include <set>
#include <vector>

#include "core/fpdfapi/page/cpdf_streamcontentparser.h"
#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/maybe_owned.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_AllStates;
class CPDF_Array;
class CPDF_Form;
class CPDF_Page;
class CPDF_PageObjectHolder;
class CPDF_Stream;
class CPDF_StreamAcc;
class CPDF_Type3Char;
class PauseIndicatorIface;

class CPDF_ContentParser {
public:
  explicit CPDF_ContentParser(CPDF_Page* pPage);
  CPDF_ContentParser(CPDF_Form* pForm,
                    const CPDF_AllStates* pGraphicStates,
                    const CFX_Matrix* pParentMatrix,
                    CPDF_Type3Char* pType3Char,
                    std::set<const uint8_t*>* pParsedSet);
  ~CPDF_ContentParser();

  const CPDF_AllStates* GetCurStates() const {
    return m_pParser ? m_pParser->GetCurStates() : nullptr;
  }
  // Returns whether to continue or not.
  bool Continue(PauseIndicatorIface* pPause);

private:
  enum class Stage : uint8_t {
    kGetContent = 1,
    kPrepareContent,
    kParse,
    kCheckClip,
    kComplete,
  };

  Stage GetContent();
  Stage PrepareContent();
  Stage Parse();
  Stage CheckClip();

  void HandlePageContentStream(CPDF_Stream* pStream);
  bool HandlePageContentArray(CPDF_Array* pArray);
  void HandlePageContentFailure();

  Stage m_CurrentStage;
};
```

```
UnownedPtr<CPDF_PageObjectHolder> const m_pObjectHolder;
UnownedPtr<CPDF_Type3Char> m_pType3Char; // Only used when parsing forms.
RetainPtr<CPDF_StreamAcc> m_pSingleStream;
std::vector<RetainPtr<CPDF_StreamAcc>> m_StreamArray;
std::vector<uint32_t> m_StreamSegmentOffsets;
MaybeOwned<uint8_t, FxFreeDeleter> m_pData;
uint32_t m_nStreams = 0;
uint32_t m_Size = 0;
uint32_t m_CurrentOffset = 0;

// Only used when parsing pages.
std::unique_ptr<std::set<const uint8_t*>> m_pParsedSet;

// |m_pParser| has a reference to |m_pParsedSet|, so must be below and thus
// destroyed first.
std::unique_ptr<CPDF_StreamContentParser> m_pParser;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_CONTENTPARSER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_DEVICECS_H_
#define CORE_FPDFAPI_PAGE_CPDF_DEVICECS_H_

#include <set>

#include "core/fpdfapi/page/cpdf_colorspace.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_DeviceCS final : public CPDF_ColorSpace {
public:
  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  ~CPDF_DeviceCS() override;

  // CPDF_ColorSpace:
  bool GetRGB(const float* pBuf, float* R, float* G, float* B) const override;
  void TranslateImageLine(uint8_t* pDestBuf,
                          const uint8_t* pSrcBuf,
                          int pixels,
                          int image_width,
                          int image_height,
                          bool bTransMask) const override;
  uint32_t v_Load(CPDF_Document* pDoc,
                 const CPDF_Array* pArray,
                 std::set<const CPDF_Object*>* pVisited) override;

private:
  explicit CPDF_DeviceCS(int family);
};

#endif // CORE_FPDFAPI_PAGE_CPDF_DEVICECS_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_DIBBASE_H
#define CORE_FPDFAPI_PAGE_CPDF_DIBBASE_H

#include <memory>
#include <vector>

#include "core/fpdfapi/page/cpdf_clippath.h"
#include "core/fpdfapi/page/cpdf_colorspace.h"
#include "core/fpdfapi/page/cpdf_graphicstates.h"
#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/dib/cfx_dibbase.h"
#include "third_party/base/span.h"

class CPDF_Color;
class CPDF_Dictionary;
class CPDF_Document;
class CPDF_Stream;
class CPDF_StreamAcc;

struct DIB_COMP_DATA {
    float m_DecodeMin;
    float m_DecodeStep;
    int m_ColorKeyMin;
    int m_ColorKeyMax;
};

namespace fxcodec {
class Jbig2Context;
class ScanlineDecoder;
} // namespace fxcodec

#define FPDF_HUGE_IMAGE_SIZE 60000000

class CPDF_DIBBase final : public CFX_DIBBase {
public:
    enum class LoadState : uint8_t { kFail, kSuccess, kContinue };

    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    bool Load(CPDF_Document* pDoc, const CPDF_Stream* pStream);

    // CFX_DIBBase
    bool SkipToScanline(int line, PauseIndicatorIface* pPause) const override;
    uint8_t* GetBuffer() const override;
    const uint8_t* GetScanline(int line) const override;
    void DownSampleScanline(int line,
                            uint8_t* dest_scan,
                            int dest_bpp,
                            int dest_width,
                            bool bFlipX,
                            int clip_left,
                            int clip_width) const override;

    RetainPtr<CPDF_ColorSpace> GetColorSpace() const { return m_pColorSpace; }
};
```

```
uint32_t GetMatteColor() const { return m_MatteColor; }

LoadState StartLoadDIBBase(CPDF_Document* pDoc,
                           const CPDF_Stream* pStream,
                           bool bHasMask,
                           const CPDF_Dictionary* pFormResources,
                           CPDF_Dictionary* pPageResources,
                           bool bStdCS,
                           uint32_t GroupFamily,
                           bool bLoadMask);

LoadState ContinueLoadDIBBase(PauseIndicatorIface* pPause);
RetainPtr<CPDF_DIBBase> DetachMask();

bool IsJBigImage() const;

private:
CPDF_DIBBase();
~CPDF_DIBBase() override;

LoadState StartLoadMask();
LoadState StartLoadMaskDIB();
bool ContinueToLoadMask();
LoadState ContinueLoadMaskDIB(PauseIndicatorIface* pPause);
bool LoadColorInfo(const CPDF_Dictionary* pFormResources,
                  const CPDF_Dictionary* pPageResources);
bool GetDecodeAndMaskArray(bool* bDefaultDecode, bool* bColorKey);
RetainPtr<CFX_DIBitmap> LoadJpxBitmap();
void LoadPalette();
LoadState CreateDecoder();
bool CreateDCTDecoder(pdfium::span<const uint8_t> src_span,
                    const CPDF_Dictionary* pParams);
void TranslateScanline24bpp(uint8_t* dest_scan,
                          const uint8_t* src_scan) const;
bool TranslateScanline24bppDefaultDecode(uint8_t* dest_scan,
                                       const uint8_t* src_scan) const;

void ValidateDictParam();
void DownSampleScanline1Bit(int orig_Bpp,
                          int dest_Bpp,
                          uint32_t src_width,
                          const uint8_t* pSrcLine,
                          uint8_t* dest_scan,
                          int dest_width,
                          bool bFlipX,
                          int clip_left,
                          int clip_width) const;

void DownSampleScanline8Bit(int orig_Bpp,
                          int dest_Bpp,
                          uint32_t src_width,
                          const uint8_t* pSrcLine,
                          uint8_t* dest_scan,
                          int dest_width,
                          bool bFlipX,
                          int clip_left,
                          int clip_width) const;

void DownSampleScanline32Bit(int orig_Bpp,
                          int dest_Bpp,
                          uint32_t src_width,
                          const uint8_t* pSrcLine,
                          uint8_t* dest_scan,
                          int dest_width,
                          bool bFlipX,
                          int clip_left,
                          int clip_width) const;
```

```
bool TransMask() const;
void SetMaskProperties();

UnownedPtr<CPDF_Document> m_pDocument;
RetainPtr<const CPDF_Stream> m_pStream;
RetainPtr<const CPDF_Dictionary> m_pDict;
RetainPtr<CPDF_StreamAcc> m_pStreamAcc;
RetainPtr<CPDF_ColorSpace> m_pColorSpace;
uint32_t m_Family = 0;
uint32_t m_bpc = 0;
uint32_t m_bpc_orig = 0;
uint32_t m_nComponents = 0;
uint32_t m_GroupFamily = 0;
uint32_t m_MatteColor = 0;
bool m_bLoadMask = false;
bool m_bDefaultDecode = true;
bool m_bImageMask = false;
bool m_bDoBpcCheck = true;
bool m_bColorKey = false;
bool m_bHasMask = false;
bool m_bStdCS = false;
std::vector<DIB_COMP_DATA> m_CompData;
std::unique_ptr<uint8_t, FxFreeDeleter> m_pLineBuf;
std::unique_ptr<uint8_t, FxFreeDeleter> m_pMaskedLine;
RetainPtr<CFX_DIBitmap> m_pCachedBitmap;
RetainPtr<CPDF_DIBBase> m_pMask;
RetainPtr<CPDF_StreamAcc> m_pGlobalAcc;
std::unique_ptr<fxcodec::ScanlineDecoder> m_pDecoder;

// Must come after |m_pCachedBitmap|.
std::unique_ptr<fxcodec::Jbig2Context> m_pJbig2Context;

RetainPtr<const CPDF_Stream> m_pMaskStream;
LoadState m_Status = LoadState::kFail;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_DIBBASE_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_DIBTRANSFERFUNC_H_
#define CORE_FPDFAPI_PAGE_CPDF_DIBTRANSFERFUNC_H_

#include <vector>

#include "core/fxge/dib/cfx_filtereddib.h"
#include "core/fxge/fx_dib.h"
#include "third_party/base/span.h"

class CPDF_TransferFunc;

class CPDF_DIBTransferFunc final : public CFX_FilteredDIB {
public:
  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  // CFX_FilteredDIB
  FXDIB_Format GetDestFormat() override;
  FX_ARGB* GetDestPalette() override;
  void TranslateScanline(const uint8_t* src_buf,
                        std::vector<uint8_t*>* dest_buf) const override;
  void TranslateDownSamples(uint8_t* dest_buf,
                           const uint8_t* src_buf,
                           int pixels,
                           int Bpp) const override;

private:
  explicit CPDF_DIBTransferFunc(
    const RetainPtr<CPDF_TransferFunc>& pTransferFunc);
  ~CPDF_DIBTransferFunc() override;

  RetainPtr<CPDF_TransferFunc> m_pTransferFunc;
  const pdfium::span<const uint8_t> m_RampR;
  const pdfium::span<const uint8_t> m_RampG;
  const pdfium::span<const uint8_t> m_RampB;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_DIBTRANSFERFUNC_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_DOCPAGEDATA_H
#define CORE_FPDFAPI_PAGE_CPDF_DOCPAGEDATA_H

#include <map>
#include <memory>
#include <set>

#include "core/fpdfapi/font/cpdf_font.h"
#include "core/fpdfapi/page/cpdf_colorspace.h"
#include "core/fpdfapi/parser/cpdf_document.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CFX_Font;
class CPDF_Dictionary;
class CPDF_FontEncoding;
class CPDF_IccProfile;
class CPDF_Image;
class CPDF_Object;
class CPDF_Pattern;
class CPDF_Stream;
class CPDF_StreamAcc;

class CPDF_DocPageData : public CPDF_Document::PageDataIface,
                        public CPDF_Font::FormFactoryIface {
public:
    static CPDF_DocPageData* FromDocument(const CPDF_Document* pDoc);

    CPDF_DocPageData();
    ~CPDF_DocPageData() override;

    // CPDF_Document::PageDataIface:
    void ClearStockFont() override;
    RetainPtr<CPDF_StreamAcc> GetFontFileStreamAcc(
        const CPDF_Stream* pFontStream) override;
    void MaybePurgeFontFileStreamAcc(const CPDF_Stream* pFontStream) override;

    // CPDF_Font::FormFactoryIface:
    std::unique_ptr<CPDF_Font::FormIface> CreateForm(
        CPDF_Document* pDocument,
        CPDF_Dictionary* pPageResources,
        CPDF_Stream* pFormStream) override;

    bool IsForceClear() const { return m_bForceClear; }

    RetainPtr<CPDF_Font> AddFont(std::unique_ptr<CFX_Font> pFont, int charset);
    RetainPtr<CPDF_Font> GetFont(CPDF_Dictionary* pFontDict);
    RetainPtr<CPDF_Font> AddStandardFont(const char* font,
                                        const CPDF_FontEncoding* pEncoding);
    RetainPtr<CPDF_Font> GetStandardFont(const ByteString& fontName,
                                        const CPDF_FontEncoding* pEncoding);
#endif

#ifdef OS_WIN
    RetainPtr<CPDF_Font> AddWindowsFont(LOGFONTA* pLogFont);
#endif
#endif
```

```

// Loads a colorspace.
RetainPtr<CPDF_ColorSpace> GetColorSpace(const CPDF_Object* pCSObj,
                                        const CPDF_Dictionary* pResources);

// Loads a colorspace in a context that might be while loading another
// colorspace. |pVisited| is passed recursively to avoid circular calls
// involving CPDF_ColorSpace::Load().
RetainPtr<CPDF_ColorSpace> GetColorSpaceGuarded(
    const CPDF_Object* pCSObj,
    const CPDF_Dictionary* pResources,
    std::set<const CPDF_Object*>* pVisited);

RetainPtr<CPDF_Pattern> GetPattern(CPDF_Object* pPatternObj,
                                  bool bShading,
                                  const CFX_Matrix& matrix);

RetainPtr<CPDF_Image> GetImage(uint32_t dwStreamObjNum);
void MaybePurgeImage(uint32_t dwStreamObjNum);

RetainPtr<CPDF_IccProfile> GetIccProfile(const CPDF_Stream* pProfileStream);

private:
// Loads a colorspace in a context that might be while loading another
// colorspace, or even in a recursive call from this method itself. |pVisited|
// is passed recursively to avoid circular calls involving
// CPDF_ColorSpace::Load() and |pVisitedInternal| is also passed recursively
// to avoid circular calls with this method calling itself.
RetainPtr<CPDF_ColorSpace> GetColorSpaceInternal(
    const CPDF_Object* pCSObj,
    const CPDF_Dictionary* pResources,
    std::set<const CPDF_Object*>* pVisited,
    std::set<const CPDF_Object*>* pVisitedInternal);

size_t CalculateEncodingDict(int charset, CPDF_Dictionary* pBaseDict);
CPDF_Dictionary* ProcessbCJK(
    CPDF_Dictionary* pBaseDict,
    int charset,
    ByteString basefont,
    std::function<void(wchar_t, wchar_t, CPDF_Array*)> Insert);
void Clear(bool bForceRelease);

bool m_bForceClear = false;

// Specific destruction order may be required between maps.
std::map<ByteString, RetainPtr<const CPDF_Stream>> m_HashProfileMap;
std::map<const CPDF_Object*, ObservedPtr<CPDF_ColorSpace>> m_ColorSpaceMap;
std::map<const CPDF_Stream*, RetainPtr<CPDF_StreamAcc>> m_FontFileMap;
std::map<const CPDF_Stream*, ObservedPtr<CPDF_IccProfile>> m_IccProfileMap;
std::map<const CPDF_Object*, ObservedPtr<CPDF_Pattern>> m_PatternMap;
std::map<uint32_t, RetainPtr<CPDF_Image>> m_ImageMap;
std::map<const CPDF_Dictionary*, ObservedPtr<CPDF_Font>> m_FontMap;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_DOCPAGEDATA_H_

```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_EXPINTFUNC_H_
#define CORE_FPDFAPI_PAGE_CPDF_EXPINTFUNC_H_

#include <set>
#include <vector>

#include "core/fpdfapi/page/cpdf_function.h"

class CPDF_ExpIntFunc final : public CPDF_Function {
public:
  CPDF_ExpIntFunc();
  ~CPDF_ExpIntFunc() override;

  // CPDF_Function
  bool v_Init(const CPDF_Object* pObj,
              std::set<const CPDF_Object*>* pVisited) override;
  bool v_Call(const float* inputs, float* results) const override;

  uint32_t m_nOrigOutputs = 0;
  float m_Exponent = 0.0f;
  std::vector<float> m_BeginValues;
  std::vector<float> m_EndValues;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_EXPINTFUNC_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_FORM_H_
#define CORE_FPDFAPI_PAGE_CPDF_FORM_H_

#include <memory>
#include <set>
#include <utility>

#include "core/fpdfapi/font/cpdf_font.h"
#include "core/fpdfapi/page/cpdf_pageobjectholder.h"

class CFX_Matrix;
class CPDF_AllStates;
class CPDF_Dictionary;
class CPDF_Document;
class CPDF_ImageObject;
class CPDF_Stream;
class CPDF_Type3Char;

class CPDF_Form final : public CPDF_PageObjectHolder,
                       public CPDF_Font::FormIface {
public:
    // Helper method to choose the first non-null resources dictionary.
    static CPDF_Dictionary* ChooseResourcesDict(CPDF_Dictionary* pResources,
                                                CPDF_Dictionary* pParentResources,
                                                CPDF_Dictionary* pPageResources);

    CPDF_Form(CPDF_Document* pDocument,
              CPDF_Dictionary* pPageResources,
              CPDF_Stream* pFormStream);
    CPDF_Form(CPDF_Document* pDocument,
              CPDF_Dictionary* pPageResources,
              CPDF_Stream* pFormStream,
              CPDF_Dictionary* pParentResources);
    ~CPDF_Form() override;

    // CPDF_Font::FormIface:
    void ParseContentForType3Char(CPDF_Type3Char* pType3Char) override;
    bool HasPageObjects() const override;
    CFX_FloatRect CalcBoundingBox() const override;
    Optional<std::pair<RetainPtr<CFX_DIBitmap>, CFX_Matrix>>
    GetBitmapAndMatrixFromSoleImageOfForm() const override;

    void ParseContent();
    void ParseContent(const CPDF_AllStates* pGraphicStates,
                     const CFX_Matrix* pParentMatrix,
                     std::set<const uint8_t*>* pParsedSet);

    const CPDF_Stream* GetStream() const;

private:
    void ParseContentInternal(const CPDF_AllStates* pGraphicStates,
                             const CFX_Matrix* pParentMatrix,
                             CPDF_Type3Char* pType3Char,
                             std::set<const uint8_t*>* pParsedSet);

    std::unique_ptr<std::set<const uint8_t*>> m_ParsedSet;
    RetainPtr<CPDF_Stream> const m_pFormStream;
};
```

};

#endif // CORE\_FPDFAPI\_PAGE\_CPDF\_FORM\_H\_

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_FORMOBJECT_H_
#define CORE_FPDFAPI_PAGE_CPDF_FORMOBJECT_H_

#include <memory>

#include "core/fpdfapi/page/cpdf_pageobject.h"
#include "core/fxcrt/fx_coordinates.h"

class CPDF_Form;

class CPDF_FormObject final : public CPDF_PageObject {
public:
  CPDF_FormObject(int32_t content_stream,
                  std::unique_ptr<CPDF_Form> pForm,
                  const CFX_Matrix& matrix);
  ~CPDF_FormObject() override;

  // CPDF_PageObject:
  Type GetType() const override;
  void Transform(const CFX_Matrix& matrix) override;
  bool IsForm() const override;
  CPDF_FormObject* AsForm() override;
  const CPDF_FormObject* AsForm() const override;

  void CalcBoundingBox();
  const CPDF_Form* form() const { return m_pForm.get(); }
  const CFX_Matrix& form_matrix() const { return m_FormMatrix; }

private:
  std::unique_ptr<CPDF_Form> const m_pForm;
  CFX_Matrix m_FormMatrix;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_FORMOBJECT_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PAGE_CPDF_FUNCTION_H
```

```
#define CORE_FPDFAPI_PAGE_CPDF_FUNCTION_H
```

```
#include <memory>
```

```
#include <set>
```

```
#include <vector>
```

```
class CPDF_ExpIntFunc;
```

```
class CPDF_Object;
```

```
class CPDF_SampledFunc;
```

```
class CPDF_StitchFunc;
```

```
class CPDF_Function {
```

```
  public:
```

```
    enum class Type {
```

```
      kTypeInvalid = -1,
```

```
      kType0Sampled = 0,
```

```
      kType2ExponentialInterpolation = 2,
```

```
      kType3Stitching = 3,
```

```
      kType4PostScript = 4,
```

```
    };
```

```
    static std::unique_ptr<CPDF_Function> Load(const CPDF_Object* pFuncObj);
```

```
    static Type IntegerToFunctionType(int iType);
```

```
    virtual ~CPDF_Function();
```

```
    bool Call(const float* inputs,
```

```
            uint32_t ninputs,
```

```
            float* results,
```

```
            int* nresults) const;
```

```
    uint32_t CountInputs() const { return m_nInputs; }
```

```
    uint32_t CountOutputs() const { return m_nOutputs; }
```

```
    float GetDomain(int i) const { return m_Domains[i]; }
```

```
    float GetRange(int i) const { return m_Ranges[i]; }
```

```
    float Interpolate(float x,
```

```
                    float xmin,
```

```
                    float xmax,
```

```
                    float ymin,
```

```
                    float ymax) const;
```

```
    const CPDF_SampledFunc* ToSampledFunc() const;
```

```
    const CPDF_ExpIntFunc* ToExpIntFunc() const;
```

```
    const CPDF_StitchFunc* ToStitchFunc() const;
```

```
protected:
```

```
  explicit CPDF_Function(Type type);
```

```
    static std::unique_ptr<CPDF_Function> Load(  
      const CPDF_Object* pFuncObj,
```

```
      std::set<const CPDF_Object*>* pVisited);
```

```
    bool Init(const CPDF_Object* pObj, std::set<const CPDF_Object*>* pVisited);
```

```
    virtual bool v_Init(const CPDF_Object* pObj,  
                      std::set<const CPDF_Object*>* pVisited) = 0;
```

```
    virtual bool v_Call(const float* inputs, float* results) const = 0;
```

```
    const Type m_Type;
```

```
uint32_t m_nInputs;  
uint32_t m_nOutputs;  
std::vector<float> m_Domains;  
std::vector<float> m_Ranges;  
};
```

```
#endif // CORE_FPDFAPI_PAGE_CPDF_FUNCTION_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PAGE_CPDF_GENERALSTATE_H
```

```
#define CORE_FPDFAPI_PAGE_CPDF_GENERALSTATE_H
```

```
#include "constants/transparency.h"
```

```
#include "core/fxcrt/fx_coordinates.h"
```

```
#include "core/fxcrt/fx_string.h"
```

```
#include "core/fxcrt/retain_ptr.h"
```

```
#include "core/fxcrt/shared_copy_on_write.h"
```

```
#include "core/fxge/fx_dib.h"
```

```
class CPDF_Object;
```

```
class CPDF_TransferFunc;
```

```
class CPDF_GeneralState {
```

```
  public:
```

```
    CPDF_GeneralState();
```

```
    CPDF_GeneralState(const CPDF_GeneralState& that);
```

```
    ~CPDF_GeneralState();
```

```
    void Emplace() { m_Ref.Emplace(); }
```

```
    bool HasRef() const { return !!m_Ref; }
```

```
    void SetRenderIntent(const ByteString& ri);
```

```
    ByteString GetBlendMode() const;
```

```
    BlendMode GetBlendType() const;
```

```
    void SetBlendType(BlendMode type);
```

```
    float GetFillAlpha() const;
```

```
    void SetFillAlpha(float alpha);
```

```
    float GetStrokeAlpha() const;
```

```
    void SetStrokeAlpha(float alpha);
```

```
    CPDF_Object* GetSoftMask() const;
```

```
    void SetSoftMask(CPDF_Object* pObject);
```

```
    const CPDF_Object* GetTR() const;
```

```
    void SetTR(CPDF_Object* pObject);
```

```
    RetainPtr<CPDF_TransferFunc> GetTransferFunc() const;
```

```
    void SetTransferFunc(const RetainPtr<CPDF_TransferFunc>& pFunc);
```

```
    void SetBlendMode(const ByteString& mode);
```

```
    const CFX_Matrix* GetSMaskMatrix() const;
```

```
    void SetSMaskMatrix(const CFX_Matrix& matrix);
```

```
    bool GetFillOP() const;
```

```
    void SetFillOP(bool op);
```

```
    bool GetStrokeOP() const;
```

```
    void SetStrokeOP(bool op);
```

```
    int GetOPMode() const;
```

```
    void SetOPMode(int mode);
```

```
void SetBG(CPDF_Object* pObject);
void SetUCR(CPDF_Object* pObject);
void SetHT(CPDF_Object* pObject);

void SetFlatness(float flatness);
void SetSmoothness(float smoothness);

bool GetStrokeAdjust() const;
void SetStrokeAdjust(bool adjust);

void SetAlphaSource(bool source);
void SetTextKnockout(bool knockout);

void SetMatrix(const CFX_Matrix& matrix);
CFX_Matrix* GetMutableMatrix();
```

```
private:
```

```
class StateData final : public Retainable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);
```

```
RetainPtr<StateData> Clone() const;
```

```
ByteString m_BlendMode = pdfium::transparency::kNormal;
BlendMode m_BlendType = BlendMode::kNormal;
RetainPtr<CPDF_Object> m_pSoftMask;
CFX_Matrix m_SMaskMatrix;
float m_StrokeAlpha = 1.0f;
float m_FillAlpha = 1.0f;
RetainPtr<const CPDF_Object> m_pTR;
RetainPtr<CPDF_TransferFunc> m_pTransferFunc;
CFX_Matrix m_Matrix;
int m_RenderIntent = 0;
bool m_StrokeAdjust = false;
bool m_AlphaSource = false;
bool m_TextKnockout = false;
bool m_StrokeOP = false;
bool m_FillOP = false;
int m_OPMode = 0;
RetainPtr<const CPDF_Object> m_pBG;
RetainPtr<const CPDF_Object> m_pUCR;
RetainPtr<const CPDF_Object> m_pHT;
float m_Flatness = 1.0f;
float m_Smoothness = 0.0f;
```

```
private:
```

```
StateData();
StateData(const StateData& that);
~StateData() override;
```

```
};
```

```
SharedCopyOnWrite<StateData> m_Ref;
```

```
};
```

```
#endif // CORE_FPDFAPI_PAGE_CPDF_GENERALSTATE_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_GRAPHICSTATES_H_
#define CORE_FPDFAPI_PAGE_CPDF_GRAPHICSTATES_H_

#include "core/fpdfapi/page/cpdf_clippath.h"
#include "core/fpdfapi/page/cpdf_colorstate.h"
#include "core/fpdfapi/page/cpdf_generalstate.h"
#include "core/fpdfapi/page/cpdf_textstate.h"
#include "core/fxge/cfx_graphstate.h"

class CPDF_GraphicStates {
public:
  CPDF_GraphicStates();
  virtual ~CPDF_GraphicStates();

  void CopyStates(const CPDF_GraphicStates& src);
  void DefaultStates();

  CPDF_ClipPath m_ClipPath;
  CFX_GraphState m_GraphState;
  CPDF_ColorState m_ColorState;
  CPDF_TextState m_TextState;
  CPDF_GeneralState m_GeneralState;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_GRAPHICSTATES_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_ICCPROFILE_H_
#define CORE_FPDFAPI_PAGE_CPDF_ICCPROFILE_H_

#include <memory>

#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/span.h"

class CPDF_Stream;

namespace fxcodesc {
class CLcmsCmm;
} // namespace fxcodesc

class CPDF_IccProfile final : public Retainable, public Observable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    const CPDF_Stream* GetStream() const { return m_pStream.Get(); }
    bool IsValid() const { return IsSRGB() || IsSupported(); }
    bool IsSRGB() const { return m_bsRGB; }
    bool IsSupported() const { return !!m_Transform; }
    fxcodesc::CLcmsCmm* transform() { return m_Transform.get(); }
    uint32_t GetComponentCount() const { return m_nSrcComponents; }

private:
    CPDF_IccProfile(const CPDF_Stream* pStream, pdfium::span<const uint8_t> span);
    ~CPDF_IccProfile() override;

    const bool m_bsRGB;
    uint32_t m_nSrcComponents = 0;
    RetainPtr<const CPDF_Stream> const m_pStream;
    std::unique_ptr<fxcodesc::CLcmsCmm> m_Transform;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_ICCPROFILE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_IMAGE_H
#define CORE_FPDFAPI_PAGE_CPDF_IMAGE_H

#include <memory>

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/span.h"

class CFX_DIBBase;
class CFX_DIBitmap;
class CPDF_Dictionary;
class CPDF_Document;
class CPDF_Page;
class CPDF_Stream;
class PauseIndicatorIface;
class IFX_SeekableReadStream;

class CPDF_Image final : public Retainable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    static bool IsValidJpegComponent(int32_t comps);
    static bool IsValidJpegBitsPerComponent(int32_t bpc);

    void ConvertStreamToIndirectObject();

    CPDF_Dictionary* GetDict() const;
    CPDF_Stream* GetStream() const { return m_pStream.Get(); }
    const CPDF_Dictionary* GetOC() const { return m_pOC.Get(); }
    CPDF_Document* GetDocument() const { return m_pDocument.Get(); }

    int32_t GetPixelHeight() const { return m_Height; }
    int32_t GetPixelWidth() const { return m_Width; }

    bool IsInline() const { return m_bIsInline; }
    bool IsMask() const { return m_bIsMask; }
    bool IsInterpol() const { return m_bInterpolate; }

    RetainPtr<CFX_DIBBase> LoadDIBBase() const;

    void SetImage(const RetainPtr<CFX_DIBitmap>& pBitmap);
    void SetJpegImage(const RetainPtr<IFX_SeekableReadStream>& pFile);
    void SetJpegImageInline(const RetainPtr<IFX_SeekableReadStream>& pFile);

    void ResetCache(CPDF_Page* pPage);

    // Returns whether to Continue() or not.
    bool StartLoadDIBBase(const CPDF_Dictionary* pFormResource,
                        CPDF_Dictionary* pPageResource,
                        bool bStdCS,
                        uint32_t GroupFamily,
                        bool bLoadMask);

    // Returns whether to Continue() or not.

```

```
bool Continue(PauseIndicatorIface* pPause);

RetainPtr<CFX_DIBBase> DetachBitmap();
RetainPtr<CFX_DIBBase> DetachMask();

RetainPtr<CFX_DIBBase> m_pDIBBase;
RetainPtr<CFX_DIBBase> m_pMask;
uint32_t m_MatteColor = 0;

private:
explicit CPDF_Image(CPDF_Document* pDoc);
CPDF_Image(CPDF_Document* pDoc, RetainPtr<CPDF_Stream> pStream);
CPDF_Image(CPDF_Document* pDoc, uint32_t dwStreamObjNum);
~CPDF_Image() override;

void FinishInitialization(CPDF_Dictionary* pStreamDict);
RetainPtr<CPDF_Dictionary> InitJPEG(pdfium::span<uint8_t> src_span);

int32_t m_Height = 0;
int32_t m_Width = 0;
bool m_bIsInline = false;
bool m_bIsMask = false;
bool m_bInterpolate = false;
UnownedPtr<CPDF_Document> const m_pDocument;
RetainPtr<CPDF_Stream> m_pStream;
RetainPtr<const CPDF_Dictionary> m_pOC;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_IMAGE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_IMAGEOBJECT_H_
#define CORE_FPDFAPI_PAGE_CPDF_IMAGEOBJECT_H_

#include "core/fpdfapi/page/cpdf_pageobject.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"

class CFX_DIBitmap;
class CPDF_Image;

class CPDF_ImageObject final : public CPDF_PageObject {
public:
  explicit CPDF_ImageObject(int32_t content_stream);
  CPDF_ImageObject();
  ~CPDF_ImageObject() override;

  // CPDF_PageObject
  Type GetType() const override;
  void Transform(const CFX_Matrix& matrix) override;
  bool IsImage() const override;
  CPDF_ImageObject* AsImage() override;
  const CPDF_ImageObject* AsImage() const override;

  void CalcBoundingBox();
  void SetImage(const RetainPtr<CPDF_Image>& pImage);
  RetainPtr<CPDF_Image> GetImage() const;
  RetainPtr<CFX_DIBitmap> GetIndependentBitmap() const;

  void set_matrix(const CFX_Matrix& matrix) { m_Matrix = matrix; }
  const CFX_Matrix& matrix() const { return m_Matrix; }

private:
  void MaybePurgeCache();

  CFX_Matrix m_Matrix;
  RetainPtr<CPDF_Image> m_pImage;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_IMAGEOBJECT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_MESHSTREAM_H_
#define CORE_FPDFAPI_PAGE_CPDF_MESHSTREAM_H_

#include <memory>
#include <tuple>
#include <vector>

#include "core/fpdfapi/page/cpdf_shadingpattern.h"
#include "core/fxcrt/cfx_bitstream.h"
#include "core/fxcrt/cfx_system.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_StreamAcc;

class CPDF_MeshVertex {
public:
    CPDF_MeshVertex();
    CPDF_MeshVertex(const CPDF_MeshVertex&);
    ~CPDF_MeshVertex();

    CFX_PointF position;
    float r;
    float g;
    float b;
};

class CFX_Matrix;
class CPDF_ColorSpace;
class CPDF_Function;
class CPDF_Stream;

class CPDF_MeshStream {
public:
    CPDF_MeshStream(ShadingType type,
                   const std::vector<std::unique_ptr<CPDF_Function>>& funcs,
                   const CPDF_Stream* pShadingStream,
                   const RetainPtr<CPDF_ColorSpace>& pCS);
    ~CPDF_MeshStream();

    bool Load();

    bool CanReadFlag() const;
    bool CanReadCoords() const;
    bool CanReadColor() const;

    uint32_t ReadFlag();
    CFX_PointF ReadCoords();
    std::tuple<float, float, float> ReadColor();

    bool ReadVertex(const CFX_Matrix& pObject2Bitmap,
                   CPDF_MeshVertex* vertex,
                   uint32_t* flag);
    std::vector<CPDF_MeshVertex> ReadVertexRow(const CFX_Matrix& pObject2Bitmap,
                                               int count);

    CFX_BitStream* BitStream() { return m_BitStream.get(); }
    uint32_t ComponentBits() const { return m_nComponentBits; }
};
```

```
uint32_t Components() const { return m_nComponents; }

private:
    static const uint32_t kMaxComponents = 8;

    const ShadingType m_type;
    const std::vector<std::unique_ptr<CPDF_Function>>& m_funcs;
    RetainPtr<const CPDF_Stream> const m_pShadingStream;
    RetainPtr<CPDF_ColorSpace> const m_pCS;
    uint32_t m_nCoordBits;
    uint32_t m_nComponentBits;
    uint32_t m_nFlagBits;
    uint32_t m_nComponents;
    uint32_t m_CoordMax;
    uint32_t m_ComponentMax;
    float m_xmin;
    float m_xmax;
    float m_ymin;
    float m_ymax;
    RetainPtr<CPDF_StreamAcc> m_pStream;
    std::unique_ptr<CFX_BitStream> m_BitStream;
    float m_ColorMin[kMaxComponents];
    float m_ColorMax[kMaxComponents];
};

#endif // CORE_FPDFAPI_PAGE_CPDF_MESHSTREAM_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_OCCONTEXT_H_
#define CORE_FPDFAPI_PAGE_CPDF_OCCONTEXT_H_

#include <map>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Array;
class CPDF_Dictionary;
class CPDF_Document;
class CPDF_PageObject;

class CPDF_OCContext final : public Retainable {
public:
    enum UsageType { View = 0, Design, Print, Export };

    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    bool CheckOCGVisible(const CPDF_Dictionary* pOCGDict) const;
    bool CheckObjectVisible(const CPDF_PageObject* pObj) const;

private:
    CPDF_OCContext(CPDF_Document* pDoc, UsageType eUsageType);
    ~CPDF_OCContext() override;

    bool LoadOCGStateFromConfig(const ByteString& csConfig,
                                const CPDF_Dictionary* pOCGDict) const;
    bool LoadOCGState(const CPDF_Dictionary* pOCGDict) const;
    bool GetOCGVisible(const CPDF_Dictionary* pOCGDict) const;
    bool GetOCGVE(const CPDF_Array* pExpression, int nLevel) const;
    bool LoadOCMDDict(const CPDF_Dictionary* pOCMDDict) const;

    UnownedPtr<CPDF_Document> const m_pDocument;
    const UsageType m_eUsageType;
    mutable std::map<const CPDF_Dictionary*, bool> m_OGCStateCache;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_OCCONTEXT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_PAGE_H_
#define CORE_FPDFAPI_PAGE_CPDF_PAGE_H_

#include <memory>
#include <utility>

#include "core/fpdfapi/page/cpdf_pageobjectholder.h"
#include "core/fpdfapi/page/ipdf_page.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/optional.h"

class CPDF_Dictionary;
class CPDF_Document;
class CPDF_Image;
class CPDF_Object;

class CPDF_Page final : public IPDF_Page, public CPDF_PageObjectHolder {
public:
    // Caller implements as desired, empty here due to layering.
    class View : public Observable {};

    // Data for the render layer to attach to this page.
    class RenderContextIface {
public:
        virtual ~RenderContextIface() {}
    };

    // Cache for the render layer to attach to this page.
    class RenderCacheIface {
public:
        virtual ~RenderCacheIface() {}
        virtual void ResetBitmapForImage(const RetainPtr<CPDF_Image>& pImage) = 0;
    };

    class RenderContextClearer {
public:
        explicit RenderContextClearer(CPDF_Page* pPage);
        ~RenderContextClearer();

private:
        UnownedPtr<CPDF_Page> const m_pPage;
    };

template <typename T, typename... Args>
friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    // IPDF_Page:
    CPDF_Page* AsPDFPage() override;
    CPDFXFA_Page* AsXFAPage() override;
    CPDF_Document* GetDocument() const override;
    float GetPageWidth() const override;
    float GetPageHeight() const override;
    CFX_Matrix GetDisplayMatrix(const FX_RECT& rect, int iRotate) const override;
};
```

```
Optional<CFX_PointF> DeviceToPage(
    const FX_RECT& rect,
    int rotate,
    const CFX_PointF& device_point) const override;
Optional<CFX_PointF> PageToDevice(
    const FX_RECT& rect,
    int rotate,
    const CFX_PointF& page_point) const override;

// CPDF_PageObjectHolder:
bool IsPage() const override;

void ParseContent();
const CFX_SizeF& GetPageSize() const { return m_PageSize; }
int GetPageRotation() const;
RenderCacheIface* GetRenderCache() const { return m_pRenderCache.get(); }
void SetRenderCache(std::unique_ptr<RenderCacheIface> pCache) {
    m_pRenderCache = std::move(pCache);
}

RenderContextIface* GetRenderContext() const {
    return m_pRenderContext.get();
}
void SetRenderContext(std::unique_ptr<RenderContextIface> pContext) {
    m_pRenderContext = std::move(pContext);
}

CPDF_Document* GetPDFDocument() const { return m_pPDFDocument.get(); }
View* GetView() const { return m_pView.get(); }
void SetView(View* pView) { m_pView.Reset(pView); }
void UpdateDimensions();

private:
CPDF_Page(CPDF_Document* pDocument, CPDF_Dictionary* pPageDict);
~CPDF_Page() override;

CPDF_Object* GetPageAttr(const ByteString& name) const;
CFX_FloatRect GetBox(const ByteString& name) const;

CFX_SizeF m_PageSize;
CFX_Matrix m_PageMatrix;
UnownedPtr<CPDF_Document> m_pPDFDocument;
std::unique_ptr<RenderCacheIface> m_pRenderCache;
std::unique_ptr<RenderContextIface> m_pRenderContext;
ObservedPtr<View> m_pView;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_PAGE_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_PAGEMODULE_H_
#define CORE_FPDFAPI_PAGE_CPDF_PAGEMODULE_H_

#include "core/fxcrt/retain_ptr.h"

class CPDF_Document;
class CPDF_ColorSpace;
class CPDF_DeviceCS;
class CPDF_PatternCS;

class CPDF_PageModule {
public:
  // Per-process singleton managed by callers.
  static void Create();
  static void Destroy();
  static CPDF_PageModule* GetInstance();

  RetainPtr<CPDF_ColorSpace> GetStockCS(int family);
  void ClearStockFont(CPDF_Document* pDoc);

private:
  CPDF_PageModule();
  ~CPDF_PageModule();

  RetainPtr<CPDF_DeviceCS> m_StockGrayCS;
  RetainPtr<CPDF_DeviceCS> m_StockRGBCS;
  RetainPtr<CPDF_DeviceCS> m_StockCMYKCS;
  RetainPtr<CPDF_PatternCS> m_StockPatternCS;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_PAGEMODULE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_PAGEOBJECT_H_
#define CORE_FPDFAPI_PAGE_CPDF_PAGEOBJECT_H_

#include "core/fpdfapi/page/cpdf_contentmarks.h"
#include "core/fpdfapi/page/cpdf_graphicstates.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"

class CPDF_FormObject;
class CPDF_ImageObject;
class CPDF_PathObject;
class CPDF_ShadingObject;
class CPDF_TextObject;

class CPDF_PageObject : public CPDF_GraphicStates {
public:
  enum Type {
    TEXT = 1,
    PATH,
    IMAGE,
    SHADING,
    FORM,
  };

  static constexpr int32_t kNoContentStream = -1;

  explicit CPDF_PageObject(int32_t content_stream);
  CPDF_PageObject(const CPDF_PageObject& src) = delete;
  CPDF_PageObject& operator=(const CPDF_PageObject& src) = delete;
  ~CPDF_PageObject() override;

  virtual Type GetType() const = 0;
  virtual void Transform(const CFX_Matrix& matrix) = 0;
  virtual bool IsText() const;
  virtual bool IsPath() const;
  virtual bool IsImage() const;
  virtual bool IsShading() const;
  virtual bool IsForm() const;
  virtual CPDF_TextObject* AsText();
  virtual const CPDF_TextObject* AsText() const;
  virtual CPDF_PathObject* AsPath();
  virtual const CPDF_PathObject* AsPath() const;
  virtual CPDF_ImageObject* AsImage();
  virtual const CPDF_ImageObject* AsImage() const;
  virtual CPDF_ShadingObject* AsShading();
  virtual const CPDF_ShadingObject* AsShading() const;
  virtual CPDF_FormObject* AsForm();
  virtual const CPDF_FormObject* AsForm() const;

  void SetDirty(bool value) { m_bDirty = value; }
  bool IsDirty() const { return m_bDirty; }
  void TransformClipPath(const CFX_Matrix& matrix);
  void TransformGeneralState(const CFX_Matrix& matrix);

  void SetRect(const CFX_FloatRect& rect) { m_Rect = rect; }
  const CFX_FloatRect& GetRect() const { return m_Rect; }
  FX_RECT GetBBox() const;
};
```

```
FX_RECT GetTransformedBBox(const CFX_Matrix& matrix) const;

// Get what content stream the object was parsed from in its page. This number
// is the index of the content stream in the "Contents" array, or 0 if there
// is a single content stream. If the object is newly created,
// |kNoContentStream| is returned.
//
// If the object is spread among more than one content stream, this is the
// index of the last stream.
int32_t GetContentStream() const { return m_ContentStream; }
void SetContentStream(int32_t new_content_stream) {
    m_ContentStream = new_content_stream;
}

CPDF_ContentMarks m_ContentMarks;

protected:
void CopyData(const CPDF_PageObject* pSrcObject);

CFX_FloatRect m_Rect;

private:
bool m_bDirty = false;
int32_t m_ContentStream;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_PAGEOBJECT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_PAGEOBJECTHOLDER_H_
#define CORE_FPDFAPI_PAGE_CPDF_PAGEOBJECTHOLDER_H_

#include <deque>
#include <map>
#include <memory>
#include <set>
#include <vector>

#include "core/fpdfapi/page/cpdf_transparency.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/fx_dib.h"

class CPDF_ContentParser;
class CPDF_Dictionary;
class CPDF_Document;
class CPDF_PageObject;
class CPDF_Stream;
class PauseIndicatorIface;

// These structs are used to keep track of resources that have already been
// generated in the page object holder.
struct GraphicsData {
    float fillAlpha;
    float strokeAlpha;
    BlendMode blendType;

    bool operator<(const GraphicsData& other) const;
};

struct FontData {
    ByteString baseFont;
    ByteString type;

    bool operator<(const FontData& other) const;
};

class CPDF_PageObjectHolder {
public:
    enum class ParseState : uint8_t { kNotParsed, kParsing, kParsed };

    using iterator = std::deque<std::unique_ptr<CPDF_PageObject>>::iterator;
    using const_iterator =
        std::deque<std::unique_ptr<CPDF_PageObject>>::const_iterator;

    CPDF_PageObjectHolder(CPDF_Document* pDoc,
                          CPDF_Dictionary* pDict,
                          CPDF_Dictionary* pPageResources,
                          CPDF_Dictionary* pResources);
    virtual ~CPDF_PageObjectHolder();

    virtual bool IsPage() const;

    void StartParse(std::unique_ptr<CPDF_ContentParser> pParser);
};
```

```

void ContinueParse(PauseIndicatorIface* pPause);
ParseState GetParseState() const { return m_ParseState; }

CPDF_Document* GetDocument() const { return m_pDocument.Get(); }

CPDF_Dictionary* GetDict() const { return m_pDict.Get(); }
size_t GetPageObjectCount() const { return m_PageObjectList.size(); }
CPDF_PageObject* GetPageObjectByIndex(size_t index) const;
void AppendPageObject(std::unique_ptr<CPDF_PageObject> pPageObj);
bool RemovePageObject(CPDF_PageObject* pPageObj);
bool ErasePageObjectAtIndex(size_t index);

iterator begin() { return m_PageObjectList.begin(); }
const_iterator begin() const { return m_PageObjectList.begin(); }

iterator end() { return m_PageObjectList.end(); }
const_iterator end() const { return m_PageObjectList.end(); }

const CFX_Matrix& GetLastCTM() const { return m_LastCTM; }
const CFX_FloatRect& GetBBox() const { return m_BBox; }

const CPDF_Transparency& GetTransparency() const { return m_Transparency; }
bool BackgroundAlphaNeeded() const { return m_bBackgroundAlphaNeeded; }
void SetBackgroundAlphaNeeded(bool needed) {
    m_bBackgroundAlphaNeeded = needed;
}

bool HasImageMask() const { return !m_MaskBoundingBoxes.empty(); }
const std::vector<CFX_FloatRect>& GetMaskBoundingBoxes() const {
    return m_MaskBoundingBoxes;
}
void AddImageMaskBoundingBox(const CFX_FloatRect& box);
void Transform(const CFX_Matrix& matrix);
bool HasDirtyStreams() const { return !m_DirtyStreams.empty(); }
std::set<int32_t> TakeDirtyStreams();

RetainPtr<CPDF_Dictionary> m_pPageResources;
RetainPtr<CPDF_Dictionary> m_pResources;
std::map<GraphicsData, ByteString> m_GraphicsMap;
std::map<FontData, ByteString> m_FontsMap;

protected:
void LoadTransInfo();

CFX_FloatRect m_BBox;
CPDF_Transparency m_Transparency;

private:
bool m_bBackgroundAlphaNeeded = false;
ParseState m_ParseState = ParseState::kNotParsed;
RetainPtr<CPDF_Dictionary> const m_pDict;
UnownedPtr<CPDF_Document> m_pDocument;
std::vector<CFX_FloatRect> m_MaskBoundingBoxes;
std::unique_ptr<CPDF_ContentParser> m_pParser;
std::deque<std::unique_ptr<CPDF_PageObject>> m_PageObjectList;
CFX_Matrix m_LastCTM;

// The indexes of Content streams that are dirty and need to be regenerated.
std::set<int32_t> m_DirtyStreams;
};
#endif // CORE_FPDFAPI_PAGE_CPDF_PAGEOBJECTHOLDER_H

```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_PATH_H_
#define CORE_FPDFAPI_PAGE_CPDF_PATH_H_

#include <vector>

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/shared_copy_on_write.h"
#include "core/fxge/cfx_pathdata.h"

class CPDF_Path {
public:
  CPDF_Path();
  CPDF_Path(const CPDF_Path& that);
  ~CPDF_Path();

  void Emplace() { m_Ref.Emplace(); }
  bool HasRef() const { return !!m_Ref; }

  const std::vector<FX_PATHPOINT>& GetPoints() const;
  void ClosePath();

  CFX_PointF GetPoint(int index) const;
  CFX_FloatRect GetBoundingBox() const;
  CFX_FloatRect GetBoundingBox(float line_width, float miter_limit) const;

  bool IsRect() const;
  void Transform(const CFX_Matrix& matrix);

  void Append(const CFX_PathData* pData, const CFX_Matrix* pMatrix);
  void AppendFloatRect(const CFX_FloatRect& rect);
  void AppendRect(float left, float bottom, float right, float top);
  void AppendPoint(const CFX_PointF& point, FXPT_TYPE type, bool close);

  // TODO(tsepez): Remove when all access thru this class.
  const CFX_PathData* GetObject() const { return m_Ref.GetObject(); }

private:
  SharedCopyOnWrite<CFX_RetainablePathData> m_Ref;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_PATH_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_PATHOBJECT_H_
#define CORE_FPDFAPI_PAGE_CPDF_PATHOBJECT_H_

#include "core/fpdfapi/page/cpdf_pageobject.h"
#include "core/fpdfapi/page/cpdf_path.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxge/render_defines.h"

class CPDF_PathObject final : public CPDF_PageObject {
public:
  explicit CPDF_PathObject(int32_t content_stream);
  CPDF_PathObject();
  ~CPDF_PathObject() override;

  // CPDF_PageObject
  Type GetType() const override;
  void Transform(const CFX_Matrix& matrix) override;
  bool IsPath() const override;
  CPDF_PathObject* AsPath() override;
  const CPDF_PathObject* AsPath() const override;

  void CalcBoundingBox();

  bool stroke() const { return m_bStroke; }
  void set_stroke(bool stroke) { m_bStroke = stroke; }

  // Layering, avoid caller knowledge of FXFILL_ values.
  bool has_no_filltype() const { return m_FillType == 0; }
  bool has_winding_filltype() const { return m_FillType == FXFILL_WINDING; }
  bool has_alternate_filltype() const { return m_FillType == FXFILL_ALTERNATE; }
  void set_no_filltype() { m_FillType = 0; }
  void set_winding_filltype() { m_FillType = FXFILL_WINDING; }
  void set_alternate_filltype() { m_FillType = FXFILL_ALTERNATE; }

  int filltype() const { return m_FillType; }
  void set_filltype(int filltype) { m_FillType = filltype; }

  CPDF_Path& path() { return m_Path; }
  const CPDF_Path& path() const { return m_Path; }

  const CFX_Matrix& matrix() const { return m_Matrix; }
  void set_matrix(const CFX_Matrix& matrix) { m_Matrix = matrix; }

private:
  bool m_bStroke = false;
  int m_FillType = 0;
  CPDF_Path m_Path;
  CFX_Matrix m_Matrix;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_PATHOBJECT_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_PATTERNCS_H_
#define CORE_FPDFAPI_PAGE_CPDF_PATTERNCS_H_

#include <set>

#include "core/fpdfapi/page/cpdf_colorspace.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Document;

class CPDF_PatternCS final : public CPDF_ColorSpace {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    ~CPDF_PatternCS() override;

    // Called for the stock pattern, since it is not initialized via
    // CPDF_ColorSpace::Load().
    void InitializeStockPattern();

    // CPDF_ColorSpace:
    bool GetRGB(const float* pBuf, float* R, float* G, float* B) const override;
    bool GetPatternRGB(const PatternValue& value,
                       float* R,
                       float* G,
                       float* B) const override;

    CPDF_PatternCS* AsPatternCS() override;
    const CPDF_PatternCS* AsPatternCS() const override;
    uint32_t v_Load(CPDF_Document* pDoc,
                   const CPDF_Array* pArray,
                   std::set<const CPDF_Object*>* pVisited) override;

private:
    explicit CPDF_PatternCS(CPDF_Document* pDoc);

    RetainPtr<CPDF_ColorSpace> m_pBaseCS;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_PATTERNCS_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_PATTERN_H
#define CORE_FPDFAPI_PAGE_CPDF_PATTERN_H

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Document;
class CPDF_Object;
class CPDF_ShadingPattern;
class CPDF_TilingPattern;

class CPDF_Pattern : public Retainable, public Observable {
public:
    // Values used in PDFs. Do not change.
    enum PatternType { kTiling = 1, kShading = 2 };

    ~CPDF_Pattern() override;

    virtual CPDF_TilingPattern* AsTilingPattern();
    virtual CPDF_ShadingPattern* AsShadingPattern();

    // All the getters that return pointers return non-NULL pointers.
    CPDF_Document* document() const { return m_pDocument.Get(); }
    CPDF_Object* pattern_obj() const { return m_pPatternObj.Get(); }
    const CFX_Matrix& pattern_to_form() const { return m_Pattern2Form; }
    const CFX_Matrix& parent_matrix() const { return m_ParentMatrix; }

protected:
    CPDF_Pattern(CPDF_Document* pDoc,
                CPDF_Object* pObj,
                const CFX_Matrix& parentMatrix);

    void SetPatternToFormMatrix();

private:
    UnownedPtr<CPDF_Document> const m_pDocument;
    RetainPtr<CPDF_Object> const m_pPatternObj;
    CFX_Matrix m_Pattern2Form;
    const CFX_Matrix m_ParentMatrix;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_PATTERN_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.  
  
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PAGE_CPDF_PSENGINE_H_  
#define CORE_FPDFAPI_PAGE_CPDF_PSENGINE_H_
```

```
#include <memory>  
#include <vector>
```

```
#include "core/fxcrt/fx_string.h"  
#include "core/fxcrt/fx_system.h"  
#include "third_party/base/span.h"
```

```
class CPDF_PSEngine;  
class CPDF_PSProc;  
class CPDF_SimpleParser;
```

```
enum PDF_PSOP : uint8_t {  
    PSOP_ADD,  
    PSOP_SUB,  
    PSOP_MUL,  
    PSOP_DIV,  
    PSOP_IDIV,  
    PSOP_MOD,  
    PSOP_NEG,  
    PSOP_ABS,  
    PSOP_CEILING,  
    PSOP_FLOOR,  
    PSOP_ROUND,  
    PSOP_TRUNCATE,  
    PSOP_SQRT,  
    PSOP_SIN,  
    PSOP_COS,  
    PSOP_ATAN,  
    PSOP_EXP,  
    PSOP_LN,  
    PSOP_LOG,  
    PSOP_CVI,  
    PSOP_CVR,  
    PSOP_EQ,  
    PSOP_NE,  
    PSOP_GT,  
    PSOP_GE,  
    PSOP_LT,  
    PSOP_LE,  
    PSOP_AND,  
    PSOP_OR,  
    PSOP_XOR,  
    PSOP_NOT,  
    PSOP_BITSHIFT,  
    PSOP_TRUE,  
    PSOP_FALSE,  
    PSOP_IF,  
    PSOP_IFELSE,  
    PSOP_POP,  
    PSOP_EXCH,  
    PSOP_DUP,  
    PSOP_COPY,  
    PSOP_INDEX,  
    PSOP_ROLL,  
}
```

```
    PSOP_PROC,
    PSOP_CONST
};

class CPDF_PSOP {
public:
    CPDF_PSOP();
    explicit CPDF_PSOP(PDF_PSOP op);
    explicit CPDF_PSOP(float value);
    ~CPDF_PSOP();

    float GetFloatValue() const;
    CPDF_PSProc* GetProc() const;
    PDF_PSOP GetOp() const { return m_op; }

private:
    const PDF_PSOP m_op;
    const float m_value;
    std::unique_ptr<CPDF_PSProc> m_proc;
};

class CPDF_PSProc {
public:
    CPDF_PSProc();
    ~CPDF_PSProc();

    bool Parse(CPDF_SimpleParser* parser, int depth);
    bool Execute(CPDF_PSEngine* pEngine);

    // These methods are exposed for testing.
    void AddOperatorForTesting(ByteStringView word);
    size_t num_operators() const { return m_Operators.size(); }
    const std::unique_ptr<CPDF_PSOP>& last_operator() {
        return m_Operators.back();
    }

private:
    static const int kMaxDepth = 128;

    void AddOperator(ByteStringView word);

    std::vector<std::unique_ptr<CPDF_PSOP>> m_Operators;
};

class CPDF_PSEngine {
public:
    CPDF_PSEngine();
    ~CPDF_PSEngine();

    bool Parse(pdfium::span<const uint8_t> input);
    bool Execute();
    bool DoOperator(PDF_PSOP op);
    void Reset() { m_StackCount = 0; }
    void Push(float value);
    float Pop();
    int PopInt();
    uint32_t GetStackSize() const { return m_StackCount; }

private:
    static constexpr uint32_t kPSEngineStackSize = 100;

    uint32_t m_StackCount = 0;
    CPDF_PSProc m_MainProc;
};
```

```
    float m_Stack[kPSEngineStackSize];  
};
```

```
#endif // CORE_FPDFAPI_PAGE_CPDF_PSENGINE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_PSFUNC_H_
#define CORE_FPDFAPI_PAGE_CPDF_PSFUNC_H_

#include <set>

#include "core/fpdfapi/page/cpdf_function.h"
#include "core/fpdfapi/page/cpdf_pseengine.h"

class CPDF_Object;

class CPDF_PSFunc final : public CPDF_Function {
public:
  CPDF_PSFunc();
  ~CPDF_PSFunc() override;

  // CPDF_Function
  bool v_Init(const CPDF_Object* pObj,
              std::set<const CPDF_Object*>* pVisited) override;
  bool v_Call(const float* inputs, float* results) const override;

private:
  mutable CPDF_PSEngine m_PS; // Pre-initialized scratch space for v_Call().
};

#endif // CORE_FPDFAPI_PAGE_CPDF_PSFUNC_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_SAMPLEDFUNC_H_
#define CORE_FPDFAPI_PAGE_CPDF_SAMPLEDFUNC_H_

#include <set>
#include <vector>

#include "core/fpdfapi/page/cpdf_function.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_StreamAcc;

class CPDF_SampledFunc final : public CPDF_Function {
public:
    struct SampleEncodeInfo {
        float encode_max;
        float encode_min;
        uint32_t sizes;
    };

    struct SampleDecodeInfo {
        float decode_max;
        float decode_min;
    };

    CPDF_SampledFunc();
    ~CPDF_SampledFunc() override;

    // CPDF_Function
    bool v_Init(const CPDF_Object* pObj,
               std::set<const CPDF_Object*>* pVisited) override;
    bool v_Call(const float* inputs, float* results) const override;

    const std::vector<SampleEncodeInfo>& GetEncodeInfo() const {
        return m_EncodeInfo;
    }
    uint32_t GetBitsPerSample() const { return m_nBitsPerSample; }

#ifdef _SKIA_SUPPORT_ || defined _SKIA_SUPPORT_PATHS_
    RetainPtr<CPDF_StreamAcc> GetSampleStream() const;
#endif

private:
    std::vector<SampleEncodeInfo> m_EncodeInfo;
    std::vector<SampleDecodeInfo> m_DecodeInfo;
    uint32_t m_nBitsPerSample;
    uint32_t m_SampleMax;
    RetainPtr<CPDF_StreamAcc> m_pSampleStream;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_SAMPLEDFUNC_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_SHADINGOBJECT_H_
#define CORE_FPDFAPI_PAGE_CPDF_SHADINGOBJECT_H_

#include "core/fpdfapi/page/cpdf_pageobject.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_ShadingPattern;

class CPDF_ShadingObject final : public CPDF_PageObject {
 public:
  CPDF_ShadingObject(int32_t content_stream,
                    CPDF_ShadingPattern* pattern,
                    const CFX_Matrix& matrix);
  ~CPDF_ShadingObject() override;

  // CPDF_PageObject:
  Type GetType() const override;
  void Transform(const CFX_Matrix& matrix) override;
  bool IsShading() const override;
  CPDF_ShadingObject* AsShading() override;
  const CPDF_ShadingObject* AsShading() const override;

  void CalcBoundingBox();

  const CPDF_ShadingPattern* pattern() const { return m_pShading.Get(); }
  const CFX_Matrix& matrix() const { return m_Matrix; }

 private:
  RetainPtr<CPDF_ShadingPattern> m_pShading;
  CFX_Matrix m_Matrix;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_SHADINGOBJECT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PAGE_CPDF_SHADINGPATTERN_H_
#define CORE_FPDFAPI_PAGE_CPDF_SHADINGPATTERN_H_
```

```
#include <memory>
#include <vector>
```

```
#include "core/fpdfapi/page/cpdf_colorspace.h"
#include "core/fpdfapi/page/cpdf_pattern.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
```

```
// Values used in PDFs except for |kInvalidShading| and |kMaxShading|.
// Do not change.
```

```
enum ShadingType {
  kInvalidShading = 0,
  kFunctionBasedShading = 1,
  kAxialShading = 2,
  kRadialShading = 3,
  kFreeFormGouraudTriangleMeshShading = 4,
  kLatticeFormGouraudTriangleMeshShading = 5,
  kCoonsPatchMeshShading = 6,
  kTensorProductPatchMeshShading = 7,
  kMaxShading = 8
};
```

```
class CFX_Matrix;
class CPDF_ColorSpace;
class CPDF_Document;
class CPDF_Function;
class CPDF_Object;
```

```
class CPDF_ShadingPattern final : public CPDF_Pattern {
public:
  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);
```

```
  ~CPDF_ShadingPattern() override;
```

```
  // CPDF_Pattern:
  CPDF_ShadingPattern* AsShadingPattern() override;
```

```
  bool IsMeshShading() const {
    return m_ShadingType == kFreeFormGouraudTriangleMeshShading ||
           m_ShadingType == kLatticeFormGouraudTriangleMeshShading ||
           m_ShadingType == kCoonsPatchMeshShading ||
           m_ShadingType == kTensorProductPatchMeshShading;
  }
```

```
  bool Load();
```

```
  ShadingType GetShadingType() const { return m_ShadingType; }
```

```
  bool IsShadingObject() const { return m_bShading; }
```

```
  const CPDF_Object* GetShadingObject() const;
```

```
  RetainPtr<CPDF_ColorSpace> GetCS() const { return m_pCS; }
```

```
  const std::vector<std::unique_ptr<CPDF_Function>>& GetFuncs() const {
    return m_pFunctions;
  }
```

**private:**

```
CPDF_ShadingPattern(CPDF_Document* pDoc,
                   CPDF_Object* pPatternObj,
                   bool bShading,
                   const CFX_Matrix& parentMatrix);
CPDF_ShadingPattern(const CPDF_ShadingPattern&) = delete;
CPDF_ShadingPattern& operator=(const CPDF_ShadingPattern&) = delete;

// Constraints in PDF 1.7 spec, 4.6.3 Shading Patterns, pages 308-331.
bool Validate() const;
bool ValidateFunctions(uint32_t nExpectedNumFunctions,
                      uint32_t nExpectedNumInputs,
                      uint32_t nExpectedNumOutputs) const;

ShadingType m_ShadingType = kInvalidShading;
const bool m_bShading;
RetainPtr<CPDF_ColorSpace> m_pCS;
std::vector<std::unique_ptr<CPDF_Function>> m_pFunctions;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_SHADINGPATTERN_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_STITCHFUNC_H_
#define CORE_FPDFAPI_PAGE_CPDF_STITCHFUNC_H_

#include <memory>
#include <set>
#include <vector>

#include "core/fpdfapi/page/cpdf_function.h"

class CPDF_StitchFunc final : public CPDF_Function {
public:
  CPDF_StitchFunc();
  ~CPDF_StitchFunc() override;

  // CPDF_Function
  bool v_Init(const CPDF_Object* pObj,
              std::set<const CPDF_Object*>* pVisited) override;
  bool v_Call(const float* inputs, float* results) const override;

  const std::vector<std::unique_ptr<CPDF_Function>>& GetSubFunctions() const {
    return m_pSubFunctions;
  }
  float GetBound(size_t i) const { return m_bounds[i]; }
  float GetEncode(size_t i) const { return m_encode[i]; }

private:
  std::vector<std::unique_ptr<CPDF_Function>> m_pSubFunctions;
  std::vector<float> m_bounds;
  std::vector<float> m_encode;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_STITCHFUNC_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_STREAMCONTENTPARSER_H
#define CORE_FPDFAPI_PAGE_CPDF_STREAMCONTENTPARSER_H

#include <map>
#include <memory>
#include <set>
#include <stack>
#include <vector>

#include "core/fpdfapi/page/cpdf_contentmarks.h"
#include "core/fxcrt/fx_number.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/cfx_pathdata.h"

class CPDF_AllStates;
class CPDF_ColorSpace;
class CPDF_Dictionary;
class CPDF_Document;
class CPDF_Font;
class CPDF_Image;
class CPDF_ImageObject;
class CPDF_Object;
class CPDF_PageObject;
class CPDF_PageObjectHolder;
class CPDF_Pattern;
class CPDF_Stream;
class CPDF_StreamParser;
class CPDF_TextObject;

class CPDF_StreamContentParser {
public:
    CPDF_StreamContentParser(CPDF_Document* pDoc,
                            CPDF_Dictionary* pPageResources,
                            CPDF_Dictionary* pParentResources,
                            const CFX_Matrix* pmtContentToUser,
                            CPDF_PageObjectHolder* pObjHolder,
                            CPDF_Dictionary* pResources,
                            const CFX_FloatRect& rcBBox,
                            const CPDF_AllStates* pStates,
                            std::set<const uint8_t*>* pParsedSet);
    ~CPDF_StreamContentParser();

    uint32_t Parse(const uint8_t* pData,
                  uint32_t dwSize,
                  uint32_t start_offset,
                  uint32_t max_cost,
                  const std::vector<uint32_t>& stream_start_offsets);
    CPDF_PageObjectHolder* GetPageObjectHolder() const {
        return m_pObjectHolder.Get();
    }
    CPDF_AllStates* GetCurStates() const { return m_pCurStates.get(); }
    bool IsColored() const { return m_bColored; }
    const float* GetType3Data() const { return m_Type3Data; }
    RetainPtr<CPDF_Font> FindFont(const ByteString& name);

    static ByteStringView FindKeyAbbreviationForTesting(ByteStringView abbr);
};
```

```
static ByteStringView FindValueAbbreviationForTesting(ByteStringView abbr);
```

```
private:
```

```
struct ContentParam {
    enum Type { OBJECT = 0, NUMBER, NAME };
```

```
ContentParam();
~ContentParam();
```

```
Type m_Type;
FX_Number m_Number;
ByteString m_Name;
RetainPtr<CPDF_Object> m_pObject;
```

```
};
```

```
static const int kParamBufSize = 16;
```

```
using OpCodes = std::map<uint32_t, void (CPDF_StreamContentParser::*) ()>;
static OpCodes InitializeOpCodes();
```

```
void AddNameParam(ByteStringView bsName);
void AddNumberParam(ByteStringView str);
void AddObjectParam(RetainPtr<CPDF_Object> pObj);
int GetNextParamPos();
void ClearAllParams();
CPDF_Object* GetObject(uint32_t index);
ByteString GetString(uint32_t index) const;
float GetNumber(uint32_t index) const;
// Calls GetNumber() |count| times and returns the values in reverse order.
// e.g. for |count| = 3, returns [GetNumber(2), GetNumber(1), GetNumber(0)].
std::vector<float> GetNumbers(size_t count) const;
int GetInteger(uint32_t index) const {
    return static_cast<int>(GetNumber(index));
}
void OnOperator(ByteStringView op);
void AddTextObject(const ByteString* pStrs,
                  float fInitKerning,
                  const std::vector<float>& kernings,
                  size_t nSegs);
float GetHorizontalTextSize(float fKerning) const;
float GetVerticalTextSize(float fKerning) const;
```

```
void OnChangeTextMatrix();
void ParsePathObject();
void AddPathPoint(float x, float y, FXPT_TYPE type, bool close);
void AddPathRect(float x, float y, float w, float h);
void AddPathObject(int FillType, bool bStroke);
CPDF_ImageObject* AddImage(RetainPtr<CPDF_Stream> pStream);
CPDF_ImageObject* AddImage(uint32_t streamObjNum);
CPDF_ImageObject* AddImage(const RetainPtr<CPDF_Image>& pImage);
```

```
void AddForm(CPDF_Stream* pStream);
void SetGraphicStates(CPDF_PageObject* pObj,
                    bool bColor,
                    bool bText,
                    bool bGraph);
```

```
RetainPtr<CPDF_ColorSpace> FindColorSpace(const ByteString& name);
RetainPtr<CPDF_Pattern> FindPattern(const ByteString& name, bool bShading);
CPDF_Dictionary* FindResourceHolder(const ByteString& type);
CPDF_Object* FindResourceObj(const ByteString& type, const ByteString& name);
```

```
// Takes ownership of |pImageObj|, returns unowned pointer to it.
CPDF_ImageObject* AddImageObject(std::unique_ptr<CPDF_ImageObject> pImageObj);
```

```
std::vector<float> GetColors() const;
std::vector<float> GetNamedColors() const;
int32_t GetCurrentStreamIndex();

void Handle_CloseFillStrokePath();
void Handle_FillStrokePath();
void Handle_CloseEOFillStrokePath();
void Handle_EOFillStrokePath();
void Handle_BeginMarkedContent_Dictionary();
void Handle_BeginImage();
void Handle_BeginMarkedContent();
void Handle_BeginText();
void Handle_CurveTo_123();
void Handle_ConcatMatrix();
void Handle_SetColorSpace_Fill();
void Handle_SetColorSpace_Stroke();
void Handle_SetDash();
void Handle_SetCharWidth();
void Handle_SetCachedDevice();
void Handle_ExecuteXObject();
void Handle_MarkPlace_Dictionary();
void Handle_EndImage();
void Handle_EndMarkedContent();
void Handle_EndText();
void Handle_FillPath();
void Handle_FillPathOld();
void Handle_EOFillPath();
void Handle_SetGray_Fill();
void Handle_SetGray_Stroke();
void Handle_SetExtendGraphState();
void Handle_ClosePath();
void Handle_SetFlat();
void Handle_BeginImageData();
void Handle_SetLineJoin();
void Handle_SetLineCap();
void Handle_SetCMYKColor_Fill();
void Handle_SetCMYKColor_Stroke();
void Handle_LineTo();
void Handle_MoveTo();
void Handle_SetMiterLimit();
void Handle_MarkPlace();
void Handle_EndPath();
void Handle_SaveGraphState();
void Handle_RestoreGraphState();
void Handle_Rectangle();
void Handle_SetRGBColor_Fill();
void Handle_SetRGBColor_Stroke();
void Handle_SetRenderIntent();
void Handle_CloseStrokePath();
void Handle_StrokePath();
void Handle_SetColor_Fill();
void Handle_SetColor_Stroke();
void Handle_SetColorPS_Fill();
void Handle_SetColorPS_Stroke();
void Handle_ShadeFill();
void Handle_SetCharSpace();
void Handle_MoveTextPoint();
void Handle_MoveTextPoint_SetLeading();
void Handle_SetFont();
void Handle_ShowText();
void Handle_ShowText_Positioning();
void Handle_SetTextLeading();
```

```

void Handle_SetTextMatrix();
void Handle_SetTextRenderMode();
void Handle_SetTextRise();
void Handle_SetWordSpace();
void Handle_SetHorzScale();
void Handle_MoveToNextLine();
void Handle_CurveTo_23();
void Handle_SetLineWidth();
void Handle_Clip();
void Handle_EOClip();
void Handle_CurveTo_13();
void Handle_NextLineShowText();
void Handle_NextLineShowText_Space();
void Handle_Invalid();

```

```

UnownedPtr<CPDF_Document> const m_pDocument;
RetainPtr<CPDF_Dictionary> const m_pPageResources;
RetainPtr<CPDF_Dictionary> const m_pParentResources;
RetainPtr<CPDF_Dictionary> const m_pResources;
UnownedPtr<CPDF_PageObjectHolder> const m_pObjectHolder;
UnownedPtr<std::set<const uint8_t*>> const m_ParsedSet;
CFX_Matrix m_mtContentToUser;
const CFX_FloatRect m_BBox;
uint32_t m_ParamStartPos = 0;
uint32_t m_ParamCount = 0;
UnownedPtr<CPDF_StreamParser> m_pSyntax;
std::unique_ptr<CPDF_AllStates> m_pCurStates;
std::stack<std::unique_ptr<CPDF_ContentMarks>> m_ContentMarksStack;
std::vector<std::unique_ptr<CPDF_TextObject>> m_ClipTextList;
UnownedPtr<CPDF_TextObject> m_pLastTextObject;
std::vector<FX_PATHPOINT> m_PathPoints;
float m_PathStartX = 0.0f;
float m_PathStartY = 0.0f;
float m_PathCurrentX = 0.0f;
float m_PathCurrentY = 0.0f;
uint8_t m_PathClipType = 0;
ByteString m_LastImageName;
RetainPtr<CPDF_Image> m_pLastImage;
bool m_bColored = false;
bool m_bResourceMissing = false;
std::vector<std::unique_ptr<CPDF_AllStates>> m_StateStack;
float m_Type3Data[6] = {0.0f};
ContentParam m_ParamBuf[kParamBufSize];

```

```

// The merged stream offsets at which a content stream ends and another
// begins.

```

```

std::vector<uint32_t> m_StreamStartOffsets;

```

```

// The merged stream offset at which the last |m_pSyntax| started parsing.

```

```

uint32_t m_StartParseOffset = 0;

```

```

};

```

```

#endif // CORE_FPDFAPI_PAGE_CPDF_STREAMCONTENTPARSER_H_

```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PAGE_CPDF_STREAMPARSER_H_  
#define CORE_FPDFAPI_PAGE_CPDF_STREAMPARSER_H_
```

```
#include <memory>  
#include <utility>
```

```
#include "core/fpdfapi/parser/cpdf_document.h"  
#include "core/fxcrt/string_pool_template.h"  
#include "core/fxcrt/weak_ptr.h"  
#include "third_party/base/span.h"
```

```
class CPDF_Dictionary;  
class CPDF_Object;  
class CPDF_Stream;
```

```
class CPDF_StreamParser {  
public:  
    enum SyntaxType { EndOfData, Number, Keyword, Name, Others };  
  
    explicit CPDF_StreamParser(pdfium::span<const uint8_t> span);  
    CPDF_StreamParser(pdfium::span<const uint8_t> span,  
                      const WeakPtr<ByteStringPool>& pPool);  
    ~CPDF_StreamParser();  
  
    SyntaxType ParseNextElement();  
    ByteStringView GetWord() const {  
        return ByteStringView(m_WordBuffer, m_WordSize);  
    }  
    uint32_t GetPos() const { return m_Pos; }  
    void SetPos(uint32_t pos) { m_Pos = pos; }  
    const RetainPtr<CPDF_Object>& GetObject() const { return m_pLastObj; }  
    RetainPtr<CPDF_Object> ReadNextObject(bool bAllowNestedArray,  
                                         bool bInArray,  
                                         uint32_t dwRecursionLevel);  
    RetainPtr<CPDF_Stream> ReadInlineStream(CPDF_Document* pDoc,  
                                           RetainPtr<CPDF_Dictionary> pDict,  
                                           const CPDF_Object* pCSObj);
```

```
private:
```

```
    friend class cpdf_streamparser_ReadHexString_Test;  
    static const uint32_t kMaxWordLength = 255;
```

```
    void GetNextWord(bool& bIsNumber);  
    ByteString ReadString();  
    ByteString ReadHexString();  
    bool PositionIsInBounds() const;  
    bool WordBufferMatches(const char* pWord) const;
```

```
    uint32_t m_Pos = 0;           // Current byte position within |m_pBuf|.   
    uint32_t m_WordSize = 0;     // Current byte position within |m_WordBuffer|.   
    WeakPtr<ByteStringPool> m_pPool;  
    RetainPtr<CPDF_Object> m_pLastObj;  
    pdfium::span<const uint8_t> m_pBuf;  
    uint8_t m_WordBuffer[kMaxWordLength + 1]; // Include space for NUL.
```

```
};
```

```
#endif // CORE_FPDFAPI_PAGE_CPDF_STREAMPARSER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_TEXTOBJECT_H_
#define CORE_FPDFAPI_PAGE_CPDF_TEXTOBJECT_H_

#include <memory>
#include <vector>

#include "core/fpdfapi/page/cpdf_pageobject.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_TextObjectItem {
public:
  CPDF_TextObjectItem();
  ~CPDF_TextObjectItem();

  uint32_t m_CharCode;
  CFX_PointF m-Origin;
};

class CPDF_TextObject final : public CPDF_PageObject {
public:
  explicit CPDF_TextObject(int32_t content_stream);
  CPDF_TextObject();
  ~CPDF_TextObject() override;

  // CPDF_PageObject
  Type GetType() const override;
  void Transform(const CFX_Matrix& matrix) override;
  bool IsText() const override;
  CPDF_TextObject* AsText() override;
  const CPDF_TextObject* AsText() const override;

  std::unique_ptr<CPDF_TextObject> Clone() const;

  size_t CountItems() const;
  void GetItemInfo(size_t index, CPDF_TextObjectItem* pInfo) const;

  size_t CountChars() const;
  void GetCharInfo(size_t index, uint32_t* charcode, float* kerning) const;
  void GetCharInfo(size_t index, CPDF_TextObjectItem* pInfo) const;
  float GetCharWidth(uint32_t charcode) const;
  int CountWords() const;
  WideString GetWordString(int nWordIndex) const;

  CFX_PointF GetPos() const { return m_Pos; }
  CFX_Matrix GetTextMatrix() const;

  RetainPtr<CPDF_Font> GetFont() const;
  float GetFontSize() const;

  TextRenderingMode GetTextRenderMode() const;

  void SetText(const ByteString& str);
  void SetPosition(CFX_PointF pos) { m_Pos = pos; }
  void SetPosition(float x, float y);
};
```

```
void RecalcPositionData();

const std::vector<uint32_t>& GetCharCodes() const { return m_CharCodes; }
const std::vector<float>& GetCharPositions() const { return m_CharPos; }

void SetSegments(const ByteString* pStrs,
                 const std::vector<float>& kernings,
                 size_t nSegs);
CFX_PointF CalcPositionData(float horz_scale);

private:
CFX_PointF m_Pos;
std::vector<uint32_t> m_CharCodes;
std::vector<float> m_CharPos;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_TEXTOBJECT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_TEXTSTATE_H_
#define CORE_FPDFAPI_PAGE_CPDF_TEXTSTATE_H_

#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/shared_copy_on_write.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Document;
class CPDF_Font;

// See PDF Reference 1.7, page 402, table 5.3.
enum class TextRenderingMode {
  MODE_UNKNOWN = -1,
  MODE_FILL = 0,
  MODE_STROKE = 1,
  MODE_FILL_STROKE = 2,
  MODE_INVISIBLE = 3,
  MODE_FILL_CLIP = 4,
  MODE_STROKE_CLIP = 5,
  MODE_FILL_STROKE_CLIP = 6,
  MODE_CLIP = 7,
  MODE_LAST = MODE_CLIP,
};

class CPDF_TextState {
public:
  CPDF_TextState();
  ~CPDF_TextState();

  void Emplace();

  RetainPtr<CPDF_Font> GetFont() const;
  void SetFont(const RetainPtr<CPDF_Font>& pFont);

  float GetFontSize() const;
  void SetFontSize(float size);

  const float* GetMatrix() const;
  float* GetMutableMatrix();

  float GetCharSpace() const;
  void SetCharSpace(float sp);

  float GetWordSpace() const;
  void SetWordSpace(float sp);

  float GetFontSizeV() const;
  float GetFontSizeH() const;
  float GetBaselineAngle() const;
  float GetShearAngle() const;

  TextRenderingMode GetTextMode() const;
  void SetTextMode(TextRenderingMode mode);

  const float* GetCTM() const;
  float* GetMutableCTM();
};
```

```
private:
class TextData final : public Retainable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    RetainPtr<TextData> Clone() const;

    void SetFont(const RetainPtr<CPDF_Font>& pFont);
    float GetFontSizeV() const;
    float GetFontSizeH() const;
    float GetBaselineAngle() const;
    float GetShearAngle() const;

    RetainPtr<CPDF_Font> m_pFont;
    UnownedPtr<CPDF_Document> m_pDocument;
    float m_FontSize;
    float m_CharSpace;
    float m_WordSpace;
    TextRenderingMode m_TextMode;
    float m_Matrix[4];
    float m_CTM[4];

private:
    TextData();
    TextData(const TextData& that);
    ~TextData() override;
};

SharedCopyOnWrite<TextData> m_Ref;
};

bool SetTextRenderingModeFromInt(int iMode, TextRenderingMode* mode);
bool TextRenderingModeIsClipMode(const TextRenderingMode& mode);
bool TextRenderingModeIsStrokeMode(const TextRenderingMode& mode);

#endif // CORE_FPDFAPI_PAGE_CPDF_TEXTSTATE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_TILINGPATTERN_H_
#define CORE_FPDFAPI_PAGE_CPDF_TILINGPATTERN_H_

#include <memory>

#include "core/fpdfapi/page/cpdf_pattern.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"

class CPDF_Document;
class CPDF_Form;
class CPDF_Object;
class CPDF_PageObject;

class CPDF_TilingPattern final : public CPDF_Pattern {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    ~CPDF_TilingPattern() override;

    // CPDF_Pattern:
    CPDF_TilingPattern* AsTilingPattern() override;

    std::unique_ptr<CPDF_Form> Load(CPDF_PageObject* pPageObj);

    bool colored() const { return m_bColored; }
    const CFX_FloatRect& bbox() const { return m_BBox; }
    float x_step() const { return m_XStep; }
    float y_step() const { return m_YStep; }

private:
    CPDF_TilingPattern(CPDF_Document* pDoc,
                      CPDF_Object* pPatternObj,
                      const CFX_Matrix& parentMatrix);
    CPDF_TilingPattern(const CPDF_TilingPattern&) = delete;
    CPDF_TilingPattern& operator=(const CPDF_TilingPattern&) = delete;

    bool m_bColored;
    CFX_FloatRect m_BBox;
    float m_XStep;
    float m_YStep;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_TILINGPATTERN_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_CPDF_TRANSFERFUNC_H_
#define CORE_FPDFAPI_PAGE_CPDF_TRANSFERFUNC_H_

#include <vector>

#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/fx_dib.h"
#include "third_party/base/span.h"

class CPDF_Document;
class CFX_DIBBase;

class CPDF_TransferFunc final : public Retainable, public Observable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    static constexpr size_t kChannelSampleSize = 256;

    FX_COLORREF TranslateColor(FX_COLORREF colorref) const;
    RetainPtr<CFX_DIBBase> TranslateImage(const RetainPtr<CFX_DIBBase>& pSrc);

    const CPDF_Document* GetDocument() const { return m_pPDFDoc.Get(); }

    // Spans are |kChannelSampleSize| in size.
    pdfium::span<const uint8_t> GetSamplesR() const;
    pdfium::span<const uint8_t> GetSamplesG() const;
    pdfium::span<const uint8_t> GetSamplesB() const;

    bool GetIdentity() const { return m_bIdentity; }

private:
    CPDF_TransferFunc(CPDF_Document* pDoc,
                     bool bIdentify,
                     std::vector<uint8_t> samples_r,
                     std::vector<uint8_t> samples_g,
                     std::vector<uint8_t> samples_b);
    ~CPDF_TransferFunc() override;

    UnownedPtr<CPDF_Document> const m_pPDFDoc;
    const bool m_bIdentity;
    const std::vector<uint8_t> m_SamplesR;
    const std::vector<uint8_t> m_SamplesG;
    const std::vector<uint8_t> m_SamplesB;
};

#endif // CORE_FPDFAPI_PAGE_CPDF_TRANSFERFUNC_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FPDFAPI_PAGE_CPDF_TRANSPARENCY_H_  
#define CORE_FPDFAPI_PAGE_CPDF_TRANSPARENCY_H_  
  
class CPDF_Transparency {  
public:  
    CPDF_Transparency();  
  
    CPDF_Transparency(const CPDF_Transparency& other);  
  
    bool IsGroup() const { return m_bGroup; }  
    bool IsIsolated() const { return m_bIsolated; }  
  
    void SetGroup() { m_bGroup = true; }  
    void SetIsolated() { m_bIsolated = true; }  
  
private:  
    bool m_bGroup = false;  
    bool m_bIsolated = false;  
};  
  
#endif // CORE_FPDFAPI_PAGE_CPDF_TRANSPARENCY_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PAGE_IPDF_PAGE_H_
#define CORE_FPDFAPI_PAGE_IPDF_PAGE_H_

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/optional.h"

class CPDF_Document;
class CPDF_Page;

// Small layering violation, incomplete type and always null if non-XFA.
class CPDFXFA_Page;

// Interface implemented by both page types (CPDF_Page and CPDFXFA_Page).
class IPDF_Page : public Retainable {
public:
  // There are actually 3 cases: a PDF page, an XFA page backed by a PDF page,
  // and an XFA page not backed by a PDF page. AsPDFPage() will return the
  // PDF page in either of the first two cases. AsXFAPage() is a straight
  // downcast and is null if not either of the last two cases. Hence, both
  // of these may return non-null on a given page.
  virtual CPDF_Page* AsPDFPage() = 0;
  virtual CPDFXFA_Page* AsXFAPage() = 0;

  virtual CPDF_Document* GetDocument() const = 0;

  virtual float GetPageWidth() const = 0;
  virtual float GetPageHeight() const = 0;
  virtual CFX_Matrix GetDisplayMatrix(const FX_RECT& rect,
                                      int iRotate) const = 0;

  virtual Optional<CFX_PointF> DeviceToPage(
    const FX_RECT& rect,
    int rotate,
    const CFX_PointF& device_point) const = 0;

  virtual Optional<CFX_PointF> PageToDevice(
    const FX_RECT& rect,
    int rotate,
    const CFX_PointF& page_point) const = 0;
};

inline CPDF_Page* ToPDFPage(IPDF_Page* pBase) {
  return pBase ? pBase->AsPDFPage() : nullptr;
}

inline CPDFXFA_Page* ToXFAPage(IPDF_Page* pBase) {
  return pBase ? pBase->AsXFAPage() : nullptr;
}

#endif // CORE_FPDFAPI_PAGE_IPDF_PAGE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_DOCUMENT_H_
#define CORE_FPDFAPI_PARSER_CPDF_DOCUMENT_H_

#include <memory>

#include "core/fpdfapi/parser/cpdf_indirect_object_holder.h"
#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/span.h"

class CPDF_Dictionary;
class IFX_SeekableReadStream;

class CFDF_Document final : public CPDF_IndirectObjectHolder {
public:
    static std::unique_ptr<CFDF_Document> CreateNewDoc();
    static std::unique_ptr<CFDF_Document> ParseMemory(
        pdfium::span<const uint8_t> span);

    CFDF_Document();
    ~CFDF_Document() override;

    ByteString WriteToString() const;
    CPDF_Dictionary* GetRoot() const { return m_pRootDict.Get(); }

private:
    void ParseStream(RetainPtr<IFX_SeekableReadStream> pFile);

    RetainPtr<CPDF_Dictionary> m_pRootDict;
    RetainPtr<IFX_SeekableReadStream> m_pFile;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_DOCUMENT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PARSER_CPDF_ARRAY_H  
#define CORE_FPDFAPI_PARSER_CPDF_ARRAY_H
```

```
#include <memory>  
#include <set>  
#include <type_traits>  
#include <utility>  
#include <vector>
```

```
#include "core/fpdfapi/parser/cpdf_indirect_object_holder.h"  
#include "core/fpdfapi/parser/cpdf_object.h"  
#include "core/fxcrt/fx_coordinates.h"  
#include "core/fxcrt/retain_ptr.h"  
#include "third_party/base/ptr_util.h"
```

```
class CPDF_Array final : public CPDF_Object {  
  public:  
    using const_iterator = std::vector<RetainPtr<CPDF_Object>>::const_iterator;
```

```
  template <typename T, typename... Args>  
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);
```

```
  // CPDF_Object:  
  Type GetType() const override;  
  RetainPtr<CPDF_Object> Clone() const override;  
  bool IsArray() const override;  
  CPDF_Array* AsArray() override;  
  const CPDF_Array* AsArray() const override;  
  bool WriteTo(IFX_ArchiveStream* archive,  
              const CPDF_Encryptor* encryptor) const override;
```

```
  bool IsEmpty() const { return m_Objects.empty(); }  
  size_t size() const { return m_Objects.size(); }  
  CPDF_Object* GetObjectAt(size_t index);  
  const CPDF_Object* GetObjectAt(size_t index) const;  
  CPDF_Object* GetDirectObjectAt(size_t index);  
  const CPDF_Object* GetDirectObjectAt(size_t index) const;  
  ByteString GetStringAt(size_t index) const;  
  WideString GetUnicodeTextAt(size_t index) const;  
  bool GetBooleanAt(size_t index, bool bDefault) const;  
  int GetIntegerAt(size_t index) const;  
  float GetNumberAt(size_t index) const;  
  CPDF_Dictionary* GetDictAt(size_t index);  
  const CPDF_Dictionary* GetDictAt(size_t index) const;  
  CPDF_Stream* GetStreamAt(size_t index);  
  const CPDF_Stream* GetStreamAt(size_t index) const;  
  CPDF_Array* GetArrayAt(size_t index);  
  const CPDF_Array* GetArrayAt(size_t index) const;  
  CFX_Matrix GetMatrix() const;  
  CFX_FloatRect GetRect() const;
```

```
  // Creates object owned by the array, returns unowned pointer to it.  
  // We have special cases for objects that can intern strings from  
  // a ByteStringPool. Prefer using these templates over direct calls  
  // to Add()/SetAt()/InsertAt() since by creating a new object with no  
  // previous references, they ensure cycles can not be introduced.  
  template <typename T, typename... Args>
```

```

typename std::enable_if<!CanInternStrings<T>::value, T*>::type AddNew(
    Args&&... args) {
    return static_cast<T*>(
        Add(pdfium::MakeRetain<T>(std::forward<Args>(args)...)));
}
template <typename T, typename... Args>
typename std::enable_if<CanInternStrings<T>::value, T*>::type AddNew(
    Args&&... args) {
    return static_cast<T*>(
        Add(pdfium::MakeRetain<T>(m_pPool, std::forward<Args>(args)...)));
}
template <typename T, typename... Args>
typename std::enable_if<!CanInternStrings<T>::value, T*>::type SetNewAt(
    size_t index,
    Args&&... args) {
    return static_cast<T*>(
        SetAt(index, pdfium::MakeRetain<T>(std::forward<Args>(args)...)));
}
template <typename T, typename... Args>
typename std::enable_if<CanInternStrings<T>::value, T*>::type SetNewAt(
    size_t index,
    Args&&... args) {
    return static_cast<T*>(SetAt(
        index, pdfium::MakeRetain<T>(m_pPool, std::forward<Args>(args)...)));
}
template <typename T, typename... Args>
typename std::enable_if<!CanInternStrings<T>::value, T*>::type InsertNewAt(
    size_t index,
    Args&&... args) {
    return static_cast<T*>(
        InsertAt(index, pdfium::MakeRetain<T>(std::forward<Args>(args)...)));
}
template <typename T, typename... Args>
typename std::enable_if<CanInternStrings<T>::value, T*>::type InsertNewAt(
    size_t index,
    Args&&... args) {
    return static_cast<T*>(InsertAt(
        index, pdfium::MakeRetain<T>(m_pPool, std::forward<Args>(args)...)));
}

// Takes ownership of |pObj|, returns unowned pointer to it.
CPDF_Object* Add(RetainPtr<CPDF_Object> pObj);
CPDF_Object* SetAt(size_t index, RetainPtr<CPDF_Object> pObj);
CPDF_Object* InsertAt(size_t index, RetainPtr<CPDF_Object> pObj);

void Clear();
void RemoveAt(size_t index);
void ConvertToIndirectObjectAt(size_t index,
                               CPDF_IndirectObjectHolder* pHolder);
bool IsLocked() const { return !!m_LockCount; }

private:
friend class CPDF_ArrayLocker;

CPDF_Array();
explicit CPDF_Array(const WeakPtr<ByteStringPool>& pPool);
~CPDF_Array() override;

RetainPtr<CPDF_Object> CloneNonCyclic(
    bool bDirect,
    std::set<const CPDF_Object*>* pVisited) const override;

std::vector<RetainPtr<CPDF_Object>> m_Objects;

```

```
WeakPtr<ByteStringPool> m_pPool;
mutable uint32_t m_LockCount = 0;
};

class CPDF_ArrayLocker {
public:
    using const_iterator = CPDF_Array::const_iterator;

    explicit CPDF_ArrayLocker(const CPDF_Array* pArray);
    ~CPDF_ArrayLocker();

    const_iterator begin() const {
        CHECK(m_pArray->IsLocked());
        return m_pArray->m_Objects.begin();
    }
    const_iterator end() const {
        CHECK(m_pArray->IsLocked());
        return m_pArray->m_Objects.end();
    }

private:
    RetainPtr<const CPDF_Array> const m_pArray;
};

inline CPDF_Array* ToArray(CPDF_Object* obj) {
    return obj ? obj->AsArray() : nullptr;
}

inline const CPDF_Array* ToArray(const CPDF_Object* obj) {
    return obj ? obj->AsArray() : nullptr;
}

inline RetainPtr<CPDF_Array> ToArray(RetainPtr<CPDF_Object> obj) {
    return RetainPtr<CPDF_Array>(ToArray(obj.Get()));
}

#endif // CORE_FPDFAPI_PARSER_CPDF_ARRAY_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_BOOLEAN_H_
#define CORE_FPDFAPI_PARSER_CPDF_BOOLEAN_H_

#include <memory>

#include "core/fpdfapi/parser/cpdf_object.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"

class CPDF_Boolean final : public CPDF_Object {
public:
  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  // CPDF_Object:
  Type GetType() const override;
  RetainPtr<CPDF_Object> Clone() const override;
  ByteString GetString() const override;
  int GetInteger() const override;
  void SetString(const ByteString& str) override;
  bool IsBoolean() const override;
  CPDF_Boolean* AsBoolean() override;
  const CPDF_Boolean* AsBoolean() const override;
  bool WriteTo(IFX_ArchiveStream* archive,
              const CPDF_Encryptor* encryptor) const override;

private:
  CPDF_Boolean();
  explicit CPDF_Boolean(bool value);
  ~CPDF_Boolean() override;

  bool m_bValue = false;
};

inline CPDF_Boolean* ToBoolean(CPDF_Object* obj) {
  return obj ? obj->AsBoolean() : nullptr;
}

inline const CPDF_Boolean* ToBoolean(const CPDF_Object* obj) {
  return obj ? obj->AsBoolean() : nullptr;
}

#endif // CORE_FPDFAPI_PARSER_CPDF_BOOLEAN_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FPDFAPI_PARSER_CPDF_CROSS_REF_AVAIL_H_  
#define CORE_FPDFAPI_PARSER_CPDF_CROSS_REF_AVAIL_H_
```

```
#include <queue>  
#include <set>
```

```
#include "core/fpdfapi/parser/cpdf_data_avail.h"  
#include "core/fxcrt/unowned_ptr.h"
```

```
class CPDF_SyntaxParser;
```

```
class CPDF_CrossRefAvail {  
public:  
    CPDF_CrossRefAvail(CPDF_SyntaxParser* parser,  
                       FX_FILESIZE last_crossref_offset);  
    ~CPDF_CrossRefAvail();  
  
    FX_FILESIZE last_crossref_offset() const { return last_crossref_offset_; }  
  
    CPDF_DataAvail::DocAvailStatus CheckAvail();
```

```
private:
```

```
enum class State {  
    kCrossRefCheck,  
    kCrossRefV4ItemCheck,  
    kCrossRefV4TrailerCheck,  
    kDone,  
};
```

```
bool CheckReadProblems();  
bool CheckCrossRef();  
bool CheckCrossRefV4();  
bool CheckCrossRefV4Item();  
bool CheckCrossRefV4Trailer();  
bool CheckCrossRefStream();
```

```
void AddCrossRefForCheck(FX_FILESIZE crossref_offset);
```

```
RetainPtr<CPDF_ReadValidator> GetValidator();
```

```
UnownedPtr<CPDF_SyntaxParser> parser_;  
const FX_FILESIZE last_crossref_offset_ = 0;  
CPDF_DataAvail::DocAvailStatus current_status_ =  
    CPDF_DataAvail::DataNotAvailable;  
State current_state_ = State::kCrossRefCheck;  
FX_FILESIZE current_offset_ = 0;  
std::queue<FX_FILESIZE> cross_refs_for_check_;  
std::set<FX_FILESIZE> registered_crossrefs_;
```

```
};  
  
#endif // CORE_FPDFAPI_PARSER_CPDF_CROSS_REF_AVAIL_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

#ifndef CORE_FPDFAPI_PARSER_CPDF_CROSS_REF_TABLE_H_
#define CORE_FPDFAPI_PARSER_CPDF_CROSS_REF_TABLE_H_

#include <map>
#include <memory>

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Dictionary;

class CPDF_CrossRefTable {
public:
  enum class ObjectType : uint8_t {
    kFree = 0x00,
    kNormal = 0x01,
    kNotCompressed = kNormal,
    kCompressed = 0x02,
    kObjStream = 0xFF,
    kNull = kObjStream,
  };

  struct ObjectInfo {
    ObjectInfo() : pos(0), type(ObjectType::kFree), gennum(0) {}
    // if type is ObjectType::kCompressed the archive_obj_num should be used.
    // if type is ObjectType::kNotCompressed the pos should be used.
    // In other cases its are unused.
    union {
      FX_FILESIZE pos;
      uint32_t archive_obj_num;
    };
    ObjectType type;
    uint16_t gennum;
  };

  // Merge cross reference tables. Apply top on current.
  static std::unique_ptr<CPDF_CrossRefTable> MergeUp(
    std::unique_ptr<CPDF_CrossRefTable> current,
    std::unique_ptr<CPDF_CrossRefTable> top);

  CPDF_CrossRefTable();
  explicit CPDF_CrossRefTable(RetainPtr<CPDF_Dictionary> trailer);
  ~CPDF_CrossRefTable();

  void AddCompressed(uint32_t obj_num, uint32_t archive_obj_num);
  void AddNormal(uint32_t obj_num, uint16_t gen_num, FX_FILESIZE pos);
  void SetFree(uint32_t obj_num);

  void SetTrailer(RetainPtr<CPDF_Dictionary> trailer);
  const CPDF_Dictionary* trailer() const { return trailer_.Get(); }
  CPDF_Dictionary* GetMutableTrailerForTesting() { return trailer_.Get(); }

  const ObjectInfo* GetObjectInfo(uint32_t obj_num) const;

  const std::map<uint32_t, ObjectInfo>& objects_info() const {
    return objects_info_;
  }

  void Update(std::unique_ptr<CPDF_CrossRefTable> new_cross_ref);

```

```
void ShrinkObjectMap(uint32_t objnum);

private:
void UpdateInfo(std::map<uint32_t, ObjectInfo>&& new_objects_info);
void UpdateTrailer(RetainPtr<CPDF_Dictionary> new_trailer);

RetainPtr<CPDF_Dictionary> trailer_;
std::map<uint32_t, ObjectInfo> objects_info_;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_CROSS_REF_TABLE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PARSER_CPDF_CRYPT_HANDLER_H
```

```
#define CORE_FPDFAPI_PARSER_CPDF_CRYPT_HANDLER_H
```

```
#include <memory>
```

```
#include "core/fdrm/fx_crypt.h"
```

```
#include "core/fxcrt/cfx_binarybuf.h"
```

```
#include "core/fxcrt/fx_memory_wrappers.h"
```

```
#include "core/fxcrt/fx_string.h"
```

```
#include "core/fxcrt/fx_system.h"
```

```
#include "third_party/base/span.h"
```

```
class CPDF_Dictionary;
```

```
class CPDF_Object;
```

```
class CPDF_SecurityHandler;
```

```
class CPDF_CryptoHandler {
```

```
public:
```

```
    CPDF_CryptoHandler(int cipher, const uint8_t* key, size_t keylen);
```

```
    ~CPDF_CryptoHandler();
```

```
    static bool IsSignatureDictionary(const CPDF_Dictionary* dictionary);
```

```
    bool DecryptObjectTree(RetainPtr<CPDF_Object> object);
```

```
    size_t EncryptGetSize(pdfium::span<const uint8_t> source) const;
```

```
    bool EncryptContent(uint32_t objnum,  
                       uint32_t gennum,  
                       pdfium::span<const uint8_t> source,  
                       uint8_t* dest_buf,  
                       uint32_t& dest_size);
```

```
    bool IsCipherAES() const;
```

```
private:
```

```
    uint32_t DecryptGetSize(uint32_t src_size);
```

```
    void* DecryptStart(uint32_t objnum, uint32_t gennum);
```

```
    ByteString Decrypt(uint32_t objnum, uint32_t gennum, const ByteString& str);
```

```
    bool DecryptStream(void* context,  
                     pdfium::span<const uint8_t> source,  
                     CFX_BinaryBuf& dest_buf);
```

```
    bool DecryptFinish(void* context, CFX_BinaryBuf& dest_buf);
```

```
    void PopulateKey(uint32_t objnum, uint32_t gennum, uint8_t* key);
```

```
    void CryptBlock(bool bEncrypt,  
                  uint32_t objnum,  
                  uint32_t gennum,  
                  pdfium::span<const uint8_t> source,  
                  uint8_t* dest_buf,  
                  uint32_t& dest_size);
```

```
    void* CryptStart(uint32_t objnum, uint32_t gennum, bool bEncrypt);
```

```
    bool CryptStream(void* context,  
                   pdfium::span<const uint8_t> source,  
                   CFX_BinaryBuf& dest_buf,  
                   bool bEncrypt);
```

```
    bool CryptFinish(void* context, CFX_BinaryBuf& dest_buf, bool bEncrypt);
```

```
    const size_t m_KeyLen;
```

```
    const int m_Cipher;
    std::unique_ptr<CRYPT_aes_context, FxFreeDeleter> m_pAESContext;
    uint8_t m_EncryptKey[32];
};

#endif // CORE_FPDFAPI_PARSER_CPDF_CRYPTO_HANDLER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_DATA_AVAIL_H_
#define CORE_FPDFAPI_PARSER_CPDF_DATA_AVAIL_H_

#include <map>
#include <memory>
#include <set>
#include <utility>
#include <vector>

#include "core/fpdfapi/parser/cpdf_document.h"
#include "core/fpdfapi/parser/cpdf_parser.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_CrossRefAvail;
class CPDF_Dictionary;
class CPDF_HintTables;
class CPDF_IndirectObjectHolder;
class CPDF_LinearizedHeader;
class CPDF_PageObjectAvail;
class CPDF_ReadValidator;
class CPDF_SyntaxParser;

enum PDF_DATAAVAIL_STATUS {
  PDF_DATAAVAIL_HEADER = 0,
  PDF_DATAAVAIL_FIRSTPAGE,
  PDF_DATAAVAIL_HINTTABLE,
  PDF_DATAAVAIL_LOADALLCROSSREF,
  PDF_DATAAVAIL_ROOT,
  PDF_DATAAVAIL_INFO,
  PDF_DATAAVAIL_PAGETREE,
  PDF_DATAAVAIL_PAGE,
  PDF_DATAAVAIL_PAGE_LATERLOAD,
  PDF_DATAAVAIL_RESOURCES,
  PDF_DATAAVAIL_DONE,
  PDF_DATAAVAIL_ERROR,
  PDF_DATAAVAIL_LOADALLFILE,
};

enum PDF_PAGENODE_TYPE {
  PDF_PAGENODE_UNKNOWN = 0,
  PDF_PAGENODE_PAGE,
  PDF_PAGENODE_PAGES,
  PDF_PAGENODE_ARRAY,
};

class CPDF_DataAvail final : public Observable::ObserverIface {
public:
  // Must match PDF_DATA_* definitions in public/fpdf_dataavail.h, but cannot
  // #include that header. fpdfsdk/fpdf_dataavail.cpp has static_asserts
  // to make sure the two sets of values match.
  enum DocAvailStatus {
    DataError = -1,          // PDF_DATA_ERROR
    DataNotAvailable = 0,   // PDF_DATA_NOTAVAIL
    DataAvailable = 1,      // PDF_DATA_AVAIL
  };

  // Must match PDF_*LINEAR* definitions in public/fpdf_dataavail.h, but cannot
```

```
// #include that header. fpdfsdk/fpdf_dataavail.cpp has static_asserts
// to make sure the two sets of values match.
enum DocLinearizationStatus {
    LinearizationUnknown = -1, // PDF_LINEARIZATION_UNKNOWN
    NotLinearized = 0, // PDF_NOT_LINEARIZED
    Linearized = 1, // PDF_LINEARIZED
};

// Must match PDF_FORM_* definitions in public/fpdf_dataavail.h, but cannot
// #include that header. fpdfsdk/fpdf_dataavail.cpp has static_asserts
// to make sure the two sets of values match.
enum DocFormStatus {
    FormError = -1, // PDF_FORM_ERROR
    FormNotAvailable = 0, // PDF_FORM_NOTAVAIL
    FormAvailable = 1, // PDF_FORM_AVAIL
    FormNotExist = 2, // PDF_FORM_NOTEXIST
};

class FileAvail {
public:
    virtual ~FileAvail();
    virtual bool IsDataAvail(FX_FILESIZE offset, size_t size) = 0;
};

class DownloadHints {
public:
    virtual ~DownloadHints();
    virtual void AddSegment(FX_FILESIZE offset, size_t size) = 0;
};

CPDF_DataAvail(FileAvail* pFileAvail,
               const RetainPtr<IFX_SeekableReadStream>& pFileRead,
               bool bSupportHintTable);
~CPDF_DataAvail() override;

// CPDF_Document::Observer:
void OnObservableDestroyed() override;

DocAvailStatus IsDocAvail(DownloadHints* pHints);
DocAvailStatus IsPageAvail(uint32_t dwPage, DownloadHints* pHints);
DocFormStatus IsFormAvail(DownloadHints* pHints);
DocLinearizationStatus IsLinearizedPDF();
int GetPageCount() const;
CPDF_Dictionary* GetPageDictionary(int index) const;
RetainPtr<CPDF_ReadValidator> GetValidator() const;

std::pair<CPDF_Parser::Error, std::unique_ptr<CPDF_Document>> ParseDocument(
    std::unique_ptr<CPDF_Document::RenderDataIface> pRenderData,
    std::unique_ptr<CPDF_Document::PageDataIface> pPageData,
    const char* password);

const CPDF_HintTables* GetHintTables() const { return m_pHintTables.get(); }

private:
class PageNode {
public:
    PageNode();
    ~PageNode();

    PDF_PAGENODE_TYPE m_type;
    uint32_t m_dwPageNo;
    std::vector<std::unique_ptr<PageNode>> m_ChildNodes;
};
```

```
static const int kMaxPageRecursionDepth = 1024;

bool CheckDocStatus();
bool CheckHeader();
bool CheckFirstPage();
bool CheckHintTables();
bool CheckRoot();
bool CheckInfo();
bool CheckPages();
bool CheckPage();
DocAvailStatus CheckResources(CPDF_Dictionary* page);
DocFormStatus CheckAcroForm();
bool CheckPageStatus();

DocAvailStatus CheckHeaderAndLinearized();
RetainPtr<CPDF_Object> ParseIndirectObjectAt(
    FX_FILESIZE pos,
    uint32_t objnum,
    CPDF_IndirectObjectHolder* pObjList) const;
RetainPtr<CPDF_Object> GetObject(uint32_t objnum, bool* pExistInFile);
bool GetPageKids(CPDF_Object* pPages);
bool PreparePageItem();
bool LoadPages();
bool CheckAndLoadAllXref();
bool LoadAllFile();
DocAvailStatus CheckLinearizedData();

bool CheckPage(uint32_t dwPage);
bool LoadDocPages();
bool LoadDocPage(uint32_t dwPage);
bool CheckPageNode(const PageNode& pageNode,
    int32_t iPage,
    int32_t& iCount,
    int level);
bool CheckUnknownPageNode(uint32_t dwPageNo, PageNode* pPageNode);
bool CheckArrayPageNode(uint32_t dwPageNo, PageNode* pPageNode);
bool CheckPageCount();
bool IsFirstCheck(uint32_t dwPage);
void ResetFirstCheck(uint32_t dwPage);
bool ValidatePage(uint32_t dwPage) const;
CPDF_SyntaxParser* GetSyntaxParser() const;

RetainPtr<CPDF_ReadValidator> m_pFileRead;
CPDF_Parser m_parser;
RetainPtr<CPDF_Dictionary> m_pRoot;
std::unique_ptr<CPDF_LinearizedHeader> m_pLinearized;
bool m_bDocAvail = false;
std::unique_ptr<CPDF_CrossRefAvail> m_pCrossRefAvail;
PDF_DATAAVAIL_STATUS m_docStatus = PDF_DATAAVAIL_HEADER;
const FX_FILESIZE m_dwFileLen;
UnownedPtr<CPDF_Document> m_pDocument;
std::vector<uint32_t> m_PageObjList;
uint32_t m_PagesObjNum = 0;
bool m_bLinearedDataOK = false;
bool m_bMainXRefLoadTried = false;
bool m_bMainXRefLoadedOK = false;
bool m_bPagesTreeLoad = false;
bool m_bPagesLoad = false;
std::unique_ptr<CPDF_PageObjectAvail> m_pFormAvail;
std::vector<RetainPtr<CPDF_Object>> m_PagesArray;
bool m_bTotalLoadPageTree = false;
bool m_bCurPageDictLoadOK = false;
```

```
PageNode m_PageNode;
std::set<uint32_t> m_pageMapCheckState;
std::set<uint32_t> m_pagesLoadState;
std::unique_ptr<CPDF_HintTables> m_pHintTables;
const bool m_bSupportHintTable;
std::map<uint32_t, std::unique_ptr<CPDF_PageObjectAvail>> m_PagesObjAvail;
std::map<const CPDF_Object*, std::unique_ptr<CPDF_PageObjectAvail>>
    m_PagesResourcesAvail;
bool m_bHeaderAvail = false;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_DATA_AVAIL_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.  
  
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PARSER_CPDF_DICTIONARY_H_  
#define CORE_FPDFAPI_PARSER_CPDF_DICTIONARY_H_
```

```
#include <map>  
#include <memory>  
#include <set>  
#include <utility>  
#include <vector>
```

```
#include "core/fpdfapi/parser/cpdf_object.h"  
#include "core/fxcrt/fx_coordinates.h"  
#include "core/fxcrt/fx_string.h"  
#include "core/fxcrt/retain_ptr.h"  
#include "core/fxcrt/string_pool_template.h"  
#include "core/fxcrt/weak_ptr.h"  
#include "third_party/base/logging.h"  
#include "third_party/base/ptr_util.h"
```

```
class CPDF_IndirectObjectHolder;
```

```
class CPDF_Dictionary final : public CPDF_Object {  
public:
```

```
    using const_iterator =  
        std::map<ByteString, RetainPtr<CPDF_Object>>::const_iterator;
```

```
    template <typename T, typename... Args>
```

```
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);
```

```
    // CPDF_Object:
```

```
    Type GetType() const override;  
    RetainPtr<CPDF_Object> Clone() const override;  
    CPDF_Dictionary* GetDict() override;  
    const CPDF_Dictionary* GetDict() const override;  
    bool IsDictionary() const override;  
    CPDF_Dictionary* AsDictionary() override;  
    const CPDF_Dictionary* AsDictionary() const override;  
    bool WriteTo(IFX_ArchiveStream* archive,  
                const CPDF_Encryptor* encryptor) const override;
```

```
    bool IsLocked() const { return !!m_LockCount; }
```

```
    size_t size() const { return m_Map.size(); }
```

```
    const CPDF_Object* GetObjectFor(const ByteString& key) const;
```

```
    CPDF_Object* GetObjectFor(const ByteString& key);
```

```
    const CPDF_Object* GetDirectObjectFor(const ByteString& key) const;
```

```
    CPDF_Object* GetDirectObjectFor(const ByteString& key);
```

```
    ByteString GetStringFor(const ByteString& key) const;
```

```
    ByteString GetStringFor(const ByteString& key,  
                            const ByteString& default_str) const;
```

```
    WideString GetUnicodeTextFor(const ByteString& key) const;
```

```
    int GetIntegerFor(const ByteString& key) const;
```

```
    int GetIntegerFor(const ByteString& key, int default_int) const;
```

```
    bool GetBooleanFor(const ByteString& key, bool bDefault) const;
```

```
    float GetNumberFor(const ByteString& key) const;
```

```
    const CPDF_Dictionary* GetDictFor(const ByteString& key) const;
```

```
    CPDF_Dictionary* GetDictFor(const ByteString& key);
```

```
    const CPDF_Stream* GetStreamFor(const ByteString& key) const;
```

```

CPDF_Stream* GetStreamFor(const ByteString& key);
const CPDF_Array* GetArrayFor(const ByteString& key) const;
CPDF_Array* GetArrayFor(const ByteString& key);
CFX_FloatRect GetRectFor(const ByteString& key) const;
CFX_Matrix GetMatrixFor(const ByteString& key) const;
float GetFloatFor(const ByteString& key) const { return GetNumberFor(key); }

```

```

bool KeyExist(const ByteString& key) const;
std::vector<ByteString> GetKeys() const;

```

```

// Creates a new object owned by the dictionary and returns an unowned
// pointer to it. Prefer using these templates over calls to SetFor(),
// since by creating a new object with no previous references, they ensure
// cycles can not be introduced.

```

```

template <typename T, typename... Args>
typename std::enable_if<!CanInternStrings<T>::value, T*>::type SetNewFor(
    const ByteString& key,
    Args&&... args) {
    CHECK(!IsLocked());
    return static_cast<T*>(
        SetFor(key, pdfium::MakeRetain<T>(std::forward<Args>(args)...)));
}

```

```

template <typename T, typename... Args>
typename std::enable_if<CanInternStrings<T>::value, T*>::type SetNewFor(
    const ByteString& key,
    Args&&... args) {
    CHECK(!IsLocked());
    return static_cast<T*>(SetFor(
        key, pdfium::MakeRetain<T>(m_pPool, std::forward<Args>(args)...)));
}

```

```

// Convenience functions to convert native objects to array form.

```

```

void SetRectFor(const ByteString& key, const CFX_FloatRect& rect);
void SetMatrixFor(const ByteString& key, const CFX_Matrix& matrix);

```

```

// Set* functions invalidate iterators for the element with the key |key|.
// Takes ownership of |pObj|, returns an unowned pointer to it.
CPDF_Object* SetFor(const ByteString& key, RetainPtr<CPDF_Object> pObj);

```

```

void ConvertToIndirectObjectFor(const ByteString& key,
                                CPDF_IndirectObjectHolder* pHolder);

```

```

// Invalidates iterators for the element with the key |key|.
RetainPtr<CPDF_Object> RemoveFor(const ByteString& key);

```

```

// Invalidates iterators for the element with the key |oldkey|.
void ReplaceKey(const ByteString& oldkey, const ByteString& newkey);

```

```

WeakPtr<ByteStringPool> GetByteStringPool() const { return m_pPool; }

```

```
private:
```

```
friend class CPDF_DictionaryLocker;
```

```

CPDF_Dictionary();
explicit CPDF_Dictionary(const WeakPtr<ByteStringPool>& pPool);
~CPDF_Dictionary() override;

```

```

ByteString MaybeIntern(const ByteString& str);
RetainPtr<CPDF_Object> CloneNonCyclic(
    bool bDirect,
    std::set<const CPDF_Object*>* visited) const override;

```

```
mutable uint32_t m_LockCount = 0;
```

```
WeakPtr<ByteStringPool> m_pPool;
std::map<ByteString, RetainPtr<CPDF_Object>> m_Map;
};

class CPDF_DictionaryLocker {
public:
    using const_iterator = CPDF_Dictionary::const_iterator;

    explicit CPDF_DictionaryLocker(const CPDF_Dictionary* pDictionary);
    ~CPDF_DictionaryLocker();

    const_iterator begin() const {
        CHECK(m_pDictionary->IsLocked());
        return m_pDictionary->m_Map.begin();
    }
    const_iterator end() const {
        CHECK(m_pDictionary->IsLocked());
        return m_pDictionary->m_Map.end();
    }

private:
    RetainPtr<const CPDF_Dictionary> const m_pDictionary;
};

inline CPDF_Dictionary* ToDictionary(CPDF_Object* obj) {
    return obj ? obj->AsDictionary() : nullptr;
}

inline const CPDF_Dictionary* ToDictionary(const CPDF_Object* obj) {
    return obj ? obj->AsDictionary() : nullptr;
}

inline RetainPtr<CPDF_Dictionary> ToDictionary(RetainPtr<CPDF_Object> obj) {
    return RetainPtr<CPDF_Dictionary>(ToDictionary(obj.Get()));
}

#endif // CORE_FPDFAPI_PARSER_CPDF_DICTIONARY_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_DOCUMENT_H_
#define CORE_FPDFAPI_PARSER_CPDF_DOCUMENT_H_

#include <functional>
#include <memory>
#include <set>
#include <utility>
#include <vector>

#include "build/build_config.h"
#include "core/fpdfapi/parser/cpdf_object.h"
#include "core/fpdfapi/parser/cpdf_parser.h"
#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CFX_Matrix;
class CPDF_LinearizedHeader;
class CPDF_Object;
class CPDF_ReadValidator;
class CPDF_StreamAcc;
class IFX_SeekableReadStream;
class JBig2_DocumentContext;

#define FPDFPERM_MODIFY 0x0008
#define FPDFPERM_ANNOT_FORM 0x0020
#define FPDFPERM_FILL_FORM 0x0100
#define FPDFPERM_EXTRACT_ACCESS 0x0200

class CPDF_Document : public Observable,
                      public CPDF_Parser::ParsedObjectsHolder {
public:
  // Type from which the XFA extension can subclass itself.
  class Extension {
public:
    virtual ~Extension() = default;
    virtual CPDF_Document* GetPDFDoc() const = 0;
    virtual int GetPageCount() const = 0;
    virtual void DeletePage(int page_index) = 0;
    virtual uint32_t GetUserPermissions() const = 0;
    virtual bool ContainsExtensionForm() const = 0;
    virtual bool ContainsExtensionFullForm() const = 0;
    virtual bool ContainsExtensionForegroundForm() const = 0;
  };

  class LinkListIface {
public:
    // CPDF_Document merely helps manage the lifetime.
    virtual ~LinkListIface() = default;
  };

  class PageDataIface {
public:
    PageDataIface();
    virtual ~PageDataIface();

    virtual void ClearStockFont() = 0;
  };
};
```

```
virtual RetainPtr<CPDF_StreamAcc> GetFontFileStreamAcc(
    const CPDF_Stream* pFontStream) = 0;
virtual void MaybePurgeFontFileStreamAcc(
    const CPDF_Stream* pFontStream) = 0;

void SetDocument(CPDF_Document* pDoc) { m_pDoc = pDoc; }
CPDF_Document* GetDocument() const { return m_pDoc.Get(); }

private:
    UnownedPtr<CPDF_Document> m_pDoc;
};

class RenderDataIface {
public:
    RenderDataIface();
    virtual ~RenderDataIface();

    void SetDocument(CPDF_Document* pDoc) { m_pDoc = pDoc; }
    CPDF_Document* GetDocument() const { return m_pDoc.Get(); }

private:
    UnownedPtr<CPDF_Document> m_pDoc;
};

static const int kPageMaxNum = 0xFFFFF;

CPDF_Document(std::unique_ptr<RenderDataIface> pRenderData,
               std::unique_ptr<PageDataIface> pPageData);
~CPDF_Document() override;

Extension* GetExtension() const { return m_pExtension.get(); }
void SetExtension(std::unique_ptr<Extension> pExt) {
    m_pExtension = std::move(pExt);
}

CPDF_Parser* GetParser() const { return m_pParser.get(); }
CPDF_Dictionary* GetRoot() const { return m_pRootDict.Get(); }
CPDF_Dictionary* GetInfo();

void DeletePage(int iPage);
int GetPageCount() const;
bool IsPageLoaded(int iPage) const;
CPDF_Dictionary* GetPageDictionary(int iPage);
int GetPageIndex(uint32_t objnum);
uint32_t GetUserPermissions() const;

// Returns a valid pointer, unless it is called during destruction.
PageDataIface* GetPageData() const { return m_pDocPage.get(); }
RenderDataIface* GetRenderData() const { return m_pDocRender.get(); }

void SetPageObjNum(int iPage, uint32_t objNum);

std::unique_ptr<JBig2_DocumentContext>* CodecContext() {
    return &m_pCodecContext;
}
LinkedListIface* GetLinksContext() const { return m_pLinksContext.get(); }
void SetLinksContext(std::unique_ptr<LinkedListIface> pContext) {
    m_pLinksContext = std::move(pContext);
}

// CPDF_Parser::ParsedObjectsHolder overrides:
bool TryInit() override;
```

```

CPDF_Parser::Error LoadDoc(
    const RetainPtr<IFX_SeekableReadStream>& pFileAccess,
    const char* password);
CPDF_Parser::Error LoadLinearizedDoc(
    const RetainPtr<CPDF_ReadValidator>& validator,
    const char* password);
bool has_valid_cross_reference_table() const {
    return m_bHasValidCrossReferenceTable;
}

void LoadPages();
void CreateNewDoc();
CPDF_Dictionary* CreateNewPage(int iPage);

void IncrementParsedPageCount() { ++m_ParsedPageCount; }
uint32_t GetParsedPageCountForTesting() { return m_ParsedPageCount; }

protected:
class StockFontClearer {
public:
    explicit StockFontClearer(CPDF_Document::PageDataIface* pPageData);
    ~StockFontClearer();

private:
    UnownedPtr<CPDF_Document::PageDataIface> const m_pPageData;
};

// Retrieve page count information by getting count value from the tree nodes
int RetrievePageCount();
// When this method is called, m_pTreeTraversal[level] exists.
CPDF_Dictionary* TraversePDFPages(int iPage, int* nPagesToGo, size_t level);
int FindPageIndex(const CPDF_Dictionary* pNode,
    uint32_t* skip_count,
    uint32_t objnum,
    int* index,
    int level) const;
RetainPtr<CPDF_Object> ParseIndirectObject(uint32_t objnum) override;
const CPDF_Dictionary* GetPagesDict() const;
CPDF_Dictionary* GetPagesDict();
bool InsertDeletePDFPage(CPDF_Dictionary* pPages,
    int nPagesToGo,
    CPDF_Dictionary* pPageDict,
    bool bInsert,
    std::set<CPDF_Dictionary*>* pVisited);
bool InsertNewPage(int iPage, CPDF_Dictionary* pPageDict);
void ResetTraversal();
void SetParser(std::unique_ptr<CPDF_Parser> pParser);
CPDF_Parser::Error HandleLoadResult(CPDF_Parser::Error error);

std::unique_ptr<CPDF_Parser> m_pParser;
RetainPtr<CPDF_Dictionary> m_pRootDict;
RetainPtr<CPDF_Dictionary> m_pInfoDict;

// Vector of pairs to know current position in the page tree. The index in the
// vector corresponds to the level being described. The pair contains a
// pointer to the dictionary being processed at the level, and an index of the
// of the child being processed within the dictionary's /Kids array.
std::vector<std::pair<CPDF_Dictionary*, size_t>> m_pTreeTraversal;

// True if the CPDF_Parser succeeded without having to rebuild the cross
// reference table.
bool m_bHasValidCrossReferenceTable = false;

```

```
// Index of the next page that will be traversed from the page tree.
bool m_bReachedMaxPageLevel = false;
int m_iNextPageToTraverse = 0;
uint32_t m_ParsedPageCount = 0;

std::unique_ptr<RenderDataIface> m_pDocRender;
std::unique_ptr<PageDataIface> m_pDocPage; // Must be after |m_pDocRender|.
std::unique_ptr<JBig2_DocumentContext> m_pCodecContext;
std::unique_ptr<LinkListIface> m_pLinksContext;
std::vector<uint32_t> m_PageList; // Page number to page's dict objnum.

// Must be second to last.
StockFontClearer m_StockFontClearer;

// Must be last. Destroy the extension before any non-extension teardown.
std::unique_ptr<Extension> m_pExtension;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_DOCUMENT_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_ENCRYPTOR_H_
#define CORE_FPDFAPI_PARSER_CPDF_ENCRYPTOR_H_

#include <stdint.h>

#include <vector>

#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/span.h"

class CPDF_CryptoHandler;

class CPDF_Encryptor {
public:
  CPDF_Encryptor(CPDF_CryptoHandler* pHandler, int objnum);
  ~CPDF_Encryptor();

  std::vector<uint8_t> Encrypt(pdfium::span<const uint8_t> src_data) const;

private:
  UnownedPtr<CPDF_CryptoHandler> const m_pHandler;
  const int m_ObjNum;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_ENCRYPTOR_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_FLATEENCODER_H_
#define CORE_FPDFAPI_PARSER_CPDF_FLATEENCODER_H_

#include <memory>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/maybe_owned.h"
#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/span.h"

class CPDF_Dictionary;
class CPDF_Stream;
class CPDF_StreamAcc;

class CPDF_FlateEncoder {
public:
  CPDF_FlateEncoder(const CPDF_Stream* pStream, bool bFlateEncode);
  ~CPDF_FlateEncoder();

  void CloneDict();
  CPDF_Dictionary* GetClonedDict();

  // Returns |m_pClonedDict| if it is valid. Otherwise returns |m_pDict|.
  const CPDF_Dictionary* GetDict() const;

  pdfium::span<const uint8_t> GetSpan() const {
    return pdfium::make_span(m_pData.Get(), m_dwSize);
  }

private:
  RetainPtr<CPDF_StreamAcc> m_pAcc;

  uint32_t m_dwSize;
  MaybeOwned<uint8_t, FxFreeDeleter> m_pData;

  // Only one of these two pointers is valid at any time.
  RetainPtr<const CPDF_Dictionary> m_pDict;
  RetainPtr<CPDF_Dictionary> m_pClonedDict;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_FLATEENCODER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_PARSER_CPDF_HINT_TABLES_H
```

```
#define CORE_FPDFAPI_PARSER_CPDF_HINT_TABLES_H
```

```
#include <memory>
```

```
#include <vector>
```

```
#include "core/fpdfapi/parser/cpdf_data_avail.h"
```

```
#include "core/fxcrt/unowned_ptr.h"
```

```
class CFX_BitStream;
```

```
class CPDF_LinearizedHeader;
```

```
class CPDF_ReadValidator;
```

```
class CPDF_Stream;
```

```
class CPDF_SyntaxParser;
```

```
class CPDF_HintTables {
```

```
  public:
```

```
    struct SharedObjGroupInfo {
```

```
      FX_FILESIZE m_szOffset = 0;
```

```
      uint32_t m_dwLength = 0;
```

```
      uint32_t m_dwObjectsCount = 0;
```

```
      uint32_t m_dwStartObjNum = 0;
```

```
    };
```

```
  class PageInfo {
```

```
    public:
```

```
    PageInfo();
```

```
    ~PageInfo();
```

```
    void set_objects_count(uint32_t objects_count) {
```

```
      m_dwObjectsCount = objects_count;
```

```
    }
```

```
    uint32_t objects_count() const { return m_dwObjectsCount; }
```

```
    void set_page_offset(FX_FILESIZE offset) { m_szOffset = offset; }
```

```
    FX_FILESIZE page_offset() const { return m_szOffset; }
```

```
    void set_page_length(uint32_t length) { m_dwLength = length; }
```

```
    uint32_t page_length() const { return m_dwLength; }
```

```
    void set_start_obj_num(uint32_t start_obj_num) {
```

```
      m_dwStartObjNum = start_obj_num;
```

```
    }
```

```
    uint32_t start_obj_num() const { return m_dwStartObjNum; }
```

```
    void AddIdentifier(uint32_t Identifier) {
```

```
      m_dwIdentifierArray.push_back(Identifier);
```

```
    }
```

```
    const std::vector<uint32_t>& Identifiers() const {
```

```
      return m_dwIdentifierArray;
```

```
    }
```

```
  private:
```

```
    uint32_t m_dwObjectsCount = 0;
```

```
    FX_FILESIZE m_szOffset = 0;
```

```
    uint32_t m_dwLength = 0;
```

```
uint32_t m_dwStartObjNum = 0;
std::vector<uint32_t> m_dwIdentifierArray;

PageInfo(const PageInfo& other) = delete;
PageInfo& operator=(const PageInfo&) = delete;
};

static std::unique_ptr<CPDF_HintTables> Parse(
    CPDF_SyntaxParser* parser,
    CPDF_LinearizedHeader* pLinearized);

CPDF_HintTables(CPDF_ReadValidator* pValidator,
    CPDF_LinearizedHeader* pLinearized);
virtual ~CPDF_HintTables();

bool GetPagePos(uint32_t index,
    FX_FILESIZE* szPageStartPos,
    FX_FILESIZE* szPageLength,
    uint32_t* dwObjNum) const;

CPDF_DataAvail::DocAvailStatus CheckPage(uint32_t index);

bool LoadHintStream(CPDF_Stream* pHintStream);

const std::vector<PageInfo>& PageInfos() const { return m_PageInfos; }
const std::vector<SharedObjGroupInfo>& SharedGroupInfos() const {
    return m_SharedObjGroupInfos;
}

FX_FILESIZE GetFirstPageObjOffset() const { return m_szFirstPageObjOffset; }

protected:
bool ReadPageHintTable(CFX_BitStream* hStream);
bool ReadSharedObjHintTable(CFX_BitStream* hStream, uint32_t offset);

private:
FX_FILESIZE HintsOffsetToFileOffset(uint32_t hints_offset) const;

// Owned by |m_pDataAvail|.
UnownedPtr<CPDF_ReadValidator> m_pValidator;

// Owned by |m_pDataAvail|.
UnownedPtr<CPDF_LinearizedHeader> const m_pLinearized;

uint32_t m_nFirstPageSharedObjs;
FX_FILESIZE m_szFirstPageObjOffset;

std::vector<PageInfo> m_PageInfos;
std::vector<SharedObjGroupInfo> m_SharedObjGroupInfos;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_HINT_TABLES_H
```

```

// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_INDIRECT_OBJECT HOLDER_H_
#define CORE_FPDFAPI_PARSER_CPDF_INDIRECT_OBJECT HOLDER_H_

#include <map>
#include <memory>
#include <type_traits>
#include <utility>
#include <vector>

#include "core/fpdfapi/parser/cpdf_object.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/string_pool_template.h"
#include "core/fxcrt/weak_ptr.h"
#include "third_party/base/ptr_util.h"

class CPDF_IndirectObjectHolder {
public:
    using const_iterator =
        std::map<uint32_t, RetainPtr<CPDF_Object>>::const_iterator;

    CPDF_IndirectObjectHolder();
    virtual ~CPDF_IndirectObjectHolder();

    CPDF_Object* GetIndirectObject(uint32_t objnum) const;
    virtual CPDF_Object* GetOrParseIndirectObject(uint32_t objnum);
    void DeleteIndirectObject(uint32_t objnum);

    // Creates and adds a new object owned by the indirect object holder,
    // and returns an unowned pointer to it. We have a special case to
    // handle objects that can intern strings from our ByteStringPool.
    template <typename T, typename... Args>
    typename std::enable_if<!CanInternStrings<T>::value, T*>::type NewIndirect(
        Args&&... args) {
        return static_cast<T*>(
            AddIndirectObject(pdfium::MakeRetain<T>(std::forward<Args>(args)...)));
    }
    template <typename T, typename... Args>
    typename std::enable_if<CanInternStrings<T>::value, T*>::type NewIndirect(
        Args&&... args) {
        return static_cast<T*>(AddIndirectObject(
            pdfium::MakeRetain<T>(m_pByteStringPool, std::forward<Args>(args)...)));
    }

    // Creates and adds a new object not owned by the indirect object holder,
    // but which can intern strings from it.
    template <typename T, typename... Args>
    typename std::enable_if<CanInternStrings<T>::value, RetainPtr<T>>::type New(
        Args&&... args) {
        return pdfium::MakeRetain<T>(m_pByteStringPool,
            std::forward<Args>(args)...);
    }

    // Takes ownership of |pObj|, returns unowned pointer to it.
    CPDF_Object* AddIndirectObject(RetainPtr<CPDF_Object> pObj);

    // Always takes ownership of |pObj|, return true if higher generation number.
    bool ReplaceIndirectObjectIfHigherGeneration(uint32_t objnum,

```

```
        RetainPtr<CPDF_Object> pObj);

uint32_t GetLastObjNum() const { return m_LastObjNum; }
void SetLastObjNum(uint32_t objnum) { m_LastObjNum = objnum; }

WeakPtr<ByteStringPool> GetByteStringPool() const {
    return m_pByteStringPool;
}

const_iterator begin() const { return m_IndirectObjs.begin(); }
const_iterator end() const { return m_IndirectObjs.end(); }

protected:
    virtual RetainPtr<CPDF_Object> ParseIndirectObject(uint32_t objnum);

private:
    uint32_t m_LastObjNum;
    std::map<uint32_t, RetainPtr<CPDF_Object>> m_IndirectObjs;
    WeakPtr<ByteStringPool> m_pByteStringPool;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_INDIRECT_OBJECT HOLDER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_LINEARIZED_HEADER_H
#define CORE_FPDFAPI_PARSER_CPDF_LINEARIZED_HEADER_H

#include <memory>

#include "core/fxcrt/fx_system.h"

class CPDF_Dictionary;
class CPDF_Object;
class CPDF_SyntaxParser;

class CPDF_LinearizedHeader {
public:
  ~CPDF_LinearizedHeader();
  static std::unique_ptr<CPDF_LinearizedHeader> Parse(
    CPDF_SyntaxParser* parser);

  // Will only return values > 0.
  FX_FILESIZE GetFileSize() const { return m_szFileSize; }
  uint32_t GetFirstPageNo() const { return m_dwFirstPageNo; }
  // Will only return values > 0.
  FX_FILESIZE GetMainXRefTableFirstEntryOffset() const {
    return m_szMainXRefTableFirstEntryOffset;
  }
  uint32_t GetPageCount() const { return m_PageCount; }
  // Will only return values > 0.
  FX_FILESIZE GetFirstPageEndOffset() const { return m_szFirstPageEndOffset; }
  // Will only return values > 0.
  uint32_t GetFirstPageObjNum() const { return m_FirstPageObjNum; }
  // Will only return values > 0.
  FX_FILESIZE GetLastXRefOffset() const { return m_szLastXRefOffset; }

  bool HasHintTable() const;
  // Will only return values > 0.
  FX_FILESIZE GetHintStart() const { return m_szHintStart; }
  uint32_t GetHintLength() const { return m_HintLength; }

protected:
  CPDF_LinearizedHeader(const CPDF_Dictionary* pDict,
    FX_FILESIZE szLastXRefOffset);

private:
  const FX_FILESIZE m_szFileSize;
  const uint32_t m_dwFirstPageNo;
  const FX_FILESIZE m_szMainXRefTableFirstEntryOffset;
  const uint32_t m_PageCount;
  const FX_FILESIZE m_szFirstPageEndOffset;
  const uint32_t m_FirstPageObjNum;
  const FX_FILESIZE m_szLastXRefOffset;
  FX_FILESIZE m_szHintStart = 0;
  uint32_t m_HintLength = 0;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_LINEARIZED_HEADER_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_NAME_H_
#define CORE_FPDFAPI_PARSER_CPDF_NAME_H_

#include <memory>

#include "core/fpdfapi/parser/cpdf_object.h"
#include "core/fxcrt/string_pool_template.h"
#include "core/fxcrt/weak_ptr.h"

class CPDF_Name final : public CPDF_Object {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    // CPDF_Object:
    Type GetType() const override;
    RetainPtr<CPDF_Object> Clone() const override;
    ByteString GetString() const override;
    WideString GetUnicodeText() const override;
    void SetString(const ByteString& str) override;
    bool IsName() const override;
    CPDF_Name* AsName() override;
    const CPDF_Name* AsName() const override;
    bool WriteTo(IFX_ArchiveStream* archive,
                const CPDF_Encryptor* encryptor) const override;

private:
    CPDF_Name(WeakPtr<ByteStringPool> pPool, const ByteString& str);
    ~CPDF_Name() override;

    ByteString m_Name;
};

inline CPDF_Name* ToName(CPDF_Object* obj) {
    return obj ? obj->AsName() : nullptr;
}

inline const CPDF_Name* ToName(const CPDF_Object* obj) {
    return obj ? obj->AsName() : nullptr;
}

#endif // CORE_FPDFAPI_PARSER_CPDF_NAME_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_NULL_H_
#define CORE_FPDFAPI_PARSER_CPDF_NULL_H_

#include <memory>

#include "core/fpdfapi/parser/cpdf_object.h"

class CPDF_Null final : public CPDF_Object {
public:
  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  // CPDF_Object.
  Type GetType() const override;
  RetainPtr<CPDF_Object> Clone() const override;
  bool WriteTo(IFX_ArchiveStream* archive,
               const CPDF_Encryptor* encryptor) const override;
  bool IsNull() const override;

private:
  CPDF_Null();
};

#endif // CORE_FPDFAPI_PARSER_CPDF_NULL_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_NUMBER_H_
#define CORE_FPDFAPI_PARSER_CPDF_NUMBER_H_

#include <memory>

#include "core/fpdfapi/parser/cpdf_object.h"
#include "core/fxcrt/fx_number.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"

class CPDF_Number final : public CPDF_Object {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    // CPDF_Object:
    Type GetType() const override;
    RetainPtr<CPDF_Object> Clone() const override;
    ByteString GetString() const override;
    float GetNumber() const override;
    int GetInteger() const override;
    void SetString(const ByteString& str) override;
    bool IsNumber() const override;
    CPDF_Number* AsNumber() override;
    const CPDF_Number* AsNumber() const override;
    bool WriteTo(IFX_ArchiveStream* archive,
                const CPDF_Encryptor* encryptor) const override;

    bool IsInteger() const { return m_Number.IsInteger(); }

private:
    CPDF_Number();
    explicit CPDF_Number(int value);
    explicit CPDF_Number(float value);
    explicit CPDF_Number(ByteStringView str);
    ~CPDF_Number() override;

    FX_Number m_Number;
};

inline CPDF_Number* ToNumber(CPDF_Object* obj) {
    return obj ? obj->AsNumber() : nullptr;
}

inline const CPDF_Number* ToNumber(const CPDF_Object* obj) {
    return obj ? obj->AsNumber() : nullptr;
}

#endif // CORE_FPDFAPI_PARSER_CPDF_NUMBER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FPDFAPI_PARSER_CPDF_OBJECT_AVAIL_H_  
#define CORE_FPDFAPI_PARSER_CPDF_OBJECT_AVAIL_H_
```

```
#include <set>  
#include <stack>
```

```
#include "core/fpdfapi/parser/cpdf_data_avail.h"  
#include "core/fxcrt/retain_ptr.h"  
#include "core/fxcrt/unowned_ptr.h"
```

```
class CPDF_Object;  
class CPDF_Reference;  
class CPDF_IndirectObjectHolder;  
class CPDF_ReadValidator;
```

```
// Helper for check availability of object tree.
```

```
class CPDF_ObjectAvail {  
public:  
    CPDF_ObjectAvail(const RetainPtr<CPDF_ReadValidator>& validator,  
                    CPDF_IndirectObjectHolder* holder,  
                    CPDF_Object* root);  
    CPDF_ObjectAvail(const RetainPtr<CPDF_ReadValidator>& validator,  
                    CPDF_IndirectObjectHolder* holder,  
                    uint32_t obj_num);  
    virtual ~CPDF_ObjectAvail();  
  
    CPDF_DataAvail::DocAvailStatus CheckAvail();  
  
protected:  
    virtual bool ExcludeObject(const CPDF_Object* object) const;  
  
private:  
    bool LoadRootObject();  
    bool CheckObjects();  
    bool AppendObjectSubRefs(const CPDF_Object* object,  
                            std::stack<uint32_t>* refs) const;  
    void CleanMemory();  
    bool HasObjectParsed(uint32_t obj_num) const;  
  
    RetainPtr<CPDF_ReadValidator> validator_;  
    UnownedPtr<CPDF_IndirectObjectHolder> holder_;  
    RetainPtr<CPDF_Object> root_;  
    std::set<uint32_t> parsed_objnums_;  
    std::stack<uint32_t> non_parsed_objects_;  
};  
  
#endif // CORE_FPDFAPI_PARSER_CPDF_OBJECT_AVAIL_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_OBJECT_H_
#define CORE_FPDFAPI_PARSER_CPDF_OBJECT_H_

#include <memory>
#include <set>
#include <type_traits>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"

class CPDF_Array;
class CPDF_Boolean;
class CPDF_Dictionary;
class CPDF_Encryptor;
class CPDF_IndirectObjectHolder;
class CPDF_Name;
class CPDF_Null;
class CPDF_Number;
class CPDF_Reference;
class CPDF_Stream;
class CPDF_String;
class IFX_ArchiveStream;

class CPDF_Object : public Retainable {
public:
    static const uint32_t kInvalidObjNum = static_cast<uint32_t>(-1);
    enum Type {
        kBoolean = 1,
        kNumber,
        kString,
        kName,
        kArray,
        kDictionary,
        kStream,
        kNullobj,
        kReference
    };
};

virtual Type GetType() const = 0;
uint32_t GetObjNum() const { return m_ObjNum; }
void SetObjNum(uint32_t objnum) { m_ObjNum = objnum; }
uint32_t GetGenNum() const { return m_GenNum; }
void SetGenNum(uint32_t gennum) { m_GenNum = gennum; }
bool IsInline() const { return m_ObjNum == 0; }

// Create a deep copy of the object.
virtual RetainPtr<CPDF_Object> Clone() const = 0;

// Create a deep copy of the object except any reference object be
// copied to the object it points to directly.
virtual RetainPtr<CPDF_Object> CloneDirectObject() const;

virtual CPDF_Object* GetDirect();
virtual const CPDF_Object* GetDirect() const;
virtual ByteString GetString() const;
virtual WideString GetUnicodeText() const;
virtual float GetNumber() const;
```

```

virtual int GetInteger() const;
virtual CPDF_Dictionary* GetDict();
virtual const CPDF_Dictionary* GetDict() const;

virtual void SetString(const ByteString& str);

virtual bool IsArray() const;
virtual bool IsBoolean() const;
virtual bool IsDictionary() const;
virtual bool IsName() const;
virtual bool IsNumber() const;
virtual bool IsReference() const;
virtual bool IsStream() const;
virtual bool IsString() const;
virtual bool IsNull() const;

virtual CPDF_Array* AsArray();
virtual const CPDF_Array* AsArray() const;
virtual CPDF_Boolean* AsBoolean();
virtual const CPDF_Boolean* AsBoolean() const;
virtual CPDF_Dictionary* AsDictionary();
virtual const CPDF_Dictionary* AsDictionary() const;
virtual CPDF_Name* AsName();
virtual const CPDF_Name* AsName() const;
virtual CPDF_Number* AsNumber();
virtual const CPDF_Number* AsNumber() const;
virtual CPDF_Reference* AsReference();
virtual const CPDF_Reference* AsReference() const;
virtual CPDF_Stream* AsStream();
virtual const CPDF_Stream* AsStream() const;
virtual CPDF_String* AsString();
virtual const CPDF_String* AsString() const;

virtual bool WriteTo(IFX_ArchiveStream* archive,
                    const CPDF_Encryptor* encryptor) const = 0;

// Create a deep copy of the object with the option to either
// copy a reference object or directly copy the object it refers to
// when |bDirect| is true.
// Also check cyclic reference against |pVisited|, no copy if it is found.
// Complex objects should implement their own CloneNonCyclic()
// function to properly check for possible loop.
virtual RetainPtr<CPDF_Object> CloneNonCyclic(
    bool bDirect,
    std::set<const CPDF_Object*>* pVisited) const;

// Return a reference to itself.
// The object must be direct (!IsInlined).
virtual RetainPtr<CPDF_Object> MakeReference(
    CPDF_IndirectObjectHolder* holder) const;

protected:
CPDF_Object() = default;
CPDF_Object(const CPDF_Object& src) = delete;
~CPDF_Object() override;

RetainPtr<CPDF_Object> CloneObjectNonCyclic(bool bDirect) const;

uint32_t m_ObjNum = 0;
uint32_t m_GenNum = 0;
};

```

```

template <typename T>

```

```
struct CanInternStrings {
    static const bool value = std::is_same<T, CPDF_Array>::value ||
                               std::is_same<T, CPDF_Dictionary>::value ||
                               std::is_same<T, CPDF_Name>::value ||
                               std::is_same<T, CPDF_String>::value;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_OBJECT_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FPDFAPI_PARSER_CPDF_OBJECT_STREAM_H_  
#define CORE_FPDFAPI_PARSER_CPDF_OBJECT_STREAM_H_
```

```
#include <map>  
#include <memory>
```

```
#include "core/fpdfapi/parser/cpdf_object.h"  
#include "core/fxcrt/retain_ptr.h"
```

```
class CPDF_IndirectObjectHolder;  
class CPDF_Stream;  
class IFX_SeekableReadStream;
```

```
// Implementation of logic of PDF "Object Streams".  
// See "PDF 32000-1:2008" Spec. section 7.5.7.
```

```
class CPDF_ObjectStream {  
public:  
    static bool IsObjectsStreamObject(const CPDF_Object* object);  
  
    static std::unique_ptr<CPDF_ObjectStream> Create(const CPDF_Stream* stream);  
  
    ~CPDF_ObjectStream();  
  
    uint32_t obj_num() const { return obj_num_; }  
    uint32_t extends_obj_num() const { return extends_obj_num_; }  
  
    bool HasObject(uint32_t obj_number) const;  
    RetainPtr<CPDF_Object> ParseObject(CPDF_IndirectObjectHolder* pObjList,  
                                       uint32_t obj_number) const;  
    const std::map<uint32_t, uint32_t>& objects_offsets() const {  
        return objects_offsets_;  
    }  
  
protected:  
    explicit CPDF_ObjectStream(const CPDF_Stream* stream);  
  
    void Init(const CPDF_Stream* stream);  
    RetainPtr<CPDF_Object> ParseObjectAtOffset(  
        CPDF_IndirectObjectHolder* pObjList,  
        uint32_t object_offset) const;  
  
    uint32_t obj_num_ = CPDF_Object::kInvalidObjNum;  
    uint32_t extends_obj_num_ = CPDF_Object::kInvalidObjNum;  
  
    RetainPtr<IFX_SeekableReadStream> data_stream_;  
    int first_object_offset_ = 0;  
    std::map<uint32_t, uint32_t> objects_offsets_;  
};  
  
#endif // CORE_FPDFAPI_PARSER_CPDF_OBJECT_STREAM_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.
```

```
#ifndef CORE_FPDFAPI_PARSER_CPDF_OBJECT_WALKER_H
#define CORE_FPDFAPI_PARSER_CPDF_OBJECT_WALKER_H

#include <memory>
#include <stack>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Object;

// Walk on all non-null sub-objects in an object in depth, include itself,
// like in flat list.
class CPDF_ObjectWalker {
public:
  class SubobjectIterator {
public:
    virtual ~SubobjectIterator();
    virtual bool IsFinished() const = 0;
    bool IsStarted() const { return is_started_; }
    const CPDF_Object* Increment();
    const CPDF_Object* object() const { return object_.Get(); }

protected:
    explicit SubobjectIterator(const CPDF_Object* object);

    virtual const CPDF_Object* IncrementImpl() = 0;
    virtual void Start() = 0;

private:
    RetainPtr<const CPDF_Object> object_;
    bool is_started_ = false;
  };

  explicit CPDF_ObjectWalker(const CPDF_Object* root);
  ~CPDF_ObjectWalker();

  const CPDF_Object* GetNext();
  void SkipWalkIntoCurrentObject();

  size_t current_depth() const { return current_depth_; }
  const CPDF_Object* GetParent() const { return parent_object_.Get(); }
  const ByteString& dictionary_key() const { return dict_key_; }

private:
  static std::unique_ptr<SubobjectIterator> MakeIterator(
    const CPDF_Object* object);

  RetainPtr<const CPDF_Object> next_object_;
  RetainPtr<const CPDF_Object> parent_object_;
  ByteString dict_key_;
  size_t current_depth_ = 0;
  std::stack<std::unique_ptr<SubobjectIterator>> stack_;
};

class CPDF_NonConstObjectWalker final : public CPDF_ObjectWalker {
public:
  explicit CPDF_NonConstObjectWalker(CPDF_Object* root)
    : CPDF_ObjectWalker(root) {}
};
```

```
CPDF_Object* GetNext() {  
    return const_cast<CPDF_Object*>(CPDF_ObjectWalker::GetNext());  
}  
};  
  
#endif // CORE_FPDFAPI_PARSER_CPDF_OBJECT_WALKER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FPDFAPI_PARSER_CPDF_PAGE_OBJECT_AVAIL_H_  
#define CORE_FPDFAPI_PARSER_CPDF_PAGE_OBJECT_AVAIL_H_
```

```
#include "core/fpdfapi/parser/cpdf_object_avail.h"
```

```
// Helper for check availability of page's object tree.  
// Exclude references to pages.
```

```
class CPDF_PageObjectAvail final : public CPDF_ObjectAvail {  
public:  
    using CPDF_ObjectAvail::CPDF_ObjectAvail;  
    ~CPDF_PageObjectAvail() override;  
  
private:  
    bool ExcludeObject(const CPDF_Object* object) const override;  
};
```

```
#endif // CORE_FPDFAPI_PARSER_CPDF_PAGE_OBJECT_AVAIL_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_PARSER_H_
#define CORE_FPDFAPI_PARSER_CPDF_PARSER_H_

#include <limits>
#include <map>
#include <memory>
#include <set>
#include <vector>

#include "core/fpdfapi/parser/cpdf_cross_ref_table.h"
#include "core/fpdfapi/parser/cpdf_indirect_object_holder.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Array;
class CPDF_CryptoHandler;
class CPDF_Dictionary;
class CPDF_LinearizedHeader;
class CPDF_Object;
class CPDF_ObjectStream;
class CPDF_ReadValidator;
class CPDF_SecurityHandler;
class CPDF_SyntaxParser;
class IFX_SeekableReadStream;

class CPDF_Parser {
public:
  class ParsedObjectsHolder : public CPDF_IndirectObjectHolder {
public:
    virtual bool TryInit() = 0;
  };

  enum Error {
    SUCCESS = 0,
    FILE_ERROR,
    FORMAT_ERROR,
    PASSWORD_ERROR,
    HANDLER_ERROR
  };

  // A limit on the maximum object number in the xref table. Theoretical limits
  // are higher, but this may be large enough in practice.
  // Note: This was 1M, but https://crbug.com/910009 encountered a PDF with
  // object numbers in the 1.7M range. The PDF only has 10K objects, but they
  // are non-consecutive.
  static constexpr uint32_t kMaxObjectNumber = 4 * 1024 * 1024;

  static const size_t kInvalidPos = std::numeric_limits<size_t>::max();

  explicit CPDF_Parser(ParsedObjectsHolder* holder);
  CPDF_Parser();
  ~CPDF_Parser();

  Error StartParse(const RetainPtr<IFX_SeekableReadStream>& pFile,
                  const char* password);
};
```

```
Error StartLinearizedParse(const RetainPtr<CPDF_ReadValidator>& validator,
                          const char* password);

void SetPassword(const char* password) { m_Password = password; }
ByteString GetPassword() const { return m_Password; }

// Take the GetPassword() value and encode it, if necessary, based on the
// password encoding conversion.
ByteString GetEncodedPassword() const;

const CPDF_Dictionary* GetTrailer() const;
CPDF_Dictionary* GetMutableTrailerForTesting();

// Returns a new trailer which combines the last read trailer with the /Root
// and /Info from previous ones.
RetainPtr<CPDF_Dictionary> GetCombinedTrailer() const;

FX_FILESIZE GetLastXRefOffset() const { return m_LastXRefOffset; }

uint32_t GetPermissions() const;
uint32_t GetRootObjNum() const;
uint32_t GetInfoObjNum() const;
const CPDF_Array* GetIDArray() const;
CPDF_Dictionary* GetRoot() const;

const CPDF_Dictionary* GetEncryptDict() const;

RetainPtr<CPDF_Object> ParseIndirectObject(uint32_t objnum);

uint32_t GetLastObjNum() const;
bool IsValidObjectNumber(uint32_t objnum) const;
FX_FILESIZE GetObjectPositionOrZero(uint32_t objnum) const;
bool IsObjectFreeOrNull(uint32_t objnum) const;
const RetainPtr<CPDF_SecurityHandler>& GetSecurityHandler() const {
    return m_pSecurityHandler;
}
bool IsObjectFree(uint32_t objnum) const;

int GetFileVersion() const { return m_FileVersion; }
bool IsXRefStream() const { return m_bXRefStream; }

RetainPtr<CPDF_Object> ParseIndirectObjectAt(FX_FILESIZE pos,
                                             uint32_t objnum);

uint32_t GetFirstPageNo() const;
const CPDF_LinearizedHeader* GetLinearizedHeader() const {
    return m_pLinearized.get();
}

const CPDF_CrossRefTable* GetCrossRefTable() const {
    return m_CrossRefTable.get();
}

bool xref_table_rebuilt() const { return m_bXRefTableRebuilt; }

CPDF_SyntaxParser* GetSyntax() const { return m_pSyntax.get(); }

void SetLinearizedHeader(std::unique_ptr<CPDF_LinearizedHeader> pLinearized);

protected:
using ObjectType = CPDF_CrossRefTable::ObjectType;
using ObjectInfo = CPDF_CrossRefTable::ObjectInfo;
```

```
bool LoadCrossRefV4(FX_FILESIZE pos, bool bSkip);
bool RebuildCrossRef();

std::unique_ptr<CPDF_SyntaxParser> m_pSyntax;

private:
friend class cpdf_parser_BadStartXrefShouldNotBuildCrossRefTable_Test;
friend class cpdf_parser_ParseStartXrefWithHeaderOffset_Test;
friend class cpdf_parser_ParseStartXref_Test;
friend class cpdf_parser_ParseLinearizedWithHeaderOffset_Test;
friend class CPDF_DataAvail;

struct CrossRefObjData {
    uint32_t obj_num = 0;
    ObjectInfo info;
};

Error StartParseInternal();
FX_FILESIZE ParseStartXref();
bool LoadAllCrossRefV4(FX_FILESIZE xref_offset);
bool LoadAllCrossRefV5(FX_FILESIZE xref_offset);
bool LoadCrossRefV5(FX_FILESIZE* pos, bool bMainXRef);
RetainPtr<CPDF_Dictionary> LoadTrailerV4();
Error SetEncryptHandler();
void ReleaseEncryptHandler();
bool LoadLinearizedAllCrossRefV4(FX_FILESIZE main_xref_offset);
bool LoadLinearizedAllCrossRefV5(FX_FILESIZE main_xref_offset);
Error LoadLinearizedMainXrefTable();
const CPDF_ObjectStream* GetObjectStream(uint32_t object_number);
std::unique_ptr<CPDF_LinearizedHeader> ParseLinearizedHeader();
void ShrinkObjectMap(uint32_t size);
// A simple check whether the cross reference table matches with
// the objects.
bool VerifyCrossRefV4();

// If out_objects is null, the parser position will be moved to end subsection
// without additional validation.
bool ParseAndAppendCrossRefSubsectionData(
    uint32_t start_objnum,
    uint32_t count,
    std::vector<CrossRefObjData>* out_objects);
bool ParseCrossRefV4(std::vector<CrossRefObjData>* out_objects);
void MergeCrossRefObjectsData(const std::vector<CrossRefObjData>& objects);

bool InitSyntaxParser(const RetainPtr<CPDF_ReadValidator>& validator);
bool ParseFileVersion();

ObjectType GetObjectType(uint32_t objnum) const;
ObjectType GetObjectTypeFromCrossRefStreamType(
    uint32_t cross_ref_stream_type) const;

std::unique_ptr<ParsedObjectsHolder> m_pOwnedObjectsHolder;
UnownedPtr<ParsedObjectsHolder> m_pObjectsHolder;

bool m_bHasParsed = false;
bool m_bXRefStream = false;
bool m_bXRefTableRebuilt = false;
int m_FileVersion = 0;
// m_CrossRefTable must be destroyed after m_pSecurityHandler due to the
// ownership of the ID array data.
std::unique_ptr<CPDF_CrossRefTable> m_CrossRefTable;
FX_FILESIZE m_LastXrefOffset;
RetainPtr<CPDF_SecurityHandler> m_pSecurityHandler;
```

```
    ByteString m_Password;
    std::unique_ptr<CPDF_LinearizedHeader> m_pLinearized;

    // A map of object numbers to indirect streams.
    std::map<uint32_t, std::unique_ptr<CPDF_ObjectStream>> m_ObjectStreamMap;

    // All indirect object numbers that are being parsed.
    std::set<uint32_t> m_ParsingObjNums;

    uint32_t m_MetadataObjnum = 0;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_PARSER_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FPDFAPI_PARSER_CPDF_READ_VALIDATOR_H_  
#define CORE_FPDFAPI_PARSER_CPDF_READ_VALIDATOR_H_
```

```
#include "core/fpdfapi/parser/cpdf_data_avail.h"  
#include "core/fxcrt/fx_stream.h"
```

```
class CPDF_ReadValidator : public IFX_SeekableReadStream {  
public:  
    class Session {  
    public:  
        explicit Session(const RetainPtr<CPDF_ReadValidator>& validator);  
        ~Session();  
  
    private:  
        UnownedPtr<CPDF_ReadValidator> validator_;  
        bool saved_read_error_;  
        bool saved_has_unavailable_data_;  
    };  
  
    template <typename T, typename... Args>  
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);  
  
    void SetDownloadHints(CPDF_DataAvail::DownloadHints* hints) {  
        hints_ = hints;  
    }  
    bool read_error() const { return read_error_; }  
    bool has_unavailable_data() const { return has_unavailable_data_; }  
    bool has_read_problems() const {  
        return read_error() || has_unavailable_data();  
    }  
  
    void ResetErrors();  
  
    bool IsWholeFileAvailable();  
  
    bool CheckDataRangeAndRequestIfUnavailable(FX_FILESIZE offset, size_t size);  
    bool CheckWholeFileAndRequestIfUnavailable();  
  
    // IFX_SeekableReadStream overrides:  
    bool ReadBlockAtOffset(void* buffer,  
                           FX_FILESIZE offset,  
                           size_t size) override;  
    FX_FILESIZE GetSize() override;  
  
protected:  
    CPDF_ReadValidator(const RetainPtr<IFX_SeekableReadStream>& file_read,  
                      CPDF_DataAvail::FileAvail* file_avail);  
    ~CPDF_ReadValidator() override;  
  
private:  
    void ScheduleDownload(FX_FILESIZE offset, size_t size);  
    bool IsDataRangeAvailable(FX_FILESIZE offset, size_t size) const;  
  
    RetainPtr<IFX_SeekableReadStream> file_read_;  
    UnownedPtr<CPDF_DataAvail::FileAvail> file_avail_;  
  
    UnownedPtr<CPDF_DataAvail::DownloadHints> hints_;  
  
    bool read_error_;
```

```
    bool has_unavailable_data_;  
    bool whole_file_already_available_;  
    const FX_FILESIZE file_size_;  
};
```

```
#endif // CORE_FPDFAPI_PARSER_CPDF_READ_VALIDATOR_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_REFERENCE_H
#define CORE_FPDFAPI_PARSER_CPDF_REFERENCE_H

#include <memory>
#include <set>

#include "core/fpdfapi/parser/cpdf_object.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_IndirectObjectHolder;

class CPDF_Reference final : public CPDF_Object {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    // CPDF_Object:
    Type GetType() const override;
    RetainPtr<CPDF_Object> Clone() const override;
    CPDF_Object* GetDirect() override;
    const CPDF_Object* GetDirect() const override;
    ByteString GetString() const override;
    float GetNumber() const override;
    int GetInteger() const override;
    CPDF_Dictionary* GetDict() override;
    const CPDF_Dictionary* GetDict() const override;
    bool IsReference() const override;
    CPDF_Reference* AsReference() override;
    const CPDF_Reference* AsReference() const override;
    bool WriteTo(IFX_ArchiveStream* archive,
                const CPDF_Encryptor* encryptor) const override;
    RetainPtr<CPDF_Object> MakeReference(
        CPDF_IndirectObjectHolder* holder) const override;

    CPDF_IndirectObjectHolder* GetObjList() const { return m_pObjList.Get(); }
    uint32_t GetRefObjNum() const { return m_RefObjNum; }
    void SetRef(CPDF_IndirectObjectHolder* pDoc, uint32_t objnum);

private:
    CPDF_Reference(CPDF_IndirectObjectHolder* pDoc, uint32_t objnum);
    ~CPDF_Reference() override;

    RetainPtr<CPDF_Object> CloneNonCyclic(
        bool bDirect,
        std::set<const CPDF_Object*>* pVisited) const override;
    CPDF_Object* SafeGetDirect();
    const CPDF_Object* SafeGetDirect() const;

    UnownedPtr<CPDF_IndirectObjectHolder> m_pObjList;
    uint32_t m_RefObjNum;
};

inline CPDF_Reference* ToReference(CPDF_Object* obj) {
    return obj ? obj->AsReference() : nullptr;
}

inline const CPDF_Reference* ToReference(const CPDF_Object* obj) {
```

```
    return obj ? obj->AsReference() : nullptr;
}
```

```
#endif // CORE_FPDFAPI_PARSER_CPDF_REFERENCE_H_
```

```

// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_SECURITY_HANDLER_H_
#define CORE_FPDFAPI_PARSER_CPDF_SECURITY_HANDLER_H_

#include <memory>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

#define FXCIPHER_NONE 0
#define FXCIPHER_RC4 1
#define FXCIPHER_AES 2
#define FXCIPHER_AES2 3

class CPDF_Array;
class CPDF_CryptoHandler;
class CPDF_Dictionary;
class CPDF_Parser;

class CPDF_SecurityHandler : public Retainable {
public:
  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  bool OnInit(const CPDF_Dictionary* pEncryptDict,
              const CPDF_Array* pIdArray,
              const ByteString& password);
  void OnCreate(CPDF_Dictionary* pEncryptDict,
               const CPDF_Array* pIdArray,
               const ByteString& user_password,
               const ByteString& owner_password);
  void OnCreate(CPDF_Dictionary* pEncryptDict,
               const CPDF_Array* pIdArray,
               const ByteString& user_password);

  uint32_t GetPermissions() const;
  bool IsMetadataEncrypted() const;

  CPDF_CryptoHandler* GetCryptoHandler() const {
    return m_pCryptoHandler.get();
  }

  // Take |password| and encode it, if necessary, based on the password encoding
  // conversion.
  ByteString GetEncodedPassword(ByteStringView password) const;

private:
  enum PasswordEncodingConversion {
    kUnknown,
    kNone,
    kLatin1ToUtf8,
    kUtf8toLatin1,
  };

  CPDF_SecurityHandler();
  ~CPDF_SecurityHandler() override;

```

```
bool LoadDict(const CPDF_Dictionary* pEncryptDict);
bool LoadDict(const CPDF_Dictionary* pEncryptDict,
               int* cipher,
               size_t* key_len);

ByteString GetUserPassword(const ByteString& owner_password) const;
bool CheckPassword(const ByteString& user_password, bool bOwner);
bool CheckPasswordImpl(const ByteString& password, bool bOwner);
bool CheckUserPassword(const ByteString& password, bool bIgnoreEncryptMeta);
bool CheckOwnerPassword(const ByteString& password);
bool AES256_CheckPassword(const ByteString& password, bool bOwner);
void AES256_SetPassword(CPDF_Dictionary* pEncryptDict,
                       const ByteString& password,
                       bool bOwner);
void AES256_SetPerms(CPDF_Dictionary* pEncryptDict);
void OnCreateInternal(CPDF_Dictionary* pEncryptDict,
                     const CPDF_Array* pIdArray,
                     const ByteString& user_password,
                     const ByteString& owner_password,
                     bool bDefault);
bool CheckSecurity(const ByteString& password);

void InitCryptoHandler();

bool m_bOwnerUnlocked = false;
int m_Version = 0;
int m_Revision = 0;
uint32_t m_Permissions = 0;
int m_Cipher = FXCIPHER_NONE;
size_t m_KeyLen = 0;
PasswordEncodingConversion m_PasswordEncodingConversion = kUnknown;
ByteString m_FileId;
RetainPtr<const CPDF_Dictionary> m_pEncryptDict;
std::unique_ptr<CPDF_CryptoHandler> m_pCryptoHandler;
uint8_t m_EncryptKey[32];
};

#endif // CORE_FPDFAPI_PARSER_CPDF_SECURITY_HANDLER_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_SEEKABLEMULTISTREAM_H_
#define CORE_FPDFAPI_PARSER_CPDF_SEEKABLEMULTISTREAM_H_

#include <vector>

#include "core/fxcrt/fx_stream.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Stream;
class CPDF_StreamAcc;

class CPDF_SeekableMultiStream final : public IFX_SeekableStream {
public:
  explicit CPDF_SeekableMultiStream(
    const std::vector<const CPDF_Stream*>& streams);
  ~CPDF_SeekableMultiStream() override;

  // IFX_SeekableReadStream
  FX_FILESIZE GetPosition() override;
  FX_FILESIZE GetSize() override;
  bool ReadBlockAtOffset(void* buffer,
                        FX_FILESIZE offset,
                        size_t size) override;
  size_t ReadBlock(void* buffer, size_t size) override;
  bool IsEOF() override;
  bool Flush() override;
  bool WriteBlockAtOffset(const void* pData,
                        FX_FILESIZE offset,
                        size_t size) override;

private:
  std::vector<RetainPtr<CPDF_StreamAcc>> m_Data;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_SEEKABLEMULTISTREAM_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_SIMPLE_PARSER_H_
#define CORE_FPDFAPI_PARSER_CPDF_SIMPLE_PARSER_H_

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "third_party/base/span.h"

class CPDF_SimpleParser {
public:
  explicit CPDF_SimpleParser(pdfium::span<const uint8_t> input);
  ~CPDF_SimpleParser();

  ByteStringView GetWord();

  void SetCurPos(uint32_t pos) { cur_pos_ = pos; }
  uint32_t GetCurPos() const { return cur_pos_; }

private:
  const pdfium::span<const uint8_t> data_;
  uint32_t cur_pos_ = 0;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_SIMPLE_PARSER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_STREAM_ACC_H_
#define CORE_FPDFAPI_PARSER_CPDF_STREAM_ACC_H_

#include <memory>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/maybe_owned.h"
#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/span.h"

class CPDF_Dictionary;
class CPDF_Stream;

class CPDF_StreamAcc final : public Retainable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    CPDF_StreamAcc(const CPDF_StreamAcc&) = delete;
    CPDF_StreamAcc& operator=(const CPDF_StreamAcc&) = delete;

    void LoadAllDataFiltered();
    void LoadAllDataFilteredWithEstimatedSize(uint32_t estimated_size);
    void LoadAllDataImageAcc(uint32_t estimated_size);
    void LoadAllDataRaw();

    const CPDF_Stream* GetStream() const { return m_pStream.Get(); }
    const CPDF_Dictionary* GetDict() const;

    uint8_t* GetData() const;
    uint32_t GetSize() const;
    pdfium::span<uint8_t> GetSpan();
    pdfium::span<const uint8_t> GetSpan() const;
    ByteString ComputeDigest() const;
    ByteString GetImageDecoder() const { return m_ImageDecoder; }
    const CPDF_Dictionary* GetImageParam() const { return m_pImageParam.Get(); }
    std::unique_ptr<uint8_t, FxFreeDeleter> DetachData();

private:
    explicit CPDF_StreamAcc(const CPDF_Stream* pStream);
    ~CPDF_StreamAcc() override;

    void LoadAllData(bool bRawAccess, uint32_t estimated_size, bool bImageAcc);
    void ProcessRawData();
    void ProcessFilteredData(uint32_t estimated_size, bool bImageAcc);

    // Reads the raw data from |m_pStream|, or return nullptr on failure.
    std::unique_ptr<uint8_t, FxFreeDeleter> ReadRawStream() const;

    MaybeOwned<uint8_t, FxFreeDeleter> m_pData;
    uint32_t m_dwSize = 0;
    ByteString m_ImageDecoder;
    RetainPtr<const CPDF_Dictionary> m_pImageParam;
    RetainPtr<const CPDF_Stream> const m_pStream;
};
```

```
#endif // CORE_FPDFAPI_PARSER_CPDF_STREAM_ACC_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_STREAM_H_
#define CORE_FPDFAPI_PARSER_CPDF_STREAM_H_

#include <memory>
#include <set>
#include <sstream>

#include "core/fpdfapi/parser/cpdf_object.h"
#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_stream.h"

class CPDF_Stream final : public CPDF_Object {
public:
    static const int kFileBufSize = 512;

    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    // CPDF_Object:
    Type GetType() const override;
    RetainPtr<CPDF_Object> Clone() const override;
    CPDF_Dictionary* GetDict() override;
    const CPDF_Dictionary* GetDict() const override;
    WideString GetUnicodeText() const override;
    bool IsStream() const override;
    CPDF_Stream* AsStream() override;
    const CPDF_Stream* AsStream() const override;
    bool WriteTo(IFX_ArchiveStream* archive,
                const CPDF_Encryptor* encryptor) const override;

    uint32_t GetRawSize() const { return m_dwSize; }
    // Will be null in case when stream is not memory based.
    // Use CPDF_StreamAcc to data access in all cases.
    uint8_t* GetInMemoryRawData() const { return m_pDataBuf.get(); }

    // Copies span into internally-owned buffer.
    void SetData(pdfium::span<const uint8_t> pData);

    void TakeData(std::unique_ptr<uint8_t, FxFreeDeleter> pData, uint32_t size);

    void SetDataFromStringstream(std::ostringstream* stream);

    // Set data and remove "Filter" and "DecodeParms" fields from stream
    // dictionary.
    void SetDataAndRemoveFilter(pdfium::span<const uint8_t> pData);
    void SetDataFromStringstreamAndRemoveFilter(std::ostringstream* stream);

    void InitStream(pdfium::span<const uint8_t> pData,
                   RetainPtr<CPDF_Dictionary> pDict);
    void InitStreamFromFile(const RetainPtr<IFX_SeekableReadStream>& pFile,
                           RetainPtr<CPDF_Dictionary> pDict);

    bool ReadRawData(FX_FILESIZE offset, uint8_t* pBuf, uint32_t buf_size) const;

    bool IsMemoryBased() const { return m_bMemoryBased; }
    bool HasFilter() const;
```

```
private:
  CPDF_Stream();
  CPDF_Stream(std::unique_ptr<uint8_t, FxFreeDeleter> pData,
              uint32_t size,
              RetainPtr<CPDF_Dictionary> pDict);
  ~CPDF_Stream() override;

  RetainPtr<CPDF_Object> CloneNonCyclic(
    bool bDirect,
    std::set<const CPDF_Object*>* pVisited) const override;

  bool m_bMemoryBased = true;
  uint32_t m_dwSize = 0;
  RetainPtr<CPDF_Dictionary> m_pDict;
  std::unique_ptr<uint8_t, FxFreeDeleter> m_pDataBuf;
  RetainPtr<IFX_SeekableReadStream> m_pFile;
};

inline CPDF_Stream* ToStream(CPDF_Object* obj) {
  return obj ? obj->AsStream() : nullptr;
}

inline const CPDF_Stream* ToStream(const CPDF_Object* obj) {
  return obj ? obj->AsStream() : nullptr;
}

inline RetainPtr<CPDF_Stream> ToStream(RetainPtr<CPDF_Object> obj) {
  return RetainPtr<CPDF_Stream>(ToStream(obj.Get()));
}

#endif // CORE_FPDFAPI_PARSER_CPDF_STREAM_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_STRING_H_
#define CORE_FPDFAPI_PARSER_CPDF_STRING_H_

#include <memory>

#include "core/fpdfapi/parser/cpdf_object.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/string_pool_template.h"
#include "core/fxcrt/weak_ptr.h"

class CPDF_String final : public CPDF_Object {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    // CPDF_Object:
    Type GetType() const override;
    RetainPtr<CPDF_Object> Clone() const override;
    ByteString GetString() const override;
    WideString GetUnicodeText() const override;
    void SetString(const ByteString& str) override;
    bool IsString() const override;
    CPDF_String* AsString() override;
    const CPDF_String* AsString() const override;
    bool WriteTo(IFX_ArchiveStream* archive,
                const CPDF_Encryptor* encryptor) const override;

    bool IsHex() const { return m_bHex; }

private:
    CPDF_String();
    CPDF_String(WeakPtr<ByteStringPool> pPool, const ByteString& str, bool bHex);
    CPDF_String(WeakPtr<ByteStringPool> pPool, const WideString& str);
    ~CPDF_String() override;

    ByteString m_String;
    bool m_bHex = false;
};

inline CPDF_String* ToString(CPDF_Object* obj) {
    return obj ? obj->AsString() : nullptr;
}

inline const CPDF_String* ToString(const CPDF_Object* obj) {
    return obj ? obj->AsString() : nullptr;
}

#endif // CORE_FPDFAPI_PARSER_CPDF_STRING_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_CPDF_SYNTAX_PARSER_H
#define CORE_FPDFAPI_PARSER_CPDF_SYNTAX_PARSER_H

#include <memory>
#include <vector>

#include "core/fpdfapi/parser/cpdf_stream.h"
#include "core/fxcrt/string_pool_template.h"
#include "core/fxcrt/weak_ptr.h"

class CPDF_CryptoHandler;
class CPDF_Dictionary;
class CPDF_IndirectObjectHolder;
class CPDF_Object;
class CPDF_ReadValidator;
class CPDF_Stream;
class IFX_SeekableReadStream;

class CPDF_SyntaxParser {
public:
    enum class ParseType { kStrict, kLoose };

    static std::unique_ptr<CPDF_SyntaxParser> CreateForTesting(
        const RetainPtr<IFX_SeekableReadStream>& pFileAccess,
        FX_FILESIZE HeaderOffset);

    explicit CPDF_SyntaxParser(
        const RetainPtr<IFX_SeekableReadStream>& pFileAccess);
    CPDF_SyntaxParser(const RetainPtr<CPDF_ReadValidator>& pValidator,
        FX_FILESIZE HeaderOffset);
    ~CPDF_SyntaxParser();

    void SetReadBufferSize(uint32_t read_buffer_size) {
        m_ReadBufferSize = read_buffer_size;
    }

    FX_FILESIZE GetPos() const { return m_Pos; }
    void SetPos(FX_FILESIZE pos);

    RetainPtr<CPDF_Object> GetObjectBody(CPDF_IndirectObjectHolder* pObjList);

    RetainPtr<CPDF_Object> GetIndirectObject(CPDF_IndirectObjectHolder* pObjList,
        ParseType parse_type);

    ByteString GetKeyword();
    void ToNextLine();
    void ToNextWord();
    bool BackwardsSearchToWord(ByteStringView word, FX_FILESIZE limit);
    FX_FILESIZE FindTag(ByteStringView tag);
    bool ReadBlock(uint8_t* pBuf, uint32_t size);
    bool GetCharAt(FX_FILESIZE pos, uint8_t& ch);
    ByteString GetNextWord(bool* bIsNumber);
    ByteString PeekNextWord(bool* bIsNumber);

    const RetainPtr<CPDF_ReadValidator>& GetValidator() const {
        return m_pFileAccess;
    }
}
```

```
uint32_t GetDirectNum();
bool GetNextChar(uint8_t& ch);

// The document size may be smaller than the file size.
// The syntax parser use position relative to document
// offset (|m_HeaderOffset|).
// The document size will be FileSize - "Header offset".
// All offsets was readed from document, should not be great than document
// size. Use it for checks instead of real file size.
FX_FILESIZE GetDocumentSize() const;

ByteString ReadString();
ByteString ReadHexString();

private:
friend class CPDF_DataAvail;
friend class cpdf_syntax_parser_ReadHexString_Test;

static const int kParserMaxRecursionDepth = 64;
static int s_CurrentRecursionDepth;

bool ReadBlockAt(FX_FILESIZE read_pos);
bool GetCharAtBackward(FX_FILESIZE pos, uint8_t* ch);
void GetNextWordInternal(bool* bIsNumber);
bool IsWholeWord(FX_FILESIZE startpos,
                FX_FILESIZE limit,
                ByteStringView tag,
                bool checkKeyword);

unsigned int ReadEOLMarkers(FX_FILESIZE pos);
FX_FILESIZE FindWordPos(ByteStringView word);
FX_FILESIZE FindStreamEndPos();
RetainPtr<CPDF_Stream> ReadStream(RetainPtr<CPDF_Dictionary> pDict);

bool IsPositionRead(FX_FILESIZE pos) const;

RetainPtr<CPDF_Object> GetObjectBodyInternal(
    CPDF_IndirectObjectHolder* pObjList,
    ParseType parse_type);

RetainPtr<CPDF_ReadValidator> m_pFileAccess;
// The syntax parser use position relative to header offset.
// The header contains at file start, and can follow after some stuff. We
// ignore this stuff.
const FX_FILESIZE m_HeaderOffset;
const FX_FILESIZE m_FileLen;
FX_FILESIZE m_Pos = 0;
WeakPtr<ByteStringPool> m_pPool;
std::vector<uint8_t> m_pFileBuf;
FX_FILESIZE m_BufOffset = 0;
uint32_t m_WordSize = 0;
uint8_t m_WordBuffer[257];
uint32_t m_ReadBufferSize = CPDF_Stream::kFileBufSize;
};

#endif // CORE_FPDFAPI_PARSER_CPDF_SYNTAX_PARSER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_FPDF_PARSER_DECODE_H
#define CORE_FPDFAPI_PARSER_FPDF_PARSER_DECODE_H

#include <memory>
#include <utility>
#include <vector>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/optional.h"
#include "third_party/base/span.h"

class CPDF_Array;
class CPDF_Dictionary;
class CPDF_Object;

namespace fxcodec {
class ScanlineDecoder;
}

// Indexed by 8-bit char code, contains unicode code points.
extern const uint16_t PDFDocEncoding[256];

bool ValidateDecoderPipeline(const CPDF_Array* pDecoders);

ByteString PDF_EncodeString(const ByteString& src, bool bHex);
WideString PDF_DecompileText(pdfium::span<const uint8_t> span);
ByteString PDF_EncodeText(const WideString& str);

std::unique_ptr<fxcodec::ScanlineDecoder> CreateFaxDecoder(
    pdfium::span<const uint8_t> src_span,
    int width,
    int height,
    const CPDF_Dictionary* pParams);

std::unique_ptr<fxcodec::ScanlineDecoder> CreateFlateDecoder(
    pdfium::span<const uint8_t> src_span,
    int width,
    int height,
    int nComps,
    int bpc,
    const CPDF_Dictionary* pParams);

bool FlateEncode(pdfium::span<const uint8_t> src_span,
    std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,
    uint32_t* dest_size);

uint32_t FlateDecode(pdfium::span<const uint8_t> src_span,
    std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,
    uint32_t* dest_size);

uint32_t RunLengthDecode(pdfium::span<const uint8_t> src_span,
    std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,
    uint32_t* dest_size);

uint32_t A85Decode(pdfium::span<const uint8_t> src_span,
```

```
    std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,  
    uint32_t* dest_size);
```

```
uint32_t HexDecode(pdfium::span<const uint8_t> src_span,  
    std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,  
    uint32_t* dest_size);
```

```
uint32_t FlateOrLZWDecode(bool bLZW,  
    pdfium::span<const uint8_t> src_span,  
    const CPDF_Dictionary* pParams,  
    uint32_t estimated_size,  
    std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,  
    uint32_t* dest_size);
```

```
Optional<std::vector<std::pair<ByteString, const CPDF_Object*>>>  
GetDecoderArray(const CPDF_Dictionary* pDict);
```

```
bool PDF_DataDecode(  
    pdfium::span<const uint8_t> src_span,  
    uint32_t estimated_size,  
    bool bImageAcc,  
    const std::vector<std::pair<ByteString, const CPDF_Object*>>& decoder_array,  
    std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,  
    uint32_t* dest_size,  
    ByteString* ImageEncoding,  
    RetainPtr<const CPDF_Dictionary>* pImageParams);
```

```
#endif // CORE_FPDFAPI_PARSER_FPDF_PARSER_DECODE_H_
```

```

// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_PARSER_FPDF_PARSER_UTILITY_H_
#define CORE_FPDFAPI_PARSER_FPDF_PARSER_UTILITY_H_

#include <ostream>
#include <vector>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/optional.h"

class CPDF_Array;
class CPDF_Dictionary;
class CPDF_Object;
class IFX_SeekableReadStream;

// Use the accessors below instead of directly accessing PDF_CharType.
extern const char PDF_CharType[256];

inline bool PDFCharIsWhitespace(uint8_t c) {
    return PDF_CharType[c] == 'W';
}
inline bool PDFCharIsNumeric(uint8_t c) {
    return PDF_CharType[c] == 'N';
}
inline bool PDFCharIsDelimiter(uint8_t c) {
    return PDF_CharType[c] == 'D';
}
inline bool PDFCharIsOther(uint8_t c) {
    return PDF_CharType[c] == 'R';
}

inline bool PDFCharIsLineEnding(uint8_t c) {
    return c == '\r' || c == '\n';
}

// On success, return a positive offset value to the PDF header. If the header
// cannot be found, or if there is an error reading from |pFile|, then return
// nullptr.
Optional<FX_FILESIZE> GetHeaderOffset(
    const RetainPtr<IFX_SeekableReadStream>& pFile);

int32_t GetDirectInteger(const CPDF_Dictionary* pDict, const ByteString& key);

ByteString PDF_NameDecode(ByteStringView orig);
ByteString PDF_NameEncode(const ByteString& orig);

// Return |nCount| elements from |pArray| as a vector of floats. |pArray| must
// have at least |nCount| elements.
std::vector<float> ReadArrayElementsToVector(const CPDF_Array* pArray,
                                             size_t nCount);

// Returns true if |dict| has a /Type name entry that matches |type|.
bool ValidateDictType(const CPDF_Dictionary* dict, const ByteString& type);

// Returns true if |dict| is non-null and all entries in |dict| are dictionaries
// of |type|.
bool ValidateDictAllResourcesOfType(const CPDF_Dictionary* dict,

```

```
    const ByteString& type);
```

```
// Shorthand for ValidateDictAllResourcesOfType(dict, "Font").
```

```
bool ValidateFontResourceDict(const CPDF_Dictionary* dict);
```

```
std::ostream& operator<<(std::ostream& buf, const CPDF_Object* pObj);
```

```
#endif // CORE_FPDFAPI_PARSER_FPDF_PARSER_UTILITY_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_CHARPOSLIST_H_
#define CORE_FPDFAPI_RENDER_CPDF_CHARPOSLIST_H_

#include <vector>

#include "core/fxcrt/fx_system.h"

class CPDF_Font;
class TextCharPos;

class CPDF_CharPosList {
public:
  CPDF_CharPosList(const std::vector<uint32_t>& charCodes,
                  const std::vector<float>& charPos,
                  CPDF_Font* pFont,
                  float font_size);
  ~CPDF_CharPosList();

  const std::vector<TextCharPos>& Get() const { return m_CharPos; }

private:
  std::vector<TextCharPos> m_CharPos;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_CHARPOSLIST_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_DEVICEBUFFER_H_
#define CORE_FPDFAPI_RENDER_CPDF_DEVICEBUFFER_H_

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CFX_DIBitmap;
class CFX_RenderDevice;
class CPDF_PageObject;
class CPDF_RenderContext;

class CPDF_DeviceBuffer {
public:
    static CFX_Matrix CalculateMatrix(CFX_RenderDevice* pDevice,
                                      const FX_RECT& rect,
                                      int max_dpi,
                                      bool scale);

    CPDF_DeviceBuffer(CPDF_RenderContext* pContext,
                     CFX_RenderDevice* pDevice,
                     const FX_RECT& rect,
                     const CPDF_PageObject* pObj,
                     int max_dpi);
    ~CPDF_DeviceBuffer();

    bool Initialize();
    void OutputToDevice();
    RetainPtr<CFX_DIBitmap> GetBitmap() const { return m_pBitmap; }
    const CFX_Matrix& GetMatrix() const { return m_Matrix; }

private:
    UnownedPtr<CFX_RenderDevice> const m_pDevice;
    UnownedPtr<CPDF_RenderContext> const m_pContext;
    UnownedPtr<const CPDF_PageObject> const m_pObject;
    RetainPtr<CFX_DIBitmap> const m_pBitmap;
    const FX_RECT m_Rect;
    const CFX_Matrix m_Matrix;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_DEVICEBUFFER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_DOCRENDERDATA_H
#define CORE_FPDFAPI_RENDER_CPDF_DOCRENDERDATA_H

#include <map>

#include "core/fpdfapi/parser/cpdf_document.h"
#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Font;
class CPDF_Object;
class CPDF_TransferFunc;
class CPDF_Type3Cache;
class CPDF_Type3Font;

class CPDF_DocRenderData : public CPDF_Document::RenderDataIface {
public:
    static CPDF_DocRenderData* FromDocument(const CPDF_Document* pDoc);

    CPDF_DocRenderData();
    ~CPDF_DocRenderData() override;

    CPDF_DocRenderData(const CPDF_DocRenderData&) = delete;
    CPDF_DocRenderData& operator=(const CPDF_DocRenderData&) = delete;

    RetainPtr<CPDF_Type3Cache> GetCachedType3(CPDF_Type3Font* pFont);
    RetainPtr<CPDF_TransferFunc> GetTransferFunc(const CPDF_Object* pObj);

protected:
    // protected for use by test subclasses.
    RetainPtr<CPDF_TransferFunc> CreateTransferFunc(
        const CPDF_Object* pObj) const;

private:
    std::map<CPDF_Font*, ObservedPtr<CPDF_Type3Cache>> m_Type3FaceMap;
    std::map<const CPDF_Object*, ObservedPtr<CPDF_TransferFunc>>
        m_TransferFuncMap;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_DOCRENDERDATA_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_IMAGECACHEENTRY_H
#define CORE_FPDFAPI_RENDER_CPDF_IMAGECACHEENTRY_H

#include "core/fpdfapi/page/cpdf_dibase.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Dictionary;
class CPDF_Document;
class CPDF_Image;
class CPDF_RenderStatus;
class PauseIndicatorIface;

class CPDF_ImageCacheEntry {
public:
    CPDF_ImageCacheEntry(CPDF_Document* pDoc,
                        const RetainPtr<CPDF_Image>& pImage);
    ~CPDF_ImageCacheEntry();

    void Reset();
    uint32_t EstimateSize() const { return m_dwCacheSize; }
    uint32_t GetTimeCount() const { return m_dwTimeCount; }
    CPDF_Image* GetImage() const { return m_pImage.Get(); }

    CPDF_DIBBase::LoadState StartGetCachedBitmap(
        const CPDF_Dictionary* pFormResources,
        CPDF_Dictionary* pPageResources,
        bool bStdCS,
        uint32_t GroupFamily,
        bool bLoadMask,
        CPDF_RenderStatus* pRenderStatus);

    // Returns whether to Continue() or not.
    bool Continue(PauseIndicatorIface* pPause, CPDF_RenderStatus* pRenderStatus);

    RetainPtr<CFX_DIBBase> DetachBitmap();
    RetainPtr<CFX_DIBBase> DetachMask();

    int m_dwTimeCount = 0;
    uint32_t m_MatteColor = 0;

private:
    void ContinueGetCachedBitmap(CPDF_RenderStatus* pRenderStatus);
    void CalcSize();

    UnownedPtr<CPDF_Document> const m_pDocument;
    RetainPtr<CPDF_Image> const m_pImage;
    RetainPtr<CFX_DIBBase> m_pCurBitmap;
    RetainPtr<CFX_DIBBase> m_pCurMask;
    RetainPtr<CFX_DIBBase> m_pCachedBitmap;
    RetainPtr<CFX_DIBBase> m_pCachedMask;
    uint32_t m_dwCacheSize = 0;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_IMAGECACHEENTRY_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_IMAGELOADER_H
#define CORE_FPDFAPI_RENDER_CPDF_IMAGELOADER_H

#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CFX_DIBBase;
class CPDF_ImageObject;
class CPDF_PageRenderCache;
class CPDF_RenderStatus;
class CPDF_TransferFunc;
class PauseIndicatorIface;

class CPDF_ImageLoader {
public:
    CPDF_ImageLoader();
    ~CPDF_ImageLoader();

    bool Start(CPDF_ImageObject* pImage,
               CPDF_PageRenderCache* pCache,
               bool bStdCS,
               uint32_t GroupFamily,
               bool bLoadMask,
               CPDF_RenderStatus* pRenderStatus);
    bool Continue(PauseIndicatorIface* pPause, CPDF_RenderStatus* pRenderStatus);

    RetainPtr<CFX_DIBBase> TranslateImage(
        const RetainPtr<CPDF_TransferFunc>& pTransferFunc);

    const RetainPtr<CFX_DIBBase>& GetBitmap() const { return m_pBitmap; }
    const RetainPtr<CFX_DIBBase>& GetMask() const { return m_pMask; }
    uint32_t MatteColor() const { return m_MatteColor; }

private:
    void HandleFailure();

    uint32_t m_MatteColor = 0;
    bool m_bCached = false;
    RetainPtr<CFX_DIBBase> m_pBitmap;
    RetainPtr<CFX_DIBBase> m_pMask;
    UnownedPtr<CPDF_PageRenderCache> m_pCache;
    UnownedPtr<CPDF_ImageObject> m_pImageObject;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_IMAGELOADER_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_IMAGERENDERER_H
#define CORE_FPDFAPI_RENDER_CPDF_IMAGERENDERER_H

#include <memory>

#include "core/fpdfapi/render/cpdf_imageloader.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/dib/cfx_imagerenderer.h"
#include "core/fxge/fx_dib.h"
#include "third_party/base/optional.h"

class CFX_DIBitmap;
class CFX_DIBBase;
class CFX_DefaultRenderDevice;
class CFX_ImageTransformer;
class CPDF_ImageObject;
class CPDF_PageObject;
class CPDF_Pattern;
class CPDF_RenderOptions;
class CPDF_RenderStatus;

class CPDF_ImageRenderer {
public:
    CPDF_ImageRenderer();
    ~CPDF_ImageRenderer();

    bool Start(CPDF_RenderStatus* pStatus,
              CPDF_ImageObject* pImageObject,
              const CFX_Matrix& mtObj2Device,
              bool bStdCS,
              BlendMode blendType);

    bool Start(CPDF_RenderStatus* pStatus,
              const RetainPtr<CFX_DIBBase>& pDIBBase,
              FX_ARGB bitmap_argb,
              int bitmap_alpha,
              const CFX_Matrix& mtImage2Device,
              const FXDIB_ResampleOptions& options,
              bool bStdCS,
              BlendMode blendType);

    bool Continue(PauseIndicatorIface* pPause);
    bool GetResult() const { return m_Result; }

private:
    enum class Mode {
        kNone = 0,
        kDefault,
        kBlend,
        kTransform,
    };

    bool StartBitmapAlpha();
    bool StartDIBBase();
    bool StartRenderDIBBase();
    bool StartLoadDIBBase();
};
```

```
bool ContinueDefault(PauseIndicatorIface* pPause);
bool ContinueBlend(PauseIndicatorIface* pPause);
bool ContinueTransform(PauseIndicatorIface* pPause);
bool DrawMaskedImage();
bool DrawPatternImage();
bool NotDrawing() const;
FX_RECT GetDrawRect() const;
CFX_Matrix GetDrawMatrix(const FX_RECT& rect) const;
void CalculateDrawImage(CFX_DefaultRenderDevice* pBitmapDevice1,
                       CFX_DefaultRenderDevice* pBitmapDevice2,
                       const RetainPtr<CFX_DIBBase>& pDIBBase,
                       const CFX_Matrix& mtNewMatrix,
                       const FX_RECT& rect) const;
const CPDF_RenderOptions& GetRenderOptions() const;
void HandleFilters();
Optional<FX_RECT> GetUnitRect() const;
bool GetDimensionsFromUnitRect(const FX_RECT& rect,
                               int* left,
                               int* top,
                               int* width,
                               int* height) const;

UnownedPtr<CPDF_RenderStatus> m_pRenderStatus;
UnownedPtr<CPDF_ImageObject> m_pImageObject;
RetainPtr<CPDF_Pattern> m_pPattern;
RetainPtr<CFX_DIBBase> m_pDIBBase;
CFX_Matrix m_mtObj2Device;
CFX_Matrix m_ImageMatrix;
CPDF_ImageLoader m_Loader;
std::unique_ptr<CFX_ImageTransformer> m_pTransformer;
std::unique_ptr<CFX_ImageRenderer> m_DeviceHandle;
Mode m_Mode = Mode::kNone;
int m_BitmapAlpha = 0;
BlendMode m_BlendType = BlendMode::kNormal;
FX_ARGB m_FillArgb = 0;
FXDIB_ResampleOptions m_ResampleOptions;
bool m_bPatternColor = false;
bool m_bStdCS = false;
bool m_Result = true;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_IMAGERENDERER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_PAGERENDERCACHE_H
#define CORE_FPDFAPI_RENDER_CPDF_PAGERENDERCACHE_H

#include <map>
#include <memory>

#include "core/fpdfapi/page/cpdf_page.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/maybe_owned.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Image;
class CPDF_ImageCacheEntry;
class CPDF_Page;
class CPDF_RenderStatus;
class CPDF_Stream;
class PauseIndicatorIface;

class CPDF_PageRenderCache : public CPDF_Page::RenderCacheIface {
public:
    explicit CPDF_PageRenderCache(CPDF_Page* pPage);
    ~CPDF_PageRenderCache() override;

    // CPDF_Page::RenderCacheIface:
    void ResetBitmapForImage(const RetainPtr<CPDF_Image>& pImage) override;

    void CacheOptimization(int32_t dwLimitCacheSize);
    uint32_t GetTimeCount() const { return m_nTimeCount; }
    CPDF_Page* GetPage() const { return m_pPage.Get(); }
    CPDF_ImageCacheEntry* GetCurImageCacheEntry() const {
        return m_pCurImageCacheEntry.Get();
    }

    bool StartGetCachedBitmap(const RetainPtr<CPDF_Image>& pImage,
                             bool bStdCS,
                             uint32_t GroupFamily,
                             bool bLoadMask,
                             CPDF_RenderStatus* pRenderStatus);

    bool Continue(PauseIndicatorIface* pPause, CPDF_RenderStatus* pRenderStatus);

private:
    void ClearImageCacheEntry(CPDF_Stream* pStream);

    UnownedPtr<CPDF_Page> const m_pPage;
    std::map<CPDF_Stream*, std::unique_ptr<CPDF_ImageCacheEntry>> m_ImageCache;
    MaybeOwned<CPDF_ImageCacheEntry> m_pCurImageCacheEntry;
    uint32_t m_nTimeCount = 0;
    uint32_t m_nCacheSize = 0;
    bool m_bCurFindCache = false;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_PAGERENDERCACHE_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_PAGERENDERCONTEXT_H_
#define CORE_FPDFAPI_RENDER_CPDF_PAGERENDERCONTEXT_H_

#include <memory>

#include "core/fpdfapi/page/cpdf_page.h"

class CFX_RenderDevice;
class CPDF_ProgressiveRenderer;
class CPDF_RenderContext;
class CPDF_RenderOptions;

// Everything about rendering is put here: for OOM recovery
class CPDF_PageRenderContext final : public CPDF_Page::RenderContextIface {
public:
  // Context merely manages the lifetime for callers.
  class AnnotListIface {
public:
  virtual ~AnnotListIface() {}
};

  CPDF_PageRenderContext();
  ~CPDF_PageRenderContext() override;

  // Specific destruction order required.
  std::unique_ptr<AnnotListIface> m_pAnnots;
  std::unique_ptr<CPDF_RenderOptions> m_pOptions;
  std::unique_ptr<CFX_RenderDevice> m_pDevice;
  std::unique_ptr<CPDF_RenderContext> m_pContext;
  std::unique_ptr<CPDF_ProgressiveRenderer> m_pRenderer;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_PAGERENDERCONTEXT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_RENDER_CPDF_PROGRESSIVERENDERER_H  
#define CORE_FPDFAPI_RENDER_CPDF_PROGRESSIVERENDERER_H
```

```
#include <memory>
```

```
#include "core/fpdfapi/page/cpdf_pageobjectholder.h"  
#include "core/fpdfapi/render/cpdf_rendercontext.h"  
#include "core/fxcrt/fx_coordinates.h"  
#include "core/fxcrt/fx_system.h"
```

```
class CPDF_RenderOptions;  
class CPDF_RenderStatus;  
class CFX_RenderDevice;  
class PauseIndicatorIface;
```

```
class CPDF_ProgressiveRenderer {
```

```
public:
```

```
    // Must match FDF_RENDER_* definitions in public/fpdf_progressive.h, but  
    // cannot #include that header. fpdfsdk/fpdf_progressive.cpp has  
    // static_asserts to make sure the two sets of values match.
```

```
    enum Status {
```

```
        Ready,                // FPDF_RENDER_READY  
        ToBeContinued,       // FPDF_RENDER_TOBECONTINUED  
        Done,                 // FPDF_RENDER_DONE  
        Failed                // FPDF_RENDER_FAILED
```

```
};
```

```
static int ToFPDFStatus(Status status) { return static_cast<int>(status); }
```

```
CPDF_ProgressiveRenderer(CPDF_RenderContext* pContext,  
                          CFX_RenderDevice* pDevice,  
                          const CPDF_RenderOptions* pOptions);  
~CPDF_ProgressiveRenderer();
```

```
Status GetStatus() const { return m_Status; }
```

```
void Start(PauseIndicatorIface* pPause);
```

```
void Continue(PauseIndicatorIface* pPause);
```

```
private:
```

```
    // Maximum page objects to render before checking for pause.
```

```
    static const int kStepLimit = 100;
```

```
Status m_Status;  
UnownedPtr<CPDF_RenderContext> const m_pContext;  
UnownedPtr<CFX_RenderDevice> const m_pDevice;  
const CPDF_RenderOptions* const m_pOptions;  
std::unique_ptr<CPDF_RenderStatus> m_pRenderStatus;  
CFX_FloatRect m_ClipRect;  
uint32_t m_LayerIndex;  
CPDF_RenderContext::Layer* m_pCurrentLayer;  
CPDF_PageObjectHolder::const_iterator m_LastObjectRendered;
```

```
};
```

```
#endif // CORE_FPDFAPI_RENDER_CPDF_PROGRESSIVERENDERER_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_RENDERCONTEXT_H_
#define CORE_FPDFAPI_RENDER_CPDF_RENDERCONTEXT_H_

#include <vector>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Dictionary;
class CPDF_Document;
class CPDF_Page;
class CPDF_PageObject;
class CPDF_PageObjectHolder;
class CPDF_PageRenderCache;
class CPDF_RenderOptions;
class CFX_DIBitmap;
class CFX_Matrix;
class CFX_RenderDevice;

class CPDF_RenderContext {
public:
  class Layer {
  public:
    Layer();
    Layer(const Layer& that);
    ~Layer();

    UnownedPtr<CPDF_PageObjectHolder> m_pObjectHolder;
    CFX_Matrix m_Matrix;
  };

  explicit CPDF_RenderContext(CPDF_Page* pPage);
  CPDF_RenderContext(CPDF_Document* pDoc, CPDF_PageRenderCache* pPageCache);
  ~CPDF_RenderContext();

  void AppendLayer(CPDF_PageObjectHolder* pObjectHolder,
                  const CFX_Matrix* pObject2Device);

  void Render(CFX_RenderDevice* pDevice,
              const CPDF_RenderOptions* pOptions,
              const CFX_Matrix* pLastMatrix);

  void Render(CFX_RenderDevice* pDevice,
              const CPDF_PageObject* pStopObj,
              const CPDF_RenderOptions* pOptions,
              const CFX_Matrix* pLastMatrix);

  void GetBackground(const RetainPtr<CFX_DIBitmap>& pBuffer,
                    const CPDF_PageObject* pObj,
                    const CPDF_RenderOptions* pOptions,
                    const CFX_Matrix& mtFinal);

  size_t CountLayers() const { return m_Layers.size(); }
  Layer* GetLayer(uint32_t index) { return &m_Layers[index]; }

  CPDF_Document* GetDocument() const { return m_pDocument.Get(); }
};
```

```
CPDF_Dictionary* GetPageResources() const { return m_pPageResources.Get(); }  
CPDF_PageRenderCache* GetPageCache() const { return m_pPageCache.Get(); }
```

**protected:**

```
UnownedPtr<CPDF_Document> const m_pDocument;  
RetainPtr<CPDF_Dictionary> m_pPageResources;  
UnownedPtr<CPDF_PageRenderCache> m_pPageCache;  
std::vector<Layer> m_Layers;  
};
```

```
#endif // CORE_FPDFAPI_RENDER_CPDF_RENDERCONTEXT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_RENDER_CPDF_RENDEROPTIONS_H  
#define CORE_FPDFAPI_RENDER_CPDF_RENDEROPTIONS_H
```

```
#include "core/fpdfapi/page/cpdf_occontext.h"  
#include "core/fxcrt/fx_system.h"  
#include "core/fxcrt/retain_ptr.h"  
#include "core/fxge/fx_dib.h"
```

```
class CPDF_RenderOptions {  
public:  
    enum Type : uint8_t { kNormal = 0, kGray, kAlpha };  
  
    struct Options {  
        Options();  
        Options(const Options& rhs);  
  
        bool bClearType = false;  
        bool bPrintGraphicText = false;  
        bool bForceDownsample = false;  
        bool bPrintPreview = false;  
        bool bBGRStripe = false;  
        bool bNoNativeText = false;  
        bool bForceHalftone = false;  
        bool bRectAA = false;  
        bool bFillFullcover = false;  
        bool bPrintImageText = false;  
        bool bOverprint = false;  
        bool bThinLine = false;  
        bool bBreakForMasks = false;  
        bool bNoTextSmooth = false;  
        bool bNoPathSmooth = false;  
        bool bNoImageSmooth = false;  
        bool bLimitedImageCache = false;  
    };  
  
    CPDF_RenderOptions();  
    CPDF_RenderOptions(const CPDF_RenderOptions& rhs);  
    ~CPDF_RenderOptions();  
  
    FX_ARGB TranslateColor(FX_ARGB argb) const;  
  
    void SetColorMode(Type mode) { m_ColorMode = mode; }  
    bool ColorModeIs(Type mode) const { return m_ColorMode == mode; }  
  
    const Options& GetOptions() const { return m_Options; }  
    Options& GetOptions() { return m_Options; }  
  
    uint32_t GetCacheSizeLimit() const;  
  
    void SetDrawAnnots(bool draw) { m_bDrawAnnots = draw; }  
    bool GetDrawAnnots() const { return m_bDrawAnnots; }  
  
    void SetOCContext(RetainPtr<CPDF_OCContext> context) {  
        m_pOCContext = context;  
    }  
    const CPDF_OCContext* GetOCContext() const { return m_pOCContext.Get(); }  
};
```

```
private:
  Type m_ColorMode = kNormal;
  bool m_bDrawAnnots = false;
  Options m_Options;
  RetainPtr<CPDF_OCContext> m_pOCContext;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_RENDEROPTIONS_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_RENDERSHADING_H_
#define CORE_FPDFAPI_RENDER_CPDF_RENDERSHADING_H_

class CFX_Matrix;
class CFX_RenderDevice;
class CPDF_PageObject;
class CPDF_RenderContext;
class CPDF_RenderOptions;
class CPDF_ShadingPattern;
struct FX_RECT;

class CPDF_RenderShading {
public:
    static void Draw(CFX_RenderDevice* pDevice,
                    CPDF_RenderContext* pContext,
                    const CPDF_PageObject* pCurObj,
                    const CPDF_ShadingPattern* pPattern,
                    const CFX_Matrix& mtMatrix,
                    const FX_RECT& clip_rect,
                    int alpha,
                    const CPDF_RenderOptions& options);

    CPDF_RenderShading() = delete;
    CPDF_RenderShading(const CPDF_RenderShading&) = delete;
    CPDF_RenderShading& operator=(const CPDF_RenderShading&) = delete;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_RENDERSHADING_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_RENDERSTATUS_H_
#define CORE_FPDFAPI_RENDER_CPDF_RENDERSTATUS_H_

#include <memory>
#include <vector>

#include "core/fpdfapi/page/cpdf_clippath.h"
#include "core/fpdfapi/page/cpdf_graphicstates.h"
#include "core/fpdfapi/page/cpdf_transparency.h"
#include "core/fpdfapi/parser/cpdf_dictionary.h"
#include "core/fpdfapi/render/cpdf_renderoptions.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/fx_dib.h"

class CFX_DIBitmap;
class CFX_PathData;
class CFX_RenderDevice;
class CPDF_Color;
class CPDF_Font;
class CPDF_FormObject;
class CPDF_ImageCacheEntry;
class CPDF_ImageObject;
class CPDF_ImageRenderer;
class CPDF_Object;
class CPDF_PageObject;
class CPDF_PageObjectHolder;
class CPDF_PathObject;
class CPDF_RenderContext;
class CPDF_ShadingObject;
class CPDF_ShadingPattern;
class CPDF_TilingPattern;
class CPDF_TransferFunc;
class CPDF_Type3Cache;
class CPDF_Type3Char;
class CPDF_Type3Font;
class PauseIndicatorIface;

class CPDF_RenderStatus {
public:
  CPDF_RenderStatus(CPDF_RenderContext* pContext, CFX_RenderDevice* pDevice);
  ~CPDF_RenderStatus();

  // Called prior to Initialize().
  void SetOptions(const CPDF_RenderOptions& options) { m_Options = options; }
  void SetDeviceMatrix(const CFX_Matrix& matrix) { m_DeviceMatrix = matrix; }
  void SetStopObject(const CPDF_PageObject* pStopObj) { m_pStopObj = pStopObj; }
  void SetFormResource(const CPDF_Dictionary* pRes) {
    m_pFormResource.Reset(pRes);
  }
  void SetType3Char(CPDF_Type3Char* pType3Char) { m_pType3Char = pType3Char; }
  void SetFillColor(FX_ARGB color) { m_T3FillColor = color; }
  void SetDropObjects(bool bDropObjects) { m_bDropObjects = bDropObjects; }
  void SetLoadMask(bool bLoadMask) { m_bLoadMask = bLoadMask; }
  void SetStdCS(bool bStdCS) { m_bStdCS = bStdCS; }
  void SetGroupFamily(uint32_t family) { m_GroupFamily = family; }
  void SetTransparency(const CPDF_Transparency& transparency) {
```

```
    m_Transparency = transparency;
}

void Initialize(const CPDF_RenderStatus* pParentStatus,
               const CPDF_GraphicStates* pInitialStates);

void RenderObjectList(const CPDF_PageObjectHolder* pObjectHolder,
                     const CFX_Matrix& mtObj2Device);
void RenderSingleObject(CPDF_PageObject* pObj,
                       const CFX_Matrix& mtObj2Device);
bool ContinueSingleObject(CPDF_PageObject* pObj,
                          const CFX_Matrix& mtObj2Device,
                          PauseIndicatorIface* pPause);
void ProcessClipPath(const CPDF_ClipPath& ClipPath,
                    const CFX_Matrix& mtObj2Device);

uint32_t GetGroupFamily() const { return m_GroupFamily; }
bool GetLoadMask() const { return m_bLoadMask; }
bool GetDropObjects() const { return m_bDropObjects; }
bool IsPrint() const { return m_bPrint; }
bool IsStopped() const { return m_bStopped; }
CPDF_RenderContext* GetContext() const { return m_pContext.Get(); }
const CPDF_Dictionary* GetFormResource() const {
    return m_pFormResource.Get();
}
CPDF_Dictionary* GetPageResource() const { return m_pPageResource.Get(); }
CFX_RenderDevice* GetRenderDevice() const { return m_pDevice; }
const CPDF_RenderOptions& GetRenderOptions() const { return m_Options; }

#ifdef _SKIA_SUPPORT_
    void DebugVerifyDeviceIsPreMultiplied() const;
#endif

RetainPtr<CPDF_TransferFunc> GetTransferFunc(
    const CPDF_Object* pObject) const;

FX_ARGB GetFillArgb(CPDF_PageObject* pObj) const {
    return GetFillArgbInternal(pObj, false);
}
FX_ARGB GetFillArgbForType3(CPDF_PageObject* pObj) const {
    return GetFillArgbInternal(pObj, true);
}

void DrawTilingPattern(CPDF_TilingPattern* pPattern,
                     CPDF_PageObject* pPageObj,
                     const CFX_Matrix& mtObj2Device,
                     bool bStroke);
void DrawShadingPattern(CPDF_ShadingPattern* pPattern,
                       const CPDF_PageObject* pPageObj,
                       const CFX_Matrix& mtObj2Device,
                       bool bStroke);
void CompositeDIBitmap(const RetainPtr<CFX_DIBitmap>& pDIBitmap,
                      int left,
                      int top,
                      FX_ARGB mask_argb,
                      int bitmap_alpha,
                      BlendMode blend_mode,
                      const CPDF_Transparency& transparency);

private:
    static std::unique_ptr<CPDF_GraphicStates> CloneObjStates(
        const CPDF_GraphicStates* pSrcStates,
        bool bStroke);
```

```

FX_ARGB GetFillArgbInternal(CPDF_PageObject* pObj, bool bType3) const;
bool ProcessTransparency(CPDF_PageObject* PageObj,
    const CFX_Matrix& mtObj2Device);
void ProcessObjectNoClip(CPDF_PageObject* pObj,
    const CFX_Matrix& mtObj2Device);
void DrawObjWithBackground(CPDF_PageObject* pObj,
    const CFX_Matrix& mtObj2Device);
bool DrawObjWithBlend(CPDF_PageObject* pObj, const CFX_Matrix& mtObj2Device);
bool ProcessPath(CPDF_PathObject* pPathObj, const CFX_Matrix& mtObj2Device);
void ProcessPathPattern(CPDF_PathObject* pPathObj,
    const CFX_Matrix& mtObj2Device,
    int* filltype,
    bool* bStroke);
void DrawPathWithPattern(CPDF_PathObject* pPathObj,
    const CFX_Matrix& mtObj2Device,
    const CPDF_Color* pColor,
    bool bStroke);
bool ClipPattern(const CPDF_PageObject* pPageObj,
    const CFX_Matrix& mtObj2Device,
    bool bStroke);
bool SelectClipPath(const CPDF_PathObject* pPathObj,
    const CFX_Matrix& mtObj2Device,
    bool bStroke);
bool ProcessImage(CPDF_ImageObject* pImageObj,
    const CFX_Matrix& mtObj2Device);
void ProcessShading(const CPDF_ShadingObject* pShadingObj,
    const CFX_Matrix& mtObj2Device);
bool ProcessType3Text(CPDF_TextObject* textobj,
    const CFX_Matrix& mtObj2Device);
bool ProcessText(CPDF_TextObject* textobj,
    const CFX_Matrix& mtObj2Device,
    CFX_PathData* pClippingPath);
void DrawTextPathWithPattern(const CPDF_TextObject* textobj,
    const CFX_Matrix& mtObj2Device,
    CPDF_Font* pFont,
    float font_size,
    const CFX_Matrix* pTextMatrix,
    bool bFill,
    bool bStroke);
bool ProcessForm(const CPDF_FormObject* pFormObj,
    const CFX_Matrix& mtObj2Device);
RetainPtr<CFX_DIBitmap> GetBackdrop(const CPDF_PageObject* pObj,
    const FX_RECT& rect,
    bool bBackAlphaRequired,
    int* left,
    int* top);
RetainPtr<CFX_DIBitmap> LoadSMask(CPDF_Dictionary* pSMaskDict,
    FX_RECT* pClipRect,
    const CFX_Matrix* pMatrix);
// Optionally write the colorspace family value into |pCSFamily|.
FX_ARGB GetBackColor(const CPDF_Dictionary* pSMaskDict,
    const CPDF_Dictionary* pGroupDict,
    int* pCSFamily);
FX_ARGB GetStrokeArgb(CPDF_PageObject* pObj) const;
FX_RECT GetObjectClippedRect(const CPDF_PageObject* pObj,
    const CFX_Matrix& mtObj2Device) const;

CPDF_RenderOptions m_Options;
RetainPtr<const CPDF_Dictionary> m_pFormResource;
RetainPtr<CPDF_Dictionary> m_pPageResource;
std::vector<CPDF_Type3Font*> m_Type3FontCache;
UnownedPtr<CPDF_RenderContext> const m_pContext;

```

```
bool m_bStopped = false;
CFX_RenderDevice* const m_pDevice;
CFX_Matrix m_DeviceMatrix;
CPDF_ClipPath m_LastClipPath;
UnownedPtr<const CPDF_PageObject> m_pCurObj;
UnownedPtr<const CPDF_PageObject> m_pStopObj;
CPDF_GraphicStates m_InitialStates;
std::unique_ptr<CPDF_ImageRenderer> m_pImageRenderer;
CPDF_Transparency m_Transparency;
bool m_bPrint = false;
bool m_bDropObjects = false;
bool m_bStdCS = false;
bool m_bLoadMask = false;
uint32_t m_GroupFamily = 0;
UnownedPtr<CPDF_Type3Char> m_pType3Char;
FX_ARGB m_T3FillColor = 0;
BlendMode m_curBlend = BlendMode::kNormal;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_RENDERSTATUS_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_SCALEDRENDERBUFFER_H_
#define CORE_FPDFAPI_RENDER_CPDF_SCALEDRENDERBUFFER_H_

#include <memory>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/unowned_ptr.h"

class CFX_DefaultRenderDevice;
class CFX_RenderDevice;
class CPDF_PageObject;
class CPDF_RenderContext;
class CPDF_RenderOptions;

class CPDF_ScaledRenderBuffer {
public:
  CPDF_ScaledRenderBuffer();
  ~CPDF_ScaledRenderBuffer();

  bool Initialize(CPDF_RenderContext* pContext,
                 CFX_RenderDevice* pDevice,
                 const FX_RECT& rect,
                 const CPDF_PageObject* pObj,
                 const CPDF_RenderOptions* pOptions,
                 int max_dpi);

  CFX_RenderDevice* GetDevice() const;
  const CFX_Matrix& GetMatrix() const { return m_Matrix; }
  void OutputToDevice();

private:
  UnownedPtr<CFX_RenderDevice> m_pDevice;
  UnownedPtr<CPDF_RenderContext> m_pContext;
  FX_RECT m_Rect;
  UnownedPtr<const CPDF_PageObject> m_pObject;
  std::unique_ptr<CFX_DefaultRenderDevice> m_pBitmapDevice;
  CFX_Matrix m_Matrix;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_SCALEDRENDERBUFFER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFAPI_RENDER_CPDF_TEXTRENDERER_H
```

```
#define CORE_FPDFAPI_RENDER_CPDF_TEXTRENDERER_H
```

```
#include <vector>
```

```
#include "core/fxcrt/fx_coordinates.h"
```

```
#include "core/fxcrt/fx_string.h"
```

```
#include "core/fxcrt/fx_system.h"
```

```
#include "core/fxge/fx_dib.h"
```

```
class CFX_RenderDevice;
```

```
class CFX_GraphStateData;
```

```
class CFX_PathData;
```

```
class CPDF_RenderOptions;
```

```
class CPDF_Font;
```

```
class CPDF_TextRenderer {
```

```
  public:
```

```
    static void DrawTextString(CFX_RenderDevice* pDevice,  
                             float origin_x,  
                             float origin_y,  
                             CPDF_Font* pFont,  
                             float font_size,  
                             const CFX_Matrix& matrix,  
                             const ByteString& str,  
                             FX_ARGB fill_argb,  
                             const CPDF_RenderOptions& options);
```

```
    static bool DrawTextPath(CFX_RenderDevice* pDevice,  
                             const std::vector<uint32_t>& charCodes,  
                             const std::vector<float>& charPos,  
                             CPDF_Font* pFont,  
                             float font_size,  
                             const CFX_Matrix& mtText2User,  
                             const CFX_Matrix* pUser2Device,  
                             const CFX_GraphStateData* pGraphState,  
                             FX_ARGB fill_argb,  
                             FX_ARGB stroke_argb,  
                             CFX_PathData* pClippingPath,  
                             int nFlag);
```

```
    static bool DrawNormalText(CFX_RenderDevice* pDevice,  
                               const std::vector<uint32_t>& charCodes,  
                               const std::vector<float>& charPos,  
                               CPDF_Font* pFont,  
                               float font_size,  
                               const CFX_Matrix& mtText2Device,  
                               FX_ARGB fill_argb,  
                               const CPDF_RenderOptions& options);
```

```
  CPDF_TextRenderer() = delete;
```

```
  CPDF_TextRenderer(const CPDF_TextRenderer&) = delete;
```

```
  CPDF_TextRenderer& operator=(const CPDF_TextRenderer&) = delete;
```

```
};
```

```
#endif // CORE_FPDFAPI_RENDER_CPDF_TEXTRENDERER_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_TYPE3CACHE_H_
#define CORE_FPDFAPI_RENDER_CPDF_TYPE3CACHE_H_

#include <map>
#include <memory>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"

class CFX_GlyphBitmap;
class CFX_Matrix;
class CPDF_Type3Font;
class CPDF_Type3GlyphMap;

class CPDF_Type3Cache final : public Retainable, public Observable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    const CFX_GlyphBitmap* LoadGlyph(uint32_t charcode,
                                       const CFX_Matrix* pMatrix);

private:
    explicit CPDF_Type3Cache(CPDF_Type3Font* pFont);
    ~CPDF_Type3Cache() override;

    std::unique_ptr<CFX_GlyphBitmap> RenderGlyph(CPDF_Type3GlyphMap* pSize,
                                               uint32_t charcode,
                                               const CFX_Matrix* pMatrix);

    RetainPtr<CPDF_Type3Font> const m_pFont;
    std::map<ByteString, std::unique_ptr<CPDF_Type3GlyphMap>> m_SizeMap;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_TYPE3CACHE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_TYPE3GLYPHMAP_H_
#define CORE_FPDFAPI_RENDER_CPDF_TYPE3GLYPHMAP_H_

#include <map>
#include <memory>
#include <utility>
#include <vector>

#include "core/fxcrt/fx_system.h"

class CFX_GlyphBitmap;

class CPDF_Type3GlyphMap {
public:
  CPDF_Type3GlyphMap();
  ~CPDF_Type3GlyphMap();

  // Returns a pair of integers (top_line, bottom_line).
  std::pair<int, int> AdjustBlue(float top, float bottom);

  const CFX_GlyphBitmap* GetBitmap(uint32_t charcode) const;
  void SetBitmap(uint32_t charcode, std::unique_ptr<CFX_GlyphBitmap> pMap);

private:
  std::vector<int> m_TopBlue;
  std::vector<int> m_BottomBlue;
  std::map<uint32_t, std::unique_ptr<CFX_GlyphBitmap>> m_GlyphMap;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_TYPE3GLYPHMAP_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFAPI_RENDER_CPDF_WINDOWSRENDERDEVICE_H_
#define CORE_FPDFAPI_RENDER_CPDF_WINDOWSRENDERDEVICE_H_

#include "core/fxge/cfx_windowsrenderdevice.h"

class CPDF_WindowsRenderDevice final : public CFX_WindowsRenderDevice {
public:
  explicit CPDF_WindowsRenderDevice(HDC hDC);
  ~CPDF_WindowsRenderDevice() override;
};

#endif // CORE_FPDFAPI_RENDER_CPDF_WINDOWSRENDERDEVICE_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CBA_FONTMAP_H_
#define CORE_FPDFDOC_CBA_FONTMAP_H_

#include <memory>
#include <vector>

#include "core/fpdfdoc/ipvt_fontmap.h"
#include "core/fxcrt/fx_codepage.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Dictionary;
class CPDF_Document;

class CBA_FontMap final : public IPVT_FontMap {
public:
    static int32_t GetNativeCharset();

    CBA_FontMap(CPDF_Document* pDocument, CPDF_Dictionary* pAnnotDict);
    ~CBA_FontMap() override;

    // IPVT_FontMap
    RetainPtr<CPDF_Font> GetPDFFont(int32_t nFontIndex) override;
    ByteString GetPDFFontAlias(int32_t nFontIndex) override;
    int32_t GetWordFontIndex(uint16_t word,
                             int32_t nCharset,
                             int32_t nFontIndex) override;
    int32_t CharCodeFromUnicode(int32_t nFontIndex, uint16_t word) override;
    int32_t CharSetFromUnicode(uint16_t word, int32_t nOldCharset) override;

    void Reset();
    void SetAPTType(const ByteString& sAPType);
    void SetDefaultFont(const RetainPtr<CPDF_Font>& pFont,
                       const ByteString& sFontName);

private:
    struct Data {
        Data();
        ~Data();

        RetainPtr<CPDF_Font> pFont;
        int32_t nCharset;
        ByteString sFontName;
    };

    struct Native {
        int32_t nCharset;
        ByteString sFontName;
    };

    void Initialize();
    RetainPtr<CPDF_Font> FindFontSameCharset(ByteString* sFontAlias,
                                             int32_t nCharset);
    RetainPtr<CPDF_Font> FindResFontSameCharset(const CPDF_Dictionary* pResDict,
                                               ByteString* sFontAlias,
                                               int32_t nCharset);
    RetainPtr<CPDF_Font> GetAnnotDefaultFont(ByteString* sAlias);
};
```

```
void AddFontToAnnotDict(const RetainPtr<CPDF_Font>& pFont,
                       const ByteString& sAlias);

bool KnowWord(int32_t nFontIndex, uint16_t word);

void Clear();
int32_t GetFontIndex(const ByteString& sFontName,
                    int32_t nCharset,
                    bool bFind);
int32_t AddFontData(const RetainPtr<CPDF_Font>& pFont,
                   const ByteString& sFontAlias,
                   int32_t nCharset);

ByteString EncodeFontAlias(const ByteString& sFontName, int32_t nCharset);
ByteString EncodeFontAlias(const ByteString& sFontName);

int32_t FindFont(const ByteString& sFontName, int32_t nCharset);
ByteString GetNativeFontName(int32_t nCharset);
ByteString GetCachedNativeFontName(int32_t nCharset);
bool IsStandardFont(const ByteString& sFontName);
RetainPtr<CPDF_Font> AddFontToDocument(CPDF_Document* pDoc,
                                     ByteString& sFontName,
                                     uint8_t nCharset);
RetainPtr<CPDF_Font> AddStandardFont(CPDF_Document* pDoc,
                                    ByteString& sFontName);
RetainPtr<CPDF_Font> AddSystemFont(CPDF_Document* pDoc,
                                   ByteString& sFontName,
                                   uint8_t nCharset);

std::vector<std::unique_ptr<Data>> m_Data;
std::vector<std::unique_ptr<Native>> m_NativeFont;
UnownedPtr<CPDF_Document> const m_pDocument;
RetainPtr<CPDF_Dictionary> const m_pAnnotDict;
RetainPtr<CPDF_Font> m_pDefaultFont;
ByteString m_sDefaultFontName;
ByteString m_sAPType = "N";
};

#endif // CORE_FPDFDOC_CBA_FONTMAP_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPFDDOC_CLINE_H_
#define CORE_FPFDDOC_CLINE_H_

#include "core/fpdfdoc/cpvt_lineinfo.h"
#include "core/fpdfdoc/cpvt_wordplace.h"

class CLine {
public:
  CLine();
  explicit CLine(const CPVT_LineInfo& lineinfo);
  ~CLine();

  CPVT_WordPlace GetBeginWordPlace() const;
  CPVT_WordPlace GetEndWordPlace() const;
  CPVT_WordPlace GetPrevWordPlace(const CPVT_WordPlace& place) const;
  CPVT_WordPlace GetNextWordPlace(const CPVT_WordPlace& place) const;
  CPVT_WordPlace LinePlace;
  CPVT_LineInfo m_LineInfo;
};

#endif // CORE_FPFDDOC_CLINE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_AACTION_H_
#define CORE_FPDFDOC_CPDF_AACTION_H_

#include "core/fpdfdoc/cpdf_action.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Dictionary;

class CPDF_AAction {
public:
  enum AActionType {
    kCursorEnter = 0,
    kCursorExit,
    kButtonDown,
    kButtonUp,
    kGetFocus,
    kLoseFocus,
    kPageOpen,
    kPageClose,
    kPageVisible,
    kPageInvisible,
    kOpenPage,
    kClosePage,
    kKeyStroke,
    kFormat,
    kValidate,
    kCalculate,
    kCloseDocument,
    kSaveDocument,
    kDocumentSaved,
    kPrintDocument,
    kDocumentPrinted,
    kDocumentOpen,
    kNumberOfActions // Must be last.
  };

  explicit CPDF_AAction(const CPDF_Dictionary* pDict);
  CPDF_AAction(const CPDF_AAction& that);
  ~CPDF_AAction();

  bool ActionExist(AActionType eType) const;
  CPDF_Action GetAction(AActionType eType) const;
  const CPDF_Dictionary* GetDict() const { return m_pDict.Get(); }

  static bool IsUserClick(AActionType eType);

private:
  RetainPtr<const CPDF_Dictionary> const m_pDict;
};

#endif // CORE_FPDFDOC_CPDF_AACTION_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_ACTIONFIELDS_H_
#define CORE_FPDFDOC_CPDF_ACTIONFIELDS_H_

#include <stddef.h>

#include <vector>

#include "core/fxcrt/unowned_ptr.h"

class CPDF_Action;
class CPDF_Object;

class CPDF_ActionFields {
public:
  explicit CPDF_ActionFields(const CPDF_Action* pAction);
  ~CPDF_ActionFields();

  std::vector<const CPDF_Object*> GetAllFields() const;
  const CPDF_Object* GetField(size_t iIndex) const;

private:
  UnownedPtr<const CPDF_Action> const m_pAction;
};

#endif // CORE_FPDFDOC_CPDF_ACTIONFIELDS_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_ACTION_H
#define CORE_FPDFDOC_CPDF_ACTION_H

#include "core/fpdfdoc/cpdf_dest.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/optional.h"

class CPDF_Dictionary;
class CPDF_Document;
class CPDF_Object;

class CPDF_Action {
public:
  enum ActionType {
    Unknown = 0,
    GoTo,
    GoToR,
    GoToE,
    Launch,
    Thread,
    URI,
    Sound,
    Movie,
    Hide,
    Named,
    SubmitForm,
    ResetForm,
    ImportData,
    JavaScript,
    SetOCGState,
    Rendition,
    Trans,
    GoTo3DView
  };

  explicit CPDF_Action(const CPDF_Dictionary* pDict);
  CPDF_Action(const CPDF_Action& that);
  ~CPDF_Action();

  const CPDF_Dictionary* GetDict() const { return m_pDict.Get(); }

  ActionType GetType() const;
  CPDF_Dest GetDest(CPDF_Document* pDoc) const;
  WideString GetFilePath() const;
  ByteString GetURI(const CPDF_Document* pDoc) const;
  bool GetHideStatus() const;
  ByteString GetNamedAction() const;
  uint32_t GetFlags() const;

  // Differentiates between empty JS entry and no JS entry.
  Optional<WideString> MaybeGetJavaScript() const;
  // Returns empty string for empty JS entry and no JS entry.
  WideString GetJavaScript() const;

  size_t GetSubActionsCount() const;
  CPDF_Action GetSubAction(size_t iIndex) const;
};
```

```
private:
  const CPDF_Object* GetJavaScriptObject() const;

  RetainPtr<const CPDF_Dictionary> const m_pDict;
};

#endif // CORE_FPDFDOC_CPDF_ACTION_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_ANNOT_H_
#define CORE_FPDFDOC_CPDF_ANNOT_H_

#include <map>
#include <memory>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/maybe_owned.h"

class CFX_RenderDevice;
class CPDF_Array;
class CPDF_Dictionary;
class CPDF_Document;
class CPDF_Form;
class CPDF_Page;
class CPDF_RenderContext;
class CPDF_RenderOptions;
class CPDF_Stream;

class CPDF_Annot {
public:
    enum AppearanceMode { Normal, Rollover, Down };
    enum class Subtype {
        UNKNOWN = 0,
        TEXT,
        LINK,
        FREETEXT,
        LINE,
        SQUARE,
        CIRCLE,
        POLYGON,
        POLYLINE,
        HIGHLIGHT,
        UNDERLINE,
        SQUIGGLY,
        STRIKEOUT,
        STAMP,
        CARET,
        INK,
        POPUP,
        FILEATTACHMENT,
        SOUND,
        MOVIE,
        WIDGET,
        SCREEN,
        PRINTERMARK,
        TRAPNET,
        WATERMARK,
        THREED,
        RICHMEDIA,
        XFAWIDGET
    };
};

static CPDF_Annot::Subtype StringToAnnotSubtype(const ByteString& sSubtype);
static ByteString AnnotSubtypeToString(CPDF_Annot::Subtype nSubtype);
```

```
static CFX_FloatRect RectFromQuadPointsArray(const CPDF_Array* pArray,
                                             size_t nIndex);
static CFX_FloatRect BoundingRectFromQuadPoints(
    const CPDF_Dictionary* pAnnotDict);
static CFX_FloatRect RectFromQuadPoints(const CPDF_Dictionary* pAnnotDict,
                                       size_t nIndex);
static size_t QuadPointCount(const CPDF_Array* pArray);

// The second constructor does not take ownership of the dictionary.
CPDF_Annot(RetainPtr<CPDF_Dictionary> pDict, CPDF_Document* pDocument);
CPDF_Annot(CPDF_Dictionary* pDict, CPDF_Document* pDocument);
~CPDF_Annot();

CPDF_Annot::Subtype GetSubtype() const;
uint32_t GetFlags() const;
CFX_FloatRect GetRect() const;
CPDF_Document* GetDocument() const { return m_pDocument.Get(); }
const CPDF_Dictionary* GetAnnotDict() const { return m_pAnnotDict.Get(); }
CPDF_Dictionary* GetAnnotDict() { return m_pAnnotDict.Get(); }

bool IsHidden() const;

bool DrawAppearance(CPDF_Page* pPage,
                   CFX_RenderDevice* pDevice,
                   const CFX_Matrix& mtUser2Device,
                   AppearanceMode mode,
                   const CPDF_RenderOptions* pOptions);
bool DrawInContext(const CPDF_Page* pPage,
                  CPDF_RenderContext* pContext,
                  const CFX_Matrix* pUser2Device,
                  AppearanceMode mode);

void ClearCachedAP();
void DrawBorder(CFX_RenderDevice* pDevice,
               const CFX_Matrix* pUser2Device,
               const CPDF_RenderOptions* pOptions);
CPDF_Form* GetAPForm(const CPDF_Page* pPage, AppearanceMode mode);
void SetOpenState(bool bOpenState) { m_bOpenState = bOpenState; }
CPDF_Annot* GetPopupAnnot() const { return m_pPopupAnnot.Get(); }
void SetPopupAnnot(CPDF_Annot* pAnnot) { m_pPopupAnnot = pAnnot; }

private:
void Init();
void GenerateAPIfNeeded();
bool ShouldGenerateAP() const;
bool ShouldDrawAnnotation() const;

CFX_FloatRect RectForDrawing() const;

RetainPtr<CPDF_Dictionary> const m_pAnnotDict;
UnownedPtr<CPDF_Document> const m_pDocument;
CPDF_Annot::Subtype m_nSubtype;
std::map<CPDF_Stream*, std::unique_ptr<CPDF_Form>> m_APMap;
// If non-null, then this is not a popup annotation.
UnownedPtr<CPDF_Annot> m_pPopupAnnot;
// |m_bOpenState| is only set for popup annotations.
bool m_bOpenState = false;
bool m_bHasGeneratedAP;
bool m_bIsTextMarkupAnnotation;
};

// Get the AP in an annotation dict for a given appearance mode.
// If |eMode| is not Normal and there is not AP for that mode, falls back to
```

```
// the Normal AP.
```

```
CPDF_Stream* GetAnnotAP(CPDF_Dictionary* pAnnotDict,  
                        CPDF_Annot::AppearanceMode eMode);
```

```
// Get the AP in an annotation dict for a given appearance mode.
```

```
// No fallbacks to Normal like in GetAnnotAP.
```

```
CPDF_Stream* GetAnnotAPNoFallback(CPDF_Dictionary* pAnnotDict,  
                                   CPDF_Annot::AppearanceMode eMode);
```

```
#endif // CORE_FPFDDOC_CPDF_ANNOT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_ANNOTLIST_H_
#define CORE_FPDFDOC_CPDF_ANNOTLIST_H_

#include <memory>
#include <vector>

#include "core/fpdfapi/render/cpdf_pagerendercontext.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"

class CFX_RenderDevice;
class CPDF_Annot;
class CPDF_Document;
class CPDF_Page;
class CPDF_RenderContext;
class CPDF_RenderOptions;

class CPDF_AnnotList : public CPDF_PageRenderContext::AnnotListIface {
public:
    explicit CPDF_AnnotList(CPDF_Page* pPage);
    ~CPDF_AnnotList() override;

    void DisplayAnnots(CPDF_Page* pPage,
                      CPDF_RenderContext* pContext,
                      bool bPrinting,
                      const CFX_Matrix* pMatrix,
                      bool bShowWidget,
                      CPDF_RenderOptions* pOptions);

    void DisplayAnnots(CPDF_Page* pPage,
                      CFX_RenderDevice* pDevice,
                      CPDF_RenderContext* pContext,
                      bool bPrinting,
                      const CFX_Matrix* pUser2Device,
                      uint32_t dwAnnotFlags,
                      CPDF_RenderOptions* pOptions,
                      FX_RECT* pClipRect);

    size_t Count() const { return m_AnnotList.size(); }
    CPDF_Annot* GetAt(size_t index) const { return m_AnnotList[index].get(); }
    const std::vector<std::unique_ptr<CPDF_Annot>>& All() const {
        return m_AnnotList;
    }

private:
    void DisplayPass(CPDF_Page* pPage,
                   CFX_RenderDevice* pDevice,
                   CPDF_RenderContext* pContext,
                   bool bPrinting,
                   const CFX_Matrix* pMatrix,
                   bool bWidget,
                   CPDF_RenderOptions* pOptions,
                   FX_RECT* clip_rect);

    UnownedPtr<CPDF_Document> const m_pDocument;
};
```

```
// The first |m_nAnnotCount| elements are from the PDF itself. The rest are
// generated pop-up annotations.
std::vector<std::unique_ptr<CPDF_Annot>> m_AnnotList;
size_t m_nAnnotCount = 0;
};

#endif // CORE_FPDFDOC_CPDF_ANNOTLIST_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_APSETTINGS_H_
#define CORE_FPDFDOC_CPDF_APSETTINGS_H_

#include "core/fpdfdoc/cpdf_iconfit.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/fx_dib.h"

class CPDF_Dictionary;
class CPDF_FormControl;
class CPDF_Stream;

// Corresponds to PDF spec section 12.5.6.19 (Widget annotation TP dictionary).
#define TEXTPOS_CAPTION 0
#define TEXTPOS_ICON 1
#define TEXTPOS_BELOW 2
#define TEXTPOS_ABOVE 3
#define TEXTPOS_RIGHT 4
#define TEXTPOS_LEFT 5
#define TEXTPOS_OVERLAID 6

class CPDF_ApSettings {
public:
  explicit CPDF_ApSettings(CPDF_Dictionary* pDict);
  CPDF_ApSettings(const CPDF_ApSettings& that);
  ~CPDF_ApSettings();

  bool HasMKEntry(const ByteString& csEntry) const;
  int GetRotation() const;

  FX_ARGB GetBorderColor(int& iColorType) const {
    return GetColor(iColorType, "BC");
  }

  float GetOriginalBorderColor(int index) const {
    return GetOriginalColor(index, "BC");
  }

  void GetOriginalBorderColor(int& iColorType, float fc[4]) const {
    GetOriginalColor(iColorType, fc, "BC");
  }

  FX_ARGB GetBackgroundColor(int& iColorType) const {
    return GetColor(iColorType, "BG");
  }

  float GetOriginalBackgroundColor(int index) const {
    return GetOriginalColor(index, "BG");
  }

  void GetOriginalBackgroundColor(int& iColorType, float fc[4]) const {
    GetOriginalColor(iColorType, fc, "BG");
  }

  WideString GetNormalCaption() const { return GetCaption("CA"); }
  WideString GetRolloverCaption() const { return GetCaption("RC"); }
};
```

```
WideString GetDownCaption() const { return GetCaption("AC"); }
CPDF_Stream* GetNormalIcon() const { return GetIcon("I"); }
CPDF_Stream* GetRolloverIcon() const { return GetIcon("RI"); }
CPDF_Stream* GetDownIcon() const { return GetIcon("IX"); }
CPDF_IconFit GetIconFit() const;

// Returns one of the TEXTPOS_* values above.
int GetTextPosition() const;

FX_ARGB GetColor(int& iColorType, const ByteString& csEntry) const;
float GetOriginalColor(int index, const ByteString& csEntry) const;
void GetOriginalColor(int& iColorType,
                     float fc[4],
                     const ByteString& csEntry) const;

WideString GetCaption(const ByteString& csEntry) const;
CPDF_Stream* GetIcon(const ByteString& csEntry) const;

private:
  RetainPtr<CPDF_Dictionary> const m_pDict;
};

#endif // CORE_FPDFDOC_CPDF_APSETTINGS_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_BOOKMARK_H_
#define CORE_FPDFDOC_CPDF_BOOKMARK_H_

#include "core/fpdfdoc/cpdf_action.h"
#include "core/fpdfdoc/cpdf_dest.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Dictionary;
class CPDF_Document;

class CPDF_Bookmark {
public:
  CPDF_Bookmark();
  CPDF_Bookmark(const CPDF_Bookmark& that);
  explicit CPDF_Bookmark(const CPDF_Dictionary* pDict);
  ~CPDF_Bookmark();

  const CPDF_Dictionary* GetDict() const { return m_pDict.Get(); }

  WideString GetTitle() const;
  CPDF_Dest GetDest(CPDF_Document* pDocument) const;
  CPDF_Action GetAction() const;

private:
  RetainPtr<const CPDF_Dictionary> m_pDict;
};

#endif // CORE_FPDFDOC_CPDF_BOOKMARK_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_BOOKMARKTREE_H_
#define CORE_FPDFDOC_CPDF_BOOKMARKTREE_H_

#include "core/fpdfdoc/cpdf_bookmark.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Document;

class CPDF_BookmarkTree {
public:
  explicit CPDF_BookmarkTree(CPDF_Document* pDoc);
  ~CPDF_BookmarkTree();

  CPDF_Bookmark GetFirstChild(CPDF_Bookmark* parent) const;
  CPDF_Bookmark GetNextSibling(CPDF_Bookmark* bookmark) const;
  CPDF_Document* GetDocument() const { return m_pDocument.Get(); }

private:
  UnownedPtr<CPDF_Document> const m_pDocument;
};

#endif // CORE_FPDFDOC_CPDF_BOOKMARKTREE_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_COLOR_UTILS_H_
#define CORE_FPDFDOC_CPDF_COLOR_UTILS_H_

#include "core/fxge/cfx_color.h"

class CPDF_Array;

namespace fxcrtd {
class ByteString;
}

namespace fpdfdoc {

CFX_Color CFXColorFromArray(const CPDF_Array& array);
CFX_Color CFXColorFromString(const fxcrtd::ByteString& str);

} // namespace fpdfdoc

#endif // CORE_FPDFDOC_CPDF_COLOR_UTILS_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_DEFAULTAPPEARANCE_H_
#define CORE_FPDFDOC_CPDF_DEFAULTAPPEARANCE_H_

#include <utility>

#include "core/fpdfapi/parser/cpdf_simple_parser.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxge/cfx_color.h"
#include "core/fxge/fx_dib.h"

class CPDF_DefaultAppearance {
public:
  CPDF_DefaultAppearance() {}
  explicit CPDF_DefaultAppearance(const ByteString& csDA) : m_csDA(csDA) {}
  CPDF_DefaultAppearance(const CPDF_DefaultAppearance& cDA)
    : m_csDA(cDA.m_csDA) {}

  Optional<ByteString> GetFont(float* fFontSize);

  Optional<CFX_Color::Type> GetColor(float fc[4]);
  std::pair<Optional<CFX_Color::Type>, FX_ARGB> GetColor();

  bool FindTagParamFromStartForTesting(CPDF_SimpleParser* parser,
                                       ByteStringView token,
                                       int nParams);

private:
  ByteString m_csDA;
};

#endif // CORE_FPDFDOC_CPDF_DEFAULTAPPEARANCE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_DEST_H_
#define CORE_FPDFDOC_CPDF_DEST_H_

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Document;
class CPDF_Array;

class CPDF_Dest {
public:
  CPDF_Dest();
  explicit CPDF_Dest(const CPDF_Array* pArray);
  CPDF_Dest(const CPDF_Dest& that);
  ~CPDF_Dest();

  const CPDF_Array* GetArray() const { return m_pArray.Get(); }

  ByteString GetRemoteName() const;
  int GetDestPageIndex(CPDF_Document* pDoc) const;

  // Returns the zoom mode, as one of the PDFDEST_VIEW_* values in fpdf_doc.h.
  int GetZoomMode() const;

  unsigned long GetNumParams() const;
  float GetParam(int index) const;

  bool GetXYZ(bool* pHasX,
              bool* pHasY,
              bool* pHasZoom,
              float* pX,
              float* pY,
              float* pZoom) const;

private:
  RetainPtr<const CPDF_Array> const m_pArray;
};

#endif // CORE_FPDFDOC_CPDF_DEST_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_DOCJSACTIONS_H_
#define CORE_FPDFDOC_CPDF_DOCJSACTIONS_H_

#include "core/fpdfdoc/cpdf_action.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Document;

class CPDF_DocJSActions {
public:
  explicit CPDF_DocJSActions(CPDF_Document* pDoc);
  ~CPDF_DocJSActions();

  int CountJSActions() const;
  CPDF_Action GetJSActionAndName(int index, WideString* csName) const;
  CPDF_Document* GetDocument() const { return m_pDocument.Get(); }

private:
  UnownedPtr<CPDF_Document> const m_pDocument;
};

#endif // CORE_FPDFDOC_CPDF_DOCJSACTIONS_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_FILESPEC_H_
#define CORE_FPDFDOC_CPDF_FILESPEC_H_

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/string_pool_template.h"
#include "core/fxcrt/weak_ptr.h"

class CPDF_Dictionary;
class CPDF_Object;
class CPDF_Stream;

class CPDF_FileSpec {
public:
  explicit CPDF_FileSpec(const CPDF_Object* pObj);
  explicit CPDF_FileSpec(CPDF_Object* pObj);
  ~CPDF_FileSpec();

  // Convert a platform dependent file name into pdf format.
  static WideString EncodeFileName(const WideString& filepath);

  // Convert a pdf file name into platform dependent format.
  static WideString DecodeFileName(const WideString& filepath);

  const CPDF_Object* GetObj() const { return m_pObj.Get(); }
  CPDF_Object* GetObj() { return m_pWritableObj.Get(); }
  WideString GetFileName() const;
  const CPDF_Stream* GetFileStream() const;
  CPDF_Stream* GetFileStream();
  const CPDF_Dictionary* GetParamsDict() const;
  CPDF_Dictionary* GetParamsDict();

  // Set this file spec to refer to a file name (not a url).
  void SetFileName(const WideString& wsFileName);

private:
  RetainPtr<const CPDF_Object> const m_pObj;
  RetainPtr<CPDF_Object> const m_pWritableObj;
};

#endif // CORE_FPDFDOC_CPDF_FILESPEC_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.  
  
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFDOC_CPDF_FORMCONTROL_H_  
#define CORE_FPDFDOC_CPDF_FORMCONTROL_H_  
  
#include "core/fpdfdoc/cpdf_aaction.h"  
#include "core/fpdfdoc/cpdf_action.h"  
#include "core/fpdfdoc/cpdf_annot.h"  
#include "core/fpdfdoc/cpdf_annotlist.h"  
#include "core/fpdfdoc/cpdf_apsettings.h"  
#include "core/fpdfdoc/cpdf_defaultappearance.h"  
#include "core/fpdfdoc/cpdf_formfield.h"  
#include "core/fpdfdoc/cpdf_iconfit.h"  
#include "core/fxcrt/fx_coordinates.h"  
#include "core/fxcrt/fx_string.h"  
#include "core/fxcrt/retain_ptr.h"  
#include "core/fxge/fx_dib.h"  
#include "third_party/base/optional.h"  
  
class CFX_RenderDevice;  
class CPDF_Dictionary;  
class CPDF_Font;  
class CPDF_FormField;  
class CPDF_InteractiveForm;  
class CPDF_OCContext;  
class CPDF_RenderOptions;  
class CPDF_Stream;  
  
class CPDF_FormControl {  
public:  
    enum HighlightingMode { None = 0, Invert, Outline, Push, Toggle };  
  
    CPDF_FormControl(CPDF_FormField* pField, CPDF_Dictionary* pWidgetDict);  
    ~CPDF_FormControl();  
  
    CPDF_FormField::Type GetType() const { return m_pField->GetType(); }  
    const CPDF_InteractiveForm* GetInteractiveForm() const {  
        return m_pForm.Get();  
    }  
    CPDF_FormField* GetField() const { return m_pField.Get(); }  
    CPDF_Dictionary* GetWidget() const { return m_pWidgetDict.Get(); }  
    CFX_FloatRect GetRect() const;  
  
    ByteString GetCheckedAPState() const;  
    WideString GetExportValue() const;  
  
    bool IsChecked() const;  
    bool IsDefaultChecked() const;  
  
    HighlightingMode GetHighlightingMode() const;  
    bool HasMKEntry(const ByteString& csEntry) const;  
    int GetRotation() const;  
  
    FX_ARGB GetBorderColor(int& iColorType) { return GetColor(iColorType, "BC"); }  
  
    float GetOriginalBorderColor(int index) {  
        return GetOriginalColor(index, "BC");  
    }  
}
```

```
void GetOriginalBorderColor(int& iColorType, float fc[4]) {
    GetOriginalColor(iColorType, fc, "BC");
}

FX_ARGB GetBackgroundColor(int& iColorType) {
    return GetColor(iColorType, "BG");
}

float GetOriginalBackgroundColor(int index) {
    return GetOriginalColor(index, "BG");
}

void GetOriginalBackgroundColor(int& iColorType, float fc[4]) {
    GetOriginalColor(iColorType, fc, "BG");
}

WideString GetNormalCaption() const { return GetCaption("CA"); }
WideString GetRolloverCaption() const { return GetCaption("RC"); }
WideString GetDownCaption() const { return GetCaption("AC"); }

CPDF_Stream* GetNormalIcon() { return GetIcon("I"); }
CPDF_Stream* GetRolloverIcon() { return GetIcon("RI"); }
CPDF_Stream* GetDownIcon() { return GetIcon("IX"); }
CPDF_IconFit GetIconFit() const;

int GetTextPosition() const;
CPDF_Action GetAction() const;
CPDF_AAction GetAdditionalAction() const;
CPDF_DefaultAppearance GetDefaultAppearance() const;

Optional<WideString> GetDefaultControlFontName() const;
int GetControlAlignment() const;

ByteString GetOnStateName() const;
void CheckControl(bool bChecked);

private:
RetainPtr<CPDF_Font> GetDefaultControlFont() const;
FX_ARGB GetColor(int& iColorType, const ByteString& csEntry);
float GetOriginalColor(int index, const ByteString& csEntry);
void GetOriginalColor(int& iColorType,
                    float fc[4],
                    const ByteString& csEntry);

WideString GetCaption(const ByteString& csEntry) const;
CPDF_Stream* GetIcon(const ByteString& csEntry);
CPDF_ApSettings GetMK() const;

UnownedPtr<CPDF_FormField> const m_pField;
RetainPtr<CPDF_Dictionary> const m_pWidgetDict;
UnownedPtr<const CPDF_InteractiveForm> const m_pForm;
};

#endif // CORE_FPDFDOC_CPDF_FORMCONTROL_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_FORMFIELD_H_
#define CORE_FPDFDOC_CPDF_FORMFIELD_H_

#include <memory>
#include <utility>
#include <vector>

#include "core/fpdfdoc/cpdf_aaction.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Dictionary;
class CPDF_Font;
class CPDF_FormControl;
class CPDF_InteractiveForm;
class CPDF_Object;
class CPDF_String;

enum class NotificationOption { kDoNotNotify = 0, kNotify };

enum class FormFieldType : uint8_t {
    kUnknown = 0,
    kPushButton = 1,
    kCheckBox = 2,
    kRadioButton = 3,
    kComboBox = 4,
    kListBox = 5,
    kTextField = 6,
    kSignature = 7,
#ifdef PDF_ENABLE_XFA
    kXFA = 8, // Generic XFA field, should use value below if possible.
    kXFA_CheckBox = 9,
    kXFA_ComboBox = 10,
    kXFA_ImageField = 11,
    kXFA_ListBox = 12,
    kXFA_PushButton = 13,
    kXFA_Signature = 14,
    kXFA_TextField = 15
#endif
};

Optional<FormFieldType> IntToFormFieldType(int value);

// If values are added to FormFieldType, these will need to be updated.
#ifdef PDF_ENABLE_XFA
constexpr size_t kFormFieldTypeCount = 16;
#else // PDF_ENABLE_XFA
constexpr size_t kFormFieldTypeCount = 8;
#endif // PDF_ENABLE_XFA

const CPDF_Object* FPDF_GetFieldAttr(const CPDF_Dictionary* pFieldDict,
                                     const char* name);
CPDF_Object* FPDF_GetFieldAttr(CPDF_Dictionary* pFieldDict, const char* name);

WideString FPDF_GetFullName(CPDF_Dictionary* pFieldDict);
```

```
class CPDF_FormField {
public:
    enum Type {
        kUnknown,
        kPushButton,
        kRadioButton,
        kCheckBox,
        kText,
        kRichText,
        kFile,
        kListBox,
        kComboBox,
        kSign
    };

    CPDF_FormField(CPDF_InteractiveForm* pForm, CPDF_Dictionary* pDict);
    ~CPDF_FormField();

    WideString GetFullName() const;

    Type GetType() const { return m_Type; }

    CPDF_Dictionary* GetFieldDict() const { return m_pDict.Get(); }

    bool ResetField(NotificationOption notify);

    int CountControls() const;

    CPDF_FormControl* GetControl(int index) const;

    int GetControlIndex(const CPDF_FormControl* pControl) const;
    FormFieldType GetFieldType() const;

    CPDF_AAAction GetAdditionalAction() const;
    WideString GetAlternateName() const;
    WideString GetMappingName() const;

    uint32_t GetFieldFlags() const;
    ByteString GetDefaultStyle() const;

    bool IsRequired() const { return m_bRequired; }
    bool IsNoExport() const { return m_bNoExport; }

    WideString GetValue() const;
    WideString GetDefaultValue() const;
    bool SetValue(const WideString& value, NotificationOption notify);

    int GetMaxLen() const;
    int CountSelectedItems() const;
    int GetSelectedIndex(int index) const;

    bool ClearSelection(NotificationOption notify);
    bool IsItemSelected(int index) const;
    bool SetItemSelection(int index, bool bSelected, NotificationOption notify);

    bool IsItemDefaultSelected(int index) const;

    int GetDefaultSelectedItem() const;
    int CountOptions() const;

    WideString GetOptionLabel(int index) const;
    WideString GetOptionValue(int index) const;
```

```
int FindOption(const WideString& csOptValue) const;

bool CheckControl(int iControlIndex,
                  bool bChecked,
                  NotificationOption notify);

int GetTopVisibleIndex() const;
int CountSelectedOptions() const;

int GetSelectedOptionIndex(int index) const;
bool IsOptionSelected(int iOptIndex) const;
bool SelectOption(int iOptIndex, bool bSelected, NotificationOption notify);

float GetFontSize() const { return m_FontSize; }
CPDF_Font* GetFont() const { return m_pFont.Get(); }

CPDF_Dictionary* GetDict() const { return m_pDict.Get(); }
CPDF_InteractiveForm* GetForm() const { return m_pForm.Get(); }

WideString GetCheckValue(bool bDefault) const;

void SetOpt(RetainPtr<CPDF_Object> pOpt);

private:
WideString GetValue(bool bDefault) const;
bool SetValue(const WideString& value,
              bool bDefault,
              NotificationOption notify);
void InitFieldFlags();
int FindListSel(CPDF_String* str);
WideString GetOptionText(int index, int sub_index) const;
void LoadDA();
bool SetCheckValue(const WideString& value,
                   bool bDefault,
                   NotificationOption notify);
void SetItemSelectionSelected(int index, const WideString& opt_value);
void SetItemSelectionUnselected(int index, const WideString& opt_value);
bool NotifyBeforeSelectionChange(const WideString& value);
void NotifyAfterSelectionChange();
bool NotifyBeforeValueChange(const WideString& value);
void NotifyAfterValueChange();
bool NotifyListOrComboBoxBeforeChange(const WideString& value);
void NotifyListOrComboBoxAfterChange();

const CPDF_Object* GetDefaultValueObject() const;
const CPDF_Object* GetValueObject() const;

// For choice fields.
const CPDF_Object* GetSelectedIndicesObject() const;

// For choice fields.
// Value object takes precedence over selected indices object.
const CPDF_Object* GetValueOrSelectedIndicesObject() const;

const std::vector<UnownedPtr<CPDF_FormControl>>& GetControls() const;

CPDF_FormField::Type m_Type = kUnknown;
bool m_bRequired = false;
bool m_bNoExport = false;
bool m_bIsMultiSelectListBox = false;
bool m_bIsUnison = false;
float m_FontSize = 0;
UnownedPtr<CPDF_InteractiveForm> const m_pForm;
```

```
    RetainPtr<CPDF_Dictionary> const m_pDict;  
    RetainPtr<CPDF_Font> m_pFont;  
};  
  
#endif // CORE_FPDFDOC_CPDF_FORMFIELD_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_ICONFIT_H_
#define CORE_FPDFDOC_CPDF_ICONFIT_H_

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Dictionary;

class CPDF_IconFit {
public:
    enum ScaleMethod { Always = 0, Bigger, Smaller, Never };

    explicit CPDF_IconFit(const CPDF_Dictionary* pDict);
    CPDF_IconFit(const CPDF_IconFit& that);
    ~CPDF_IconFit();

    ScaleMethod GetScaleMethod() const;
    bool IsProportionalScale() const;
    bool GetFittingBounds() const;
    CFX_PointF GetIconBottomLeftPosition() const;
    CFX_PointF GetIconPosition() const;

private:
    RetainPtr<const CPDF_Dictionary> const m_pDict;
};

#endif // CORE_FPDFDOC_CPDF_ICONFIT_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_ICON_H_
#define CORE_FPDFDOC_CPDF_ICON_H_

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Stream;

class CPDF_Icon final {
public:
  CPDF_Icon(CPDF_Stream* pStream);
  ~CPDF_Icon();

  CFX_SizeF GetImageSize() const;
  CFX_Matrix GetImageMatrix() const;
  ByteString GetImageAlias() const;

private:
  RetainPtr<CPDF_Stream> const m_pStream;
};

#endif // CORE_FPDFDOC_CPDF_ICON_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FPDFDOC_CPDF_INTERACTIVEFORM_H_
#define CORE_FPDFDOC_CPDF_INTERACTIVEFORM_H_
```

```
#include <map>
#include <memory>
#include <vector>
```

```
#include "core/fpdfapi/parser/fpdf_parser_decode.h"
#include "core/fpdfdoc/cpdf_defaultappearance.h"
#include "core/fpdfdoc/cpdf_formfield.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
```

```
class CFieldTree;
class CFDF_Document;
class CPDF_Document;
class CPDF_Dictionary;
class CPDF_Font;
class CPDF_FormControl;
class CPDF_Object;
class CPDF_Page;
```

```
RetainPtr<CPDF_Font> AddNativeInteractiveFormFont(CPDF_Dictionary*& pFormDict,
                                                CPDF_Document* pDocument,
                                                ByteString* csNameTag);
```

```
class CPDF_InteractiveForm {
public:
  class NotifierIface {
public:
    virtual ~NotifierIface() = default;

    virtual bool BeforeValueChange(CPDF_FormField* pField,
                                   const WideString& csValue) = 0;
    virtual void AfterValueChange(CPDF_FormField* pField) = 0;
    virtual bool BeforeSelectionChange(CPDF_FormField* pField,
                                       const WideString& csValue) = 0;
    virtual void AfterSelectionChange(CPDF_FormField* pField) = 0;
    virtual void AfterCheckedStatusChange(CPDF_FormField* pField) = 0;
    virtual void AfterFormReset(CPDF_InteractiveForm* pForm) = 0;
  };
};
```

```
explicit CPDF_InteractiveForm(CPDF_Document* pDocument);
~CPDF_InteractiveForm();
```

```
static void SetUpdateAP(bool bUpdateAP);
static bool IsUpdateAPEnabled();
static uint8_t GetNativeCharSet();
static ByteString GetNativeFontName(uint8_t iCharSet, void* pLogFont);
static RetainPtr<CPDF_Font> AddStandardFont(CPDF_Document* pDocument,
                                           ByteString csFontName);
static RetainPtr<CPDF_Font> AddNativeFont(uint8_t iCharSet,
                                           CPDF_Document* pDocument);
static RetainPtr<CPDF_Font> AddNativeFont(CPDF_Document* pDocument);
```

```

size_t CountFields(const WideString& csFieldName) const;
CPDF_FormField* GetField(uint32_t index, const WideString& csFieldName) const;
CPDF_FormField* GetFieldByDict(CPDF_Dictionary* pFieldDict) const;

CPDF_FormControl* GetControlAtPoint(CPDF_Page* pPage,
                                     const CFX_PointF& point,
                                     int* z_order) const;
CPDF_FormControl* GetControlByDict(const CPDF_Dictionary* pWidgetDict) const;

bool NeedConstructAP() const;
int CountFieldsInCalculationOrder();
CPDF_FormField* GetFieldInCalculationOrder(int index);
int FindFieldInCalculationOrder(const CPDF_FormField* pField);

RetainPtr<CPDF_Font> GetFormFont(ByteString csNameTag) const;
CPDF_DefaultAppearance GetDefaultAppearance() const;
int GetFormAlignment() const;

bool CheckRequiredFields(const std::vector<CPDF_FormField*>* fields,
                        bool bIncludeOrExclude) const;

std::unique_ptr<CFDF_Document> ExportToFDF(const WideString& pdf_path,
                                           bool bSimpleFileSpec) const;

std::unique_ptr<CFDF_Document> ExportToFDF(
    const WideString& pdf_path,
    const std::vector<CPDF_FormField*>& fields,
    bool bIncludeOrExclude,
    bool bSimpleFileSpec) const;

void ResetForm(NotificationOption notify);

// TODO(tsepez): Use a span.
void ResetForm(const std::vector<CPDF_FormField*>& fields,
              bool bIncludeOrExclude,
              NotificationOption notify);

void SetNotifierIface(NotifierIface* pNotify);
bool HasXFAForm() const;
void FixPageFields(CPDF_Page* pPage);

NotifierIface* GetFormNotify() const { return m_pFormNotify.Get(); }
CPDF_Document* GetDocument() const { return m_pDocument.Get(); }
CPDF_Dictionary* GetFormDict() const { return m_pFormDict.Get(); }

const std::vector<UnownedPtr<CPDF_FormControl>>& GetControlsForField(
    const CPDF_FormField* pField);

private:
void LoadField(CPDF_Dictionary* pFieldDict, int nLevel);
void AddTerminalField(CPDF_Dictionary* pFieldDict);
CPDF_FormControl* AddControl(CPDF_FormField* pField,
                            CPDF_Dictionary* pWidgetDict);

static bool s_bUpdateAP;

ByteString m_bsEncoding;
UnownedPtr<CPDF_Document> const m_pDocument;
RetainPtr<CPDF_Dictionary> m_pFormDict;
std::unique_ptr<CFieldTree> m_pFieldTree;
std::map<const CPDF_Dictionary*, std::unique_ptr<CPDF_FormControl>>
    m_ControlMap;
// Points into |m_ControlMap|.

```

```
    std::map<const CPDF_FormField*, std::vector<UnownedPtr<CPDF_FormControl>>>  
        m_ControlLists;  
    UnownedPtr<NotifierIface> m_pFormNotify;  
};  
  
#endif // CORE_FPDFDOC_CPDF_INTERACTIVEFORM_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_LINK_H_
#define CORE_FPDFDOC_CPDF_LINK_H_

#include "core/fpdfdoc/cpdf_action.h"
#include "core/fpdfdoc/cpdf_dest.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Dictionary;

class CPDF_Link {
public:
  CPDF_Link();
  explicit CPDF_Link(CPDF_Dictionary* pDict);
  CPDF_Link(const CPDF_Link& that);
  ~CPDF_Link();

  CPDF_Dictionary* GetDict() const { return m_pDict.Get(); }
  CFX_FloatRect GetRect();
  CPDF_Dest GetDest(CPDF_Document* pDoc);
  CPDF_Action GetAction();

private:
  RetainPtr<CPDF_Dictionary> m_pDict;
};

#endif // CORE_FPDFDOC_CPDF_LINK_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_LINKLIST_H_
#define CORE_FPDFDOC_CPDF_LINKLIST_H_

#include <map>
#include <vector>

#include "core/fpdfapi/parser/cpdf_document.h"
#include "core/fpdfdoc/cpdf_link.h"
#include "core/fxcrt/fx_system.h"

class CPDF_Page;
class CPDF_Dictionary;

class CPDF_LinkList : public CPDF_Document::LinkListIface {
public:
  CPDF_LinkList();
  ~CPDF_LinkList() override;

  CPDF_Link GetLinkAtPoint(CPDF_Page* pPage,
                          const CFX_PointF& point,
                          int* z_order);

private:
  const std::vector<CPDF_Dictionary*> GetPageLinks(CPDF_Page* pPage);
  void LoadPageLinks(CPDF_Page* pPage, std::vector<CPDF_Dictionary*>* pList);

  std::map<uint32_t, std::vector<CPDF_Dictionary*>> m_PageMap;
};

#endif // CORE_FPDFDOC_CPDF_LINKLIST_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_METADATA_H_
#define CORE_FPDFDOC_CPDF_METADATA_H_

#include <vector>

#include "core/fxcrt/retain_ptr.h"

class CPDF_Stream;

enum class UnsupportedFeature : uint8_t {
    kDocumentXFAForm = 1,
    kDocumentPortableCollection = 2,
    kDocumentAttachment = 3,
    kDocumentSecurity = 4,
    kDocumentSharedReview = 5,
    kDocumentSharedFormAcrobat = 6,
    kDocumentSharedFormFilesystem = 7,
    kDocumentSharedFormEmail = 8,

    kAnnotation3d = 11,
    kAnnotationMovie = 12,
    kAnnotationSound = 13,
    kAnnotationScreenMedia = 14,
    kAnnotationScreenRichMedia = 15,
    kAnnotationAttachment = 16,
    kAnnotationSignature = 17
};

class CPDF_Metadata {
public:
    explicit CPDF_Metadata(const CPDF_Stream* pStream);
    ~CPDF_Metadata();

    std::vector<UnsupportedFeature> CheckForSharedForm() const;

private:
    RetainPtr<const CPDF_Stream> stream_;
};

#endif // CORE_FPDFDOC_CPDF_METADATA_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_NAMETREE_H_
#define CORE_FPDFDOC_CPDF_NAMETREE_H_

#include <memory>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Array;
class CPDF_Dictionary;
class CPDF_Document;
class CPDF_Object;

class CPDF_NameTree {
public:
    explicit CPDF_NameTree(CPDF_Dictionary* pRoot);
    CPDF_NameTree(CPDF_Document* pDoc, const ByteString& category);
    ~CPDF_NameTree();

    bool AddValueAndName(RetainPtr<CPDF_Object> pObj, const WideString& name);
    bool DeleteValueAndName(int nIndex);

    CPDF_Object* LookupValueAndName(int nIndex, WideString* csName) const;
    CPDF_Object* LookupValue(const WideString& csName) const;
    CPDF_Array* LookupNamedDest(CPDF_Document* pDoc, const WideString& sName);

    int GetIndex(const WideString& csName) const;
    size_t GetCount() const;
    CPDF_Dictionary* GetRoot() const { return m_pRoot.Get(); }

private:
    RetainPtr<CPDF_Dictionary> m_pRoot;
};

#endif // CORE_FPDFDOC_CPDF_NAMETREE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_NUMBERTREE_H_
#define CORE_FPDFDOC_CPDF_NUMBERTREE_H_

#include "core/fxcrt/retain_ptr.h"

class CPDF_Dictionary;
class CPDF_Object;

class CPDF_NumberTree {
public:
  explicit CPDF_NumberTree(const CPDF_Dictionary* pRoot);
  ~CPDF_NumberTree();

  const CPDF_Object* LookupValue(int num) const;

protected:
  RetainPtr<const CPDF_Dictionary> const m_pRoot;
};

#endif // CORE_FPDFDOC_CPDF_NUMBERTREE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_PAGELABEL_H_
#define CORE_FPDFDOC_CPDF_PAGELABEL_H_

#include "core/fxcrt/fx_string.h"
#include "third_party/base/optional.h"

class CPDF_Document;

class CPDF_PageLabel {
public:
    explicit CPDF_PageLabel(CPDF_Document* pDocument);
    ~CPDF_PageLabel();

    Optional<WideString> GetLabel(int nPage) const;

private:
    UnownedPtr<CPDF_Document> const m_pDocument;
};

#endif // CORE_FPDFDOC_CPDF_PAGELABEL_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_STRUCTELEMENT_H_
#define CORE_FPDFDOC_CPDF_STRUCTELEMENT_H_

#include <vector>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Dictionary;
class CPDF_Object;
class CPDF_StructElement;
class CPDF_StructTree;

class CPDF_StructKid {
public:
    enum Type { kInvalid, kElement, kPageContent, kStreamContent, kObject };

    CPDF_StructKid();
    CPDF_StructKid(const CPDF_StructKid& that);
    ~CPDF_StructKid();

    Type m_Type = kInvalid;
    uint32_t m_PageObjNum = 0; // For {PageContent, StreamContent, Object} types.
    uint32_t m_RefObjNum = 0; // For {StreamContent, Object} types.
    uint32_t m_ContentId = 0; // For {PageContent, StreamContent} types.
    RetainPtr<CPDF_StructElement> m_pElement; // For Element.
    RetainPtr<const CPDF_Dictionary> m_pDict; // For Element.
};

class CPDF_StructElement final : public Retainable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    ByteString GetType() const { return m_Type; }
    WideString GetAltText() const;
    WideString GetTitle() const;

    // Never returns nullptr.
    const CPDF_Dictionary* GetDict() const { return m_pDict.Get(); }

    size_t CountKids() const;
    CPDF_StructElement* GetKidIfElement(size_t index) const;
    std::vector<CPDF_StructKid*> GetKids() { return &m_Kids; }

private:
    CPDF_StructElement(CPDF_StructTree* pTree,
                      CPDF_StructElement* pParent,
                      const CPDF_Dictionary* pDict);
    ~CPDF_StructElement() override;

    void LoadKids(const CPDF_Dictionary* pDict);
    void LoadKid(uint32_t PageObjNum,
                 const CPDF_Object* pKidObj,
                 CPDF_StructKid* pKid);
};
```

```
UnownedPtr<CPDF_StructTree> const m_pTree;
UnownedPtr<CPDF_StructElement> const m_pParent;
RetainPtr<const CPDF_Dictionary> const m_pDict;
const ByteString m_Type;
std::vector<CPDF_StructKid> m_Kids;
};

#endif // CORE_FPDFDOC_CPDF_STRUCTURELEMENT_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_STRUCTTREE_H_
#define CORE_FPDFDOC_CPDF_STRUCTTREE_H_

#include <map>
#include <memory>
#include <vector>

#include "core/fxcrt/retain_ptr.h"

class CPDF_Dictionary;
class CPDF_Document;
class CPDF_StructElement;

class CPDF_StructTree {
public:
    static std::unique_ptr<CPDF_StructTree> LoadPage(
        const CPDF_Document* pDoc,
        const CPDF_Dictionary* pPageDict);

    explicit CPDF_StructTree(const CPDF_Document* pDoc);
    ~CPDF_StructTree();

    size_t CountTopElements() const { return m_Kids.size(); }
    CPDF_StructElement* GetTopElement(size_t i) const { return m_Kids[i].Get(); }
    const CPDF_Dictionary* GetRoleMap() const { return m_pRoleMap.Get(); }
    const CPDF_Dictionary* GetPage() const { return m_pPage.Get(); }
    const CPDF_Dictionary* GetTreeRoot() const { return m_pTreeRoot.Get(); }

private:
    using StructElementMap =
        std::map<const CPDF_Dictionary*, RetainPtr<CPDF_StructElement>>;

    void LoadPageTree(const CPDF_Dictionary* pPageDict);
    RetainPtr<CPDF_StructElement> AddPageNode(const CPDF_Dictionary* pDict,
        StructElementMap* map,
        int nLevel);

    bool AddTopLevelNode(const CPDF_Dictionary* pDict,
        const RetainPtr<CPDF_StructElement>& pElement);

    RetainPtr<const CPDF_Dictionary> const m_pTreeRoot;
    RetainPtr<const CPDF_Dictionary> const m_pRoleMap;
    RetainPtr<const CPDF_Dictionary> m_pPage;
    std::vector<RetainPtr<CPDF_StructElement>> m_Kids;
};

#endif // CORE_FPDFDOC_CPDF_STRUCTTREE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_VARIABLETEXT_H_
#define CORE_FPDFDOC_CPDF_VARIABLETEXT_H_

#include <memory>
#include <vector>

#include "core/fpdfdoc/cpvt_floatrect.h"
#include "core/fpdfdoc/cpvt_line.h"
#include "core/fpdfdoc/cpvt_lineinfo.h"
#include "core/fpdfdoc/cpvt_wordplace.h"
#include "core/fpdfdoc/cpvt_wordrange.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"

class CPVT_Word;
class CSection;
class IPVT_FontMap;
struct CPVT_WordInfo;

#define VARIABLETEXT_HALF 0.5f

class CPDF_VariableText {
public:
  class Iterator {
public:
    explicit Iterator(CPDF_VariableText* pVT);
    ~Iterator();

    bool NextWord();
    bool PrevWord();
    bool NextLine();
    bool GetWord(CPVT_Word& word) const;
    bool GetLine(CPVT_Line& line) const;
    void SetAt(int32_t nWordIndex);
    void SetAt(const CPVT_WordPlace& place);
    const CPVT_WordPlace& GetWordPlace() const { return m_CurPos; }

private:
    CPVT_WordPlace m_CurPos;
    UnownedPtr<CPDF_VariableText> const m_pVT;
  };
};

class Provider {
public:
  explicit Provider(IPVT_FontMap* pFontMap);
  virtual ~Provider();

  virtual uint32_t GetCharWidth(int32_t nFontIndex, uint16_t word);
  virtual int32_t GetTypeAscent(int32_t nFontIndex);
  virtual int32_t GetTypeDescent(int32_t nFontIndex);
  virtual int32_t GetWordFontIndex(uint16_t word,
                                   int32_t charset,
                                   int32_t nFontIndex);
  virtual bool IsLatinWord(uint16_t word);
  virtual int32_t GetDefaultFontIndex();
};
```

```
private:
  UnownedPtr<IPVT_FontMap> const m_pFontMap;
};

CPDF_VariableText();
~CPDF_VariableText();

void SetProvider(CPDF_VariableText::Provider* pProvider);
CPDF_VariableText::Iterator* GetIterator();

void SetContentRect(const CPVT_FloatRect& rect);
CFX_FloatRect GetContentRect() const;
void SetPlateRect(const CFX_FloatRect& rect);
const CFX_FloatRect& GetPlateRect() const;

void SetAlignment(int32_t nFormat) { m_nAlignment = nFormat; }
void SetPasswordChar(uint16_t wSubWord) { m_wSubWord = wSubWord; }
void SetLimitChar(int32_t nLimitChar) { m_nLimitChar = nLimitChar; }
void SetCharSpace(float fCharSpace) { m_fCharSpace = fCharSpace; }
void SetMultiLine(bool bMultiLine) { m_bMultiLine = bMultiLine; }
void SetAutoReturn(bool bAuto) { m_bLimitWidth = bAuto; }
void SetFontSize(float fFontSize) { m_fFontSize = fFontSize; }
void SetCharArray(int32_t nCharArray) { m_nCharArray = nCharArray; }
void SetAutoFontSize(bool bAuto) { m_bAutoFontSize = bAuto; }
void Initialize();

bool IsValid() const { return m_bInitialized; }

void RearrangeAll();
void RearrangePart(const CPVT_WordRange& PlaceRange);
void SetText(const WideString& text);
CPVT_WordPlace InsertWord(const CPVT_WordPlace& place,
                          uint16_t word,
                          int32_t charset);
CPVT_WordPlace InsertSection(const CPVT_WordPlace& place);
CPVT_WordPlace DeleteWords(const CPVT_WordRange& PlaceRange);
CPVT_WordPlace DeleteWord(const CPVT_WordPlace& place);
CPVT_WordPlace BackSpaceWord(const CPVT_WordPlace& place);

int32_t GetTotalWords() const;
float GetFontSize() const { return m_fFontSize; }
int32_t GetAlignment() const { return m_nAlignment; }
uint16_t GetPasswordChar() const { return GetSubWord(); }
int32_t GetCharArray() const { return m_nCharArray; }
int32_t GetLimitChar() const { return m_nLimitChar; }
bool IsMultiLine() const { return m_bMultiLine; }
float GetCharSpace() const { return m_fCharSpace; }
bool IsAutoReturn() const { return m_bLimitWidth; }

CPVT_WordPlace GetBeginWordPlace() const;
CPVT_WordPlace GetEndWordPlace() const;
CPVT_WordPlace GetPrevWordPlace(const CPVT_WordPlace& place) const;
CPVT_WordPlace GetNextWordPlace(const CPVT_WordPlace& place) const;
CPVT_WordPlace SearchWordPlace(const CFX_PointF& point) const;
CPVT_WordPlace GetUpWordPlace(const CPVT_WordPlace& place,
                              const CFX_PointF& point) const;
CPVT_WordPlace GetDownWordPlace(const CPVT_WordPlace& place,
                                 const CFX_PointF& point) const;
CPVT_WordPlace GetLineBeginPlace(const CPVT_WordPlace& place) const;
CPVT_WordPlace GetLineEndPlace(const CPVT_WordPlace& place) const;
CPVT_WordPlace GetSectionBeginPlace(const CPVT_WordPlace& place) const;
CPVT_WordPlace GetSectionEndPlace(const CPVT_WordPlace& place) const;
```

```
void UpdateWordPlace(CPVT_WordPlace& place) const;
CPVT_WordPlace AdjustLineHeader(const CPVT_WordPlace& place,
                                bool bPrevOrNext) const;
int32_t WordPlaceToWordIndex(const CPVT_WordPlace& place) const;
CPVT_WordPlace WordIndexToWordPlace(int32_t index) const;

uint16_t GetSubWord() const { return m_wSubWord; }

float GetPlateWidth() const { return m_rcPlate.right - m_rcPlate.left; }
float GetPlateHeight() const { return m_rcPlate.top - m_rcPlate.bottom; }
CFX_PointF GetBTPoint() const;
CFX_PointF GetETPoint() const;

CFX_PointF InToOut(const CFX_PointF& point) const;
CFX_PointF OutToIn(const CFX_PointF& point) const;
CFX_FloatRect InToOut(const CPVT_FloatRect& rect) const;
CPVT_FloatRect OutToIn(const CFX_FloatRect& rect) const;

float GetFontAscent(int32_t nFontIndex, float fFontSize);
float GetFontDescent(int32_t nFontIndex, float fFontSize);
int32_t GetDefaultFontIndex();
float GetLineLeading();
int32_t GetAlignment();
float GetWordWidth(const CPVT_WordInfo& WordInfo);
float GetWordWidth(int32_t nFontIndex,
                   uint16_t Word,
                   uint16_t SubWord,
                   float fCharSpace,
                   float fFontSize,
                   float fWordTail);
float GetWordAscent(const CPVT_WordInfo& WordInfo);
float GetWordDescent(const CPVT_WordInfo& WordInfo);
float GetWordAscent(const CPVT_WordInfo& WordInfo, float fFontSize);
float GetWordDescent(const CPVT_WordInfo& WordInfo, float fFontSize);
float GetLineAscent();
float GetLineDescent();
float GetLineIndent();

private:
uint32_t GetCharWidth(int32_t nFontIndex, uint16_t Word, uint16_t SubWord);
int32_t GetTypeAscent(int32_t nFontIndex);
int32_t GetTypeDescent(int32_t nFontIndex);
int32_t GetWordFontIndex(uint16_t word, int32_t charset, int32_t nFontIndex);
bool IsLatinWord(uint16_t word);

CPVT_WordPlace AddSection(const CPVT_WordPlace& place);
CPVT_WordPlace AddLine(const CPVT_WordPlace& place,
                       const CPVT_LineInfo& lineinfo);
CPVT_WordPlace AddWord(const CPVT_WordPlace& place,
                       const CPVT_WordInfo& wordinfo);
float GetWordFontSize();
int32_t GetWordFontIndex(const CPVT_WordInfo& WordInfo);

void ClearSectionRightWords(const CPVT_WordPlace& place);

bool ClearEmptySection(const CPVT_WordPlace& place);
void ClearEmptySections(const CPVT_WordRange& PlaceRange);
void LinkLatterSection(const CPVT_WordPlace& place);
void ClearWords(const CPVT_WordRange& PlaceRange);
CPVT_WordPlace ClearLeftWord(const CPVT_WordPlace& place);
CPVT_WordPlace ClearRightWord(const CPVT_WordPlace& place);

CPVT_FloatRect Rearrange(const CPVT_WordRange& PlaceRange);
```

```
float GetAutoFontSize();
bool IsBigger(float fFontSize) const;
CPVT_FloatRect RearrangeSections(const CPVT_WordRange& PlaceRange);

bool m_bInitialized = false;
bool m_bMultiLine = false;
bool m_bLimitWidth = false;
bool m_bAutoFontSize = false;
uint16_t m_wSubWord = 0;
int32_t m_nLimitChar = 0;
int32_t m_nCharArray = 0;
int32_t m_nAlignment = 0;
float m_fLineLeading = 0.0f;
float m_fCharSpace = 0.0f;
float m_fFontSize = 0.0f;
std::vector<std::unique_ptr<CSection>> m_SectionArray;
UnownedPtr<CPDF_VariableText::Provider> m_pVTProvider;
std::unique_ptr<CPDF_VariableText::Iterator> m_pVTIterator;
CFX_FloatRect m_rcPlate;
CPVT_FloatRect m_rcContent;
};

#endif // CORE_FPDFDOC_CPDF_VARIABLETEXT_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPDF_VIEWERPREFERENCES_H_
#define CORE_FPDFDOC_CPDF_VIEWERPREFERENCES_H_

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/optional.h"

class CPDF_Array;
class CPDF_Dictionary;
class CPDF_Document;

class CPDF_ViewerPreferences {
public:
  explicit CPDF_ViewerPreferences(const CPDF_Document* pDoc);
  ~CPDF_ViewerPreferences();

  bool IsDirectionR2L() const;
  bool PrintScaling() const;
  int32_t NumCopies() const;
  CPDF_Array* PrintPageRange() const;
  ByteString Duplex() const;

  // Gets the entry for |bsKey|.
  Optional<ByteString> GenericName(const ByteString& bsKey) const;

private:
  CPDF_Dictionary* GetViewerPreferences() const;

  UnownedPtr<const CPDF_Document> const m_pDoc;
};

#endif // CORE_FPDFDOC_CPDF_VIEWERPREFERENCES_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPVT_FLOATRECT_H_
#define CORE_FPDFDOC_CPVT_FLOATRECT_H_

#include "core/fxcrt/fx_coordinates.h"

class CPVT_FloatRect final : public CFX_FloatRect {
public:
  CPVT_FloatRect() = default;

  CPVT_FloatRect(float other_left,
                 float other_top,
                 float other_right,
                 float other_bottom)
    : CFX_FloatRect(other_left, other_bottom, other_right, other_top) {}

  explicit CPVT_FloatRect(const CFX_FloatRect& rect)
    : CFX_FloatRect(rect.left, rect.bottom, rect.right, rect.top) {}

  float Height() const {
    if (top > bottom)
      return top - bottom;
    return bottom - top;
  }
};

#endif // CORE_FPDFDOC_CPVT_FLOATRECT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPVT_FONTMAP_H_
#define CORE_FPDFDOC_CPVT_FONTMAP_H_

#include <stdint.h>

#include "core/fpdfdoc/ipvt_fontmap.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_Document;
class CPDF_Dictionary;
class CPDF_Font;

class CPVT_FontMap final : public IPVT_FontMap {
public:
  CPVT_FontMap(CPDF_Document* pDoc,
               CPDF_Dictionary* pResDict,
               const RetainPtr<CPDF_Font>& pDefFont,
               const ByteString& sDefFontAlias);
  ~CPVT_FontMap() override;

  // IPVT_FontMap:
  RetainPtr<CPDF_Font> GetPDFFont(int32_t nFontIndex) override;
  ByteString GetPDFFontAlias(int32_t nFontIndex) override;
  int32_t GetWordFontIndex(uint16_t word,
                           int32_t charset,
                           int32_t nFontIndex) override;
  int32_t CharCodeFromUnicode(int32_t nFontIndex, uint16_t word) override;
  int32_t CharSetFromUnicode(uint16_t word, int32_t nOldCharset) override;

  static RetainPtr<CPDF_Font> GetAnnotSysPDFFont(CPDF_Document* pDoc,
                                                  CPDF_Dictionary* pResDict,
                                                  ByteString* sSysFontAlias);

private:
  UnownedPtr<CPDF_Document> const m_pDocument;
  RetainPtr<CPDF_Dictionary> const m_pResDict;
  RetainPtr<CPDF_Font> const m_pDefFont;
  RetainPtr<CPDF_Font> m_pSysFont;
  const ByteString m_sDefFontAlias;
  ByteString m_sSysFontAlias;
};

#endif // CORE_FPDFDOC_CPVT_FONTMAP_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPVT_GENERATEAP_H_
#define CORE_FPDFDOC_CPVT_GENERATEAP_H_

#include "core/fpdfdoc/cpdf_annot.h"
#include "core/fxcrt/fx_system.h"

class CPDF_Dictionary;
class CPDF_Document;

class CPVT_GenerateAP {
public:
    enum FormType { kTextField, kComboBox, kListBox };

    static void GenerateFormAP(CPDF_Document* pDoc,
                              CPDF_Dictionary* pAnnotDict,
                              FormType type);

    static void GenerateEmptyAP(CPDF_Document* pDoc, CPDF_Dictionary* pAnnotDict);

    static bool GenerateAnnotAP(CPDF_Document* pDoc,
                                CPDF_Dictionary* pAnnotDict,
                                CPDF_Annot::Subtype subtype);

    CPVT_GenerateAP() = delete;
    CPVT_GenerateAP(const CPVT_GenerateAP&) = delete;
    CPVT_GenerateAP& operator=(const CPVT_GenerateAP&) = delete;
};

#endif // CORE_FPDFDOC_CPVT_GENERATEAP_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPVT_LINE_H_
#define CORE_FPDFDOC_CPVT_LINE_H_

#include "core/fpdfdoc/cpvt_wordplace.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"

class CPVT_Line {
public:
  CPVT_Line();

  CPVT_WordPlace lineplace;
  CPVT_WordPlace lineEnd;
  CFX_PointF ptLine;
  float fLineWidth;
  float fLineAscent;
  float fLineDescent;
};

inline CPVT_Line::CPVT_Line()
    : fLineWidth(0.0f), fLineAscent(0.0f), fLineDescent(0.0f) {}

#endif // CORE_FPDFDOC_CPVT_LINE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPVT_LINEINFO_H_
#define CORE_FPDFDOC_CPVT_LINEINFO_H_

#include "core/fxcrt/fx_system.h"

class CPVT_LineInfo {
public:
  CPVT_LineInfo();

  int32_t nTotalWord;
  int32_t nBeginWordIndex;
  int32_t nEndWordIndex;
  float fLineX;
  float fLineY;
  float fLineWidth;
  float fLineAscent;
  float fLineDescent;
};

inline CPVT_LineInfo::CPVT_LineInfo()
  : nTotalWord(0),
    nBeginWordIndex(-1),
    nEndWordIndex(-1),
    fLineX(0.0f),
    fLineY(0.0f),
    fLineWidth(0.0f),
    fLineAscent(0.0f),
    fLineDescent(0.0f) {}

#endif // CORE_FPDFDOC_CPVT_LINEINFO_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPVT_WORD_H_
#define CORE_FPDFDOC_CPVT_WORD_H_

#include "core/fpdfdoc/cpvt_wordplace.h"
#include "core/fxcrt/fx_system.h"

class CPVT_Word {
public:
  CPVT_Word();

  uint16_t Word;
  int32_t nCharset;
  CPVT_WordPlace WordPlace;
  CFX_PointF ptWord;
  float fAscent;
  float fDescent;
  float fWidth;
  int32_t nFontIndex;
  float fFontSize;
};

inline CPVT_Word::CPVT_Word()
  : Word(0),
    nCharset(0),
    fAscent(0.0f),
    fDescent(0.0f),
    fWidth(0.0f),
    nFontIndex(-1),
    fFontSize(0.0f) {}

#endif // CORE_FPDFDOC_CPVT_WORD_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPVT_WORDINFO_H_
#define CORE_FPDFDOC_CPVT_WORDINFO_H_

#include "core/fxcrt/fx_system.h"

struct CPVT_WordInfo {
  CPVT_WordInfo();
  CPVT_WordInfo(uint16_t word, int32_t charset, int32_t fontIndex);
  CPVT_WordInfo(const CPVT_WordInfo& word);
  ~CPVT_WordInfo();

  CPVT_WordInfo& operator=(const CPVT_WordInfo& word);

  uint16_t Word;
  int32_t nCharset;
  float fWordX;
  float fWordY;
  float fWordTail;
  int32_t nFontIndex;
};

#endif // CORE_FPDFDOC_CPVT_WORDINFO_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPVT_WORDPLACE_H_
#define CORE_FPDFDOC_CPVT_WORDPLACE_H_

#include "core/fxcrt/fx_system.h"

struct CPVT_WordPlace {
  CPVT_WordPlace() : nSecIndex(-1), nLineIndex(-1), nWordIndex(-1) {}

  CPVT_WordPlace(int32_t other_nSecIndex,
                 int32_t other_nLineIndex,
                 int32_t other_nWordIndex)
    : nSecIndex(other_nSecIndex),
      nLineIndex(other_nLineIndex),
      nWordIndex(other_nWordIndex) {}

  void Reset() {
    nSecIndex = -1;
    nLineIndex = -1;
    nWordIndex = -1;
  }

  void AdvanceSection() {
    nSecIndex++;
    nLineIndex = 0;
    nWordIndex = -1;
  }

  inline bool operator==(const CPVT_WordPlace& wp) const {
    return wp.nSecIndex == nSecIndex && wp.nLineIndex == nLineIndex &&
           wp.nWordIndex == nWordIndex;
  }

  inline bool operator!=(const CPVT_WordPlace& wp) const {
    return !(*this == wp);
  }

  inline bool operator<(const CPVT_WordPlace& wp) const {
    if (nSecIndex != wp.nSecIndex)
      return nSecIndex < wp.nSecIndex;
    if (nLineIndex != wp.nLineIndex)
      return nLineIndex < wp.nLineIndex;
    return nWordIndex < wp.nWordIndex;
  }

  inline bool operator>(const CPVT_WordPlace& wp) const {
    if (nSecIndex != wp.nSecIndex)
      return nSecIndex > wp.nSecIndex;
    if (nLineIndex != wp.nLineIndex)
      return nLineIndex > wp.nLineIndex;
    return nWordIndex > wp.nWordIndex;
  }

  inline bool operator<=(const CPVT_WordPlace& wp) const {
    return *this < wp || *this == wp;
  }

  inline bool operator>=(const CPVT_WordPlace& wp) const {
    return *this > wp || *this == wp;
  }

  inline int32_t LineCmp(const CPVT_WordPlace& wp) const {
    if (nSecIndex != wp.nSecIndex)
```

```
    return nSecIndex - wp.nSecIndex;
    return nLineIndex - wp.nLineIndex;
}

int32_t nSecIndex;
int32_t nLineIndex;
int32_t nWordIndex;
};

#endif // CORE_FPDFDOC_CPVT_WORDPLACE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CPVT_WORDRANGE_H_
#define CORE_FPDFDOC_CPVT_WORDRANGE_H_

#include <algorithm>
#include <utility>

#include "core/fpdfdoc/cpvt_wordplace.h"
#include "core/fxcrt/fx_system.h"

struct CPVT_WordRange {
    CPVT_WordRange() = default;

    CPVT_WordRange(const CPVT_WordPlace& begin, const CPVT_WordPlace& end)
        : BeginPos(begin), EndPos(end) {
        Normalize();
    }

    inline bool IsEmpty() const { return BeginPos == EndPos; }
    inline bool operator==(const CPVT_WordRange& wr) const {
        return wr.BeginPos == BeginPos && wr.EndPos == EndPos;
    }
    inline bool operator!=(const CPVT_WordRange& wr) const {
        return !(*this == wr);
    }

    void Normalize() {
        if (BeginPos > EndPos)
            std::swap(BeginPos, EndPos);
    }

    CPVT_WordPlace BeginPos;
    CPVT_WordPlace EndPos;
};

#endif // CORE_FPDFDOC_CPVT_WORDRANGE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CSECTION_H_
#define CORE_FPDFDOC_CSECTION_H_

#include <memory>
#include <vector>

#include "core/fpdfdoc/ccline.h"
#include "core/fpdfdoc/cpvt_wordinfo.h"
#include "core/fpdfdoc/cpvt_wordrange.h"
#include "core/fpdfdoc/ctypeset.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"

class CPDF_VariableText;
class CPVT_LineInfo;
struct CPVT_WordLine;
struct CPVT_WordPlace;

class CSection final {
public:
    explicit CSection(CPDF_VariableText* pVT);
    ~CSection();

    void ResetLinePlace();
    CPVT_WordPlace AddWord(const CPVT_WordPlace& place,
                          const CPVT_WordInfo& wordinfo);
    CPVT_WordPlace AddLine(const CPVT_LineInfo& lineinfo);
    void ClearWords(const CPVT_WordRange& PlaceRange);
    void ClearWord(const CPVT_WordPlace& place);
    CPVT_FloatRect Rearrange();
    CFX_SizeF GetSectionSize(float fFontSize);
    CPVT_WordPlace GetBeginWordPlace() const;
    CPVT_WordPlace GetEndWordPlace() const;
    CPVT_WordPlace GetPrevWordPlace(const CPVT_WordPlace& place) const;
    CPVT_WordPlace GetNextWordPlace(const CPVT_WordPlace& place) const;
    void UpdateWordPlace(CPVT_WordPlace& place) const;
    CPVT_WordPlace SearchWordPlace(const CFX_PointF& point) const;
    CPVT_WordPlace SearchWordPlace(float fx,
                                    const CPVT_WordPlace& lineplace) const;
    CPVT_WordPlace SearchWordPlace(float fx, const CPVT_WordRange& range) const;

    CPVT_WordPlace SecPlace;
    CPVT_FloatRect m_Rect;
    std::vector<std::unique_ptr<CLine>> m_LineArray;
    std::vector<std::unique_ptr<CPVT_WordInfo>> m_WordArray;

private:
    friend class CTypeset;

    void ClearLeftWords(int32_t nWordIndex);
    void ClearRightWords(int32_t nWordIndex);
    void ClearMidWords(int32_t nBeginIndex, int32_t nEndIndex);

    UnownedPtr<CPDF_VariableText> const m_pVT;
};

#endif // CORE_FPDFDOC_CSECTION_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_CTYPESET_H_
#define CORE_FPDFDOC_CTYPESET_H_

#include "core/fpdfdoc/cpvt_floatrect.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"

class CPDF_VariableText;
class CSection;

class CTypeset final {
public:
  explicit CTypeset(CSection* pSection);
  ~CTypeset();

  CFX_SizeF GetEditSize(float fFontSize);
  CPVT_FloatRect Typeset();
  CPVT_FloatRect CharArray();

private:
  void SplitLines(bool bTypeset, float fFontSize);
  void OutputLines();

  CPVT_FloatRect m_rcRet;
  UnownedPtr<CPDF_VariableText> const m_pVT;
  CSection* const m_pSection;
};

#endif // CORE_FPDFDOC_CTYPESET_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFDOC_IPVT_FONTMAP_H_
#define CORE_FPDFDOC_IPVT_FONTMAP_H_

#include <stdint.h>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"

class CPDF_Font;

class IPVT_FontMap {
public:
    virtual ~IPVT_FontMap() = default;

    virtual RetainPtr<CPDF_Font> GetPDFFont(int32_t nFontIndex) = 0;
    virtual ByteString GetPDFFontAlias(int32_t nFontIndex) = 0;
    virtual int32_t GetWordFontIndex(uint16_t word,
                                     int32_t charset,
                                     int32_t nFontIndex) = 0;
    virtual int32_t CharCodeFromUnicode(int32_t nFontIndex, uint16_t word) = 0;
    virtual int32_t CharSetFromUnicode(uint16_t word, int32_t nOldCharset) = 0;
};

#endif // CORE_FPDFDOC_IPVT_FONTMAP_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFTEXT_CPDF_LINKEXTRACT_H_
#define CORE_FPDFTEXT_CPDF_LINKEXTRACT_H_

#include <vector>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"

class CPDF_TextPage;

class CPDF_LinkExtract {
public:
  explicit CPDF_LinkExtract(const CPDF_TextPage* pTextPage);
  ~CPDF_LinkExtract();

  void ExtractLinks();
  size_t CountLinks() const { return m_LinkArray.size(); }
  WideString GetURL(size_t index) const;
  std::vector<CFX_FloatRect> GetRects(size_t index) const;
  bool GetTextRange(size_t index, int* start_char_index, int* char_count) const;

protected:
  void ParseLink();
  bool CheckWebLink(WideString* str, int32_t* nStart, int32_t* nCount);
  bool CheckMailLink(WideString* str);

private:
  struct Link {
    int m_Start;
    int m_Count;
    WideString m_strUrl;
  };

  UnownedPtr<const CPDF_TextPage> const m_pTextPage;
  WideString m_strPageText;
  std::vector<Link> m_LinkArray;
};

#endif // CORE_FPDFTEXT_CPDF_LINKEXTRACT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFTEXT_CPDF_TEXTPAGEFIND_H_
#define CORE_FPDFTEXT_CPDF_TEXTPAGEFIND_H_

#include <memory>
#include <vector>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/optional.h"

class CPDF_TextPage;

class CPDF_TextPageFind {
public:
  struct Options {
    bool bMatchCase = false;
    bool bMatchWholeWord = false;
    bool bConsecutive = false;
  };

  static std::unique_ptr<CPDF_TextPageFind> Create(
    const CPDF_TextPage* pTextPage,
    const WideString& findwhat,
    const Options& options,
    Optional<size_t> startPos);

  ~CPDF_TextPageFind();

  bool FindNext();
  bool FindPrev();
  int GetCurOrder() const;
  int GetMatchedCount() const;

private:
  CPDF_TextPageFind(const CPDF_TextPage* pTextPage,
                    const std::vector<WideString>& findwhat_array,
                    const Options& options,
                    Optional<size_t> startPos);

  // Should be called immediately after construction.
  bool FindFirst();

  int GetCharIndex(int index) const;

  UnownedPtr<const CPDF_TextPage> const m_pTextPage;
  const WideString m_strText;
  const std::vector<WideString> m_csFindWhatArray;
  Optional<size_t> m_findNextStart;
  Optional<size_t> m_findPreStart;
  int m_resStart = 0;
  int m_resEnd = -1;
  const Options m_options;
};

#endif // CORE_FPDFTEXT_CPDF_TEXTPAGEFIND_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFTEXT_CPDF_TEXTPAGE_H_
#define CORE_FPDFTEXT_CPDF_TEXTPAGE_H_

#include <deque>
#include <functional>
#include <vector>

#include "core/fpdfapi/page/cpdf_pageobjectholder.h"
#include "core/fxcrt/cfx_widertextbuf.h"
#include "core/fxcrt/cfx_coordinates.h"
#include "core/fxcrt/cfx_string.h"
#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/optional.h"

class CPDF_Font;
class CPDF_FormObject;
class CPDF_Page;
class CPDF_TextObject;

#define FPDFTEXT_CHAR_NORMAL 0
#define FPDFTEXT_CHAR_GENERATED 1
#define FPDFTEXT_CHAR_UNUNICODE 2
#define FPDFTEXT_CHAR_HYPHEN 3
#define FPDFTEXT_CHAR_PIECE 4

#define TEXT_SPACE_CHAR L' '
#define TEXT_LINEFEED_CHAR L'\n'
#define TEXT_RETURN_CHAR L'\r'
#define TEXT_HYPHEN_CHAR L'-'
#define TEXT_HYPHEN L"-"
#define TEXT_CHARRATIO_GAPDELTA 0.070

enum class FPDFText_MarkedContent { Pass = 0, Done, Delay };

enum class FPDFText_Direction { Left = -1, Right = 1 };

class FPDF_CHAR_INFO {
public:
    FPDF_CHAR_INFO();
    ~FPDF_CHAR_INFO();

    wchar_t m_Unicode = 0;
    wchar_t m_Charcode = 0;
    int32_t m_Flag = 0;
    float m_FontSize = 0;
    CFX_PointF m-Origin;
    CFX_FloatRect m_CharBox;
    UnownedPtr<CPDF_TextObject> m_pTextObj;
    CFX_Matrix m_Matrix;
};

class PAGECHAR_INFO {
public:
    PAGECHAR_INFO();
    PAGECHAR_INFO(const PAGECHAR_INFO&);
    ~PAGECHAR_INFO();
};
```

```
int m_Index = 0;
int m_CharCode = 0;
wchar_t m_Unicode = 0;
int32_t m_Flag = 0;
CFX_PointF m-Origin;
CFX_FloatRect m_CharBox;
UnownedPtr<CPDF_TextObject> m_pTextObj;
CFX_Matrix m_Matrix;
};

struct PDFTEXT_Obj {
    PDFTEXT_Obj();
    PDFTEXT_Obj(const PDFTEXT_Obj& that);
    ~PDFTEXT_Obj();

    UnownedPtr<CPDF_TextObject> m_pTextObj;
    CFX_Matrix m_formMatrix;
};

class CPDF_TextPage {
public:
    CPDF_TextPage(const CPDF_Page* pPage, FPDFText_Direction flags);
    ~CPDF_TextPage();

    void ParseTextPage();
    bool IsParsed() const { return m_bIsParsed; }
    int CharIndexFromTextIndex(int TextIndex) const;
    int TextIndexFromCharIndex(int CharIndex) const;
    size_t size() const { return m_CharList.size(); }
    int CountChars() const;
    void GetCharInfo(size_t index, FPDF_CHAR_INFO* info) const;
    std::vector<CFX_FloatRect> GetRectArray(int start, int nCount) const;
    int GetIndexAtPos(const CFX_PointF& point, const CFX_SizeF& tolerance) const;
    WideString GetTextByRect(const CFX_FloatRect& rect) const;
    WideString GetTextByObject(const CPDF_TextObject* pTextObj) const;

    // Returns string with the text from |m_TextBuf| that are covered by the input
    // range. |start| and |count| are in terms of the |m_CharIndex|, so the range
    // will be converted into appropriate indices.
    WideString GetPageText(int start, int count) const;
    WideString GetAllPageText() const { return GetPageText(0, CountChars()); }

    int CountRects(int start, int nCount);
    bool GetRect(int rectIndex, CFX_FloatRect* pRect) const;

private:
    enum class TextOrientation {
        Unknown,
        Horizontal,
        Vertical,
    };

    enum class GenerateCharacter {
        None,
        Space,
        LineBreak,
        Hyphen,
    };

    bool IsHyphen(wchar_t curChar) const;
    bool IsControlChar(const PAGECHAR_INFO& charInfo);
    void ProcessObject();
    void ProcessFormObject(CPDF_FormObject* pFormObj,
```

```
        const CFX_Matrix& formMatrix);
void ProcessTextObject(PDFTEXT_Obj pObj);
void ProcessTextObject(CPDF_TextObject* pTextObj,
        const CFX_Matrix& formMatrix,
        const CPDF_PageObjectHolder* pObjList,
        CPDF_PageObjectHolder::const_iterator ObjPos);
GenerateCharacter ProcessInsertObject(const CPDF_TextObject* pObj,
        const CFX_Matrix& formMatrix);
const PAGECHAR_INFO* GetPrevCharInfo() const;
Optional<PAGECHAR_INFO> GenerateCharInfo(wchar_t unicode);
bool IsSameAsPreTextObject(CPDF_TextObject* pTextObj,
        const CPDF_PageObjectHolder* pObjList,
        CPDF_PageObjectHolder::const_iterator iter);
bool IsSameTextObject(CPDF_TextObject* pTextObj1, CPDF_TextObject* pTextObj2);
void CloseTempLine();
FPDFText_MarkedContent PreMarkedContent(PDFTEXT_Obj pObj);
void ProcessMarkedContent(PDFTEXT_Obj pObj);
void FindPreviousTextObject();
void AddCharInfoByLRDirection(wchar_t wChar, const PAGECHAR_INFO& info);
void AddCharInfoByRLDirection(wchar_t wChar, const PAGECHAR_INFO& info);
TextOrientation GetTextObjectWritingMode(
        const CPDF_TextObject* pTextObj) const;
TextOrientation FindTextlineFlowOrientation() const;
void AppendGeneratedCharacter(wchar_t unicode, const CFX_Matrix& formMatrix);
void SwapTempTextBuf(int32_t iCharListStartAppend, int32_t iBufStartAppend);
WideString GetTextByPredicate(
        const std::function<bool(const PAGECHAR_INFO&)>& predicate) const;

UnownedPtr<const CPDF_Page> const m_pPage;
std::vector<uint16_t> m_CharIndex;
std::deque<PAGECHAR_INFO> m_CharList;
std::deque<PAGECHAR_INFO> m_TempCharList;
CFX_WideTextBuf m_TextBuf;
CFX_WideTextBuf m_TempTextBuf;
const FPDFText_Direction m_parserflag;
UnownedPtr<CPDF_TextObject> m_pPreTextObj;
CFX_Matrix m_perMatrix;
bool m_bIsParsed = false;
CFX_Matrix m_DisplayMatrix;
std::vector<CFX_FloatRect> m_SelRects;
std::vector<PDFTEXT_Obj> m_LineObj;
TextOrientation m_TextlineDir = TextOrientation::Unknown;
CFX_FloatRect m_CurlineRect;
};

#endif // CORE_FPDFTEXT_CPDF_TEXTPAGE_H
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FPDFTEXT_UNICODENORMALIZATIONDATA_H_
#define CORE_FPDFTEXT_UNICODENORMALIZATIONDATA_H_

#include <stdint.h>

extern const uint16_t g_UnicodeData_Normalization[];
extern const uint16_t g_UnicodeData_Normalization_Map1[];
extern const uint16_t g_UnicodeData_Normalization_Map2[];
extern const uint16_t g_UnicodeData_Normalization_Map3[];
extern const uint16_t g_UnicodeData_Normalization_Map4[];

#endif // CORE_FPDFTEXT_UNICODENORMALIZATIONDATA_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_BASIC_BASICMODULE_H_
#define CORE_FXCODEC_BASIC_BASICMODULE_H_

#include <memory>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_system.h"
#include "third_party/base/span.h"

namespace fxcodec {

class ScanlineDecoder;

class BasicModule {
public:
    static std::unique_ptr<ScanlineDecoder> CreateRunLengthDecoder(
        pdfium::span<const uint8_t> src_buf,
        int width,
        int height,
        int nComps,
        int bpc);

    static bool RunLengthEncode(pdfium::span<const uint8_t> src_span,
                                std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,
                                uint32_t* dest_size);

    static bool A85Encode(pdfium::span<const uint8_t> src_span,
                           std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,
                           uint32_t* dest_size);

    BasicModule() = delete;
    BasicModule(const BasicModule&) = delete;
    BasicModule& operator=(const BasicModule&) = delete;
};

} // namespace fxcodec

using BasicModule = fxcodec::BasicModule;

#endif // CORE_FXCODEC_BASIC_BASICMODULE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_BMP_BMPMODULE_H_
#define CORE_FXCODEC_BMP_BMPMODULE_H_

#include <memory>
#include <vector>

#include "core/fxcodec/codec_module_iface.h"
#include "third_party/base/span.h"

namespace fxcodec {

class CFX_DIBAttribute;

class BmpModule final : public ModuleIface {
public:
    class Delegate {
    public:
        virtual bool BmpInputImagePositionBuf(uint32_t rcd_pos) = 0;
        virtual void BmpReadScanline(uint32_t row_num,
                                      pdfium::span<const uint8_t> row_buf) = 0;
    };

    enum class Status : uint8_t { kFail, kSuccess, kContinue };

    BmpModule();
    ~BmpModule() override;

    // ModuleIface:
    FX_FILESIZE GetAvailInput(Context* pContext) const override;
    bool Input(Context* pContext,
               RetainPtr<CFX_CodecMemory> codec_memory,
               CFX_DIBAttribute* pAttribute) override;

    std::unique_ptr<Context> Start(Delegate* pDelegate);
    Status ReadHeader(Context* pContext,
                     int32_t* width,
                     int32_t* height,
                     bool* tb_flag,
                     int32_t* components,
                     int32_t* pal_num,
                     const std::vector<uint32_t>*& palette,
                     CFX_DIBAttribute* pAttribute);
    Status LoadImage(Context* pContext);
};

} // namespace fxcodec

using BmpModule = fxcodec::BmpModule;

#endif // CORE_FXCODEC_BMP_BMPMODULE_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_BMP_CFX_BMPCONTEXT_H_
#define CORE_FXCODEC_BMP_CFX_BMPCONTEXT_H_

#include "core/fxcodec/bmp/bmpmodule.h"
#include "core/fxcodec/bmp/cfx_bmpdecompressor.h"
#include "core/fxcodec/bmp/fx_bmp.h"
#include "core/fxcrt/unowned_ptr.h"

namespace fxcodec {

class CFX_BmpContext final : public ModuleIface::Context {
 public:
  CFX_BmpContext(BmpModule* pModule, BmpModule::Delegate* pDelegate);
  ~CFX_BmpContext() override;

  CFX_BmpDecompressor m_Bmp;
  UnownedPtr<BmpModule> const m_pModule;
  UnownedPtr<BmpModule::Delegate> const m_pDelegate;
};

} // namespace fxcodec

#endif // CORE_FXCODEC_BMP_CFX_BMPCONTEXT_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_BMP_CFX_BMPDECOMPRESSOR_H
#define CORE_FXCODEC_BMP_CFX_BMPDECOMPRESSOR_H

#include <vector>

#include "core/fxcodec/bmp/bmpmodule.h"
#include "core/fxcodec/bmp/fx_bmp.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"

class CFX_CodecMemory;

namespace fxcodec {

class CFX_BmpContext;

class CFX_BmpDecompressor {
public:
    explicit CFX_BmpDecompressor(CFX_BmpContext* context);
    ~CFX_BmpDecompressor();

    BmpModule::Status DecodeImage();
    BmpModule::Status ReadHeader();
    void SetInputBuffer(RetainPtr<CFX_CodecMemory> codec_memory);
    FX_FILESIZE GetAvailInput() const;

    const std::vector<uint32_t>* palette() const { return &palette_; }
    uint32_t width() const { return width_; }
    uint32_t height() const { return height_; }
    int32_t components() const { return components_; }
    bool img_tb_flag() const { return img_tb_flag_; }
    int32_t pal_num() const { return pal_num_; }
    int32_t dpi_x() const { return dpi_x_; }
    int32_t dpi_y() const { return dpi_y_; }

private:
    enum class DecodeStatus : uint8_t {
        kHeader,
        kPal,
        kDataPre,
        kData,
        kTail,
    };

    BmpModule::Status ReadBmpHeader();
    BmpModule::Status ReadBmpHeaderIfh();
    BmpModule::Status ReadBmpHeaderDimensions();
    BmpModule::Status ReadBmpBitfields();
    BmpModule::Status ReadBmpPalette();
    bool GetDataPosition(uint32_t cur_pos);
    void ReadNextScanline();
    BmpModule::Status DecodeRGB();
    BmpModule::Status DecodeRLE8();
    BmpModule::Status DecodeRLE4();
    bool ReadData(uint8_t* destination, uint32_t size);
    void SaveDecodingStatus(DecodeStatus status);
    bool ValidateColorIndex(uint8_t val) const;
};
```

```
bool ValidateFlag() const;
bool SetHeight(int32_t signed_height);

UnownedPtr<CFX_BmpContext> const context_;
std::vector<uint8_t> out_row_buffer_;
std::vector<uint32_t> palette_;
uint32_t header_offset_ = 0;
uint32_t width_ = 0;
uint32_t height_ = 0;
uint32_t compress_flag_ = 0;
int32_t components_ = 0;
size_t src_row_bytes_ = 0;
size_t out_row_bytes_ = 0;
bool img_tb_flag_ = false;
uint16_t bit_counts_ = 0;
uint32_t color_used_ = 0;
int32_t pal_num_ = 0;
int32_t pal_type_ = 0;
uint32_t data_size_ = 0;
uint32_t img_ifh_size_ = 0;
uint32_t row_num_ = 0;
uint32_t col_num_ = 0;
int32_t dpi_x_ = 0;
int32_t dpi_y_ = 0;
uint32_t mask_red_ = 0;
uint32_t mask_green_ = 0;
uint32_t mask_blue_ = 0;
DecodeStatus decode_status_ = DecodeStatus::kHeader;
RetainPtr<CFX_CodecMemory> input_buffer_;
};

} // namespace fxcodec

#endif // CORE_FXCODEC_BMP_CFX_BMPDECOMPRESSOR_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_BMP_FX_BMP_H_
#define CORE_FXCODEC_BMP_FX_BMP_H_

#include <stdint.h>

#pragma pack(1)
struct BmpFileHeader {
    uint16_t bfType;
    uint32_t bfSize;
    uint16_t bfReserved1;
    uint16_t bfReserved2;
    uint32_t bfOffBits;
};

struct BmpCoreHeader {
    uint32_t bcSize;
    uint16_t bcWidth;
    uint16_t bcHeight;
    uint16_t bcPlanes;
    uint16_t bcBitCount;
};

struct BmpInfoHeader {
    uint32_t biSize;
    int32_t biWidth;
    int32_t biHeight;
    uint16_t biPlanes;
    uint16_t biBitCount;
    uint32_t biCompression;
    uint32_t biSizeImage;
    int32_t biXPelsPerMeter;
    int32_t biYPelsPerMeter;
    uint32_t biClrUsed;
    uint32_t biClrImportant;
};
#pragma pack()

static_assert(sizeof(BmpFileHeader) == 14, "BmpFileHeader has wrong size");
static_assert(sizeof(BmpCoreHeader) == 12, "BmpCoreHeader has wrong size");
static_assert(sizeof(BmpInfoHeader) == 40, "BmpInfoHeader has wrong size");

#endif // CORE_FXCODEC_BMP_FX_BMP_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FXCODEC_CFX_CODEC_MEMORY_H_  
#define CORE_FXCODEC_CFX_CODEC_MEMORY_H_
```

```
#include <memory>
```

```
#include "core/fxcrt/fx_memory_wrappers.h"  
#include "core/fxcrt/retain_ptr.h"  
#include "third_party/base/span.h"
```

```
class CFX_CodecMemory final : public Retainable {  
public:  
    template <typename T, typename... Args>  
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);  
  
    pdfium::span<uint8_t> GetSpan() { return {buffer_.get(), size_}; }  
    uint8_t* GetBuffer() { return buffer_.get(); }  
    size_t GetSize() const { return size_; }  
    size_t GetPosition() const { return pos_; }  
    bool IsEOF() const { return pos_ >= size_; }  
    size_t ReadBlock(void* buffer, size_t size);  
  
    // Sets the cursor position to |pos| if possible.  
    bool Seek(size_t pos);  
  
    // Try to change the size of the buffer, keep the old one on failure.  
    bool TryResize(size_t new_buffer_size);  
  
    // Schlep the bytes down the buffer.  
    void Consume(size_t consumed);  
  
private:  
    explicit CFX_CodecMemory(size_t buffer_size);  
    ~CFX_CodecMemory() override;  
  
    std::unique_ptr<uint8_t, FxFreeDeleter> buffer_;  
    size_t size_ = 0;  
    size_t pos_ = 0;  
};  
  
#endif // CORE_FXCODEC_CFX_CODEC_MEMORY_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODER_CODEC_MODULE_IFACE_H_
#define CORE_FXCODER_CODEC_MODULE_IFACE_H_

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

class CFX_CodecMemory;

namespace fxcodec {

class CFX_DIBAttribute;

class ModuleIface {
public:
  class Context {
  public:
    virtual ~Context() = default;
  };

  virtual ~ModuleIface() = default;

  // Returns the number of unprocessed bytes remaining in the input buffer.
  virtual FX_FILESIZE GetAvailInput(Context* pContext) const = 0;

  // Provides a new input buffer to the codec. Returns true on success,
  // setting details about the image extracted from the buffer into |pAttribute|
  // (if provided and the codec is capable providing that information).
  virtual bool Input(Context* pContext,
                    RetainPtr<CFX_CodecMemory> codec_memory,
                    CFX_DIBAttribute* pAttribute) = 0;
};

} // namespace fxcodec

using fxcodec::ModuleIface;

#endif // CORE_FXCODER_CODEC_MODULE_IFACE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_FAX_FAXMODULE_H_
#define CORE_FXCODEC_FAX_FAXMODULE_H_

#include <memory>

#include "build/build_config.h"
#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_system.h"
#include "third_party/base/span.h"

namespace fxcodec {

class ScanlineDecoder;

class FaxModule {
public:
    static std::unique_ptr<ScanlineDecoder> CreateDecoder(
        pdfium::span<const uint8_t> src_span,
        int width,
        int height,
        int K,
        bool EndOfLine,
        bool EncodedByteAlign,
        bool BlackIs1,
        int Columns,
        int Rows);

    // Return the ending bit position.
    static int FaxG4Decode(const uint8_t* src_buf,
                          uint32_t src_size,
                          int starting_bitpos,
                          int width,
                          int height,
                          int pitch,
                          uint8_t* dest_buf);

#ifdef OS_WIN
    static void FaxEncode(const uint8_t* src_buf,
                          int width,
                          int height,
                          int pitch,
                          std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,
                          uint32_t* dest_size);
#endif // defined(OS_WIN)

    FaxModule() = delete;
    FaxModule(const FaxModule&) = delete;
    FaxModule& operator=(const FaxModule&) = delete;
};

} // namespace fxcodec

using FaxModule = fxcodec::FaxModule;

#endif // CORE_FXCODEC_FAX_FAXMODULE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_FLTE_FLATEMODULE_H
#define CORE_FXCODEC_FLTE_FLATEMODULE_H

#include <memory>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_system.h"
#include "third_party/base/span.h"

namespace fxcodec {

class ScanlineDecoder;

class FlateModule {
public:
    static std::unique_ptr<ScanlineDecoder> CreateDecoder(
        pdfium::span<const uint8_t> src_span,
        int width,
        int height,
        int nComps,
        int bpc,
        int predictor,
        int Colors,
        int BitsPerComponent,
        int Columns);

    static uint32_t FlateOrLZWDecode(
        bool bLZW,
        pdfium::span<const uint8_t> src_span,
        bool bEarlyChange,
        int predictor,
        int Colors,
        int BitsPerComponent,
        int Columns,
        uint32_t estimated_size,
        std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,
        uint32_t* dest_size);

    static bool Encode(const uint8_t* src_buf,
                      uint32_t src_size,
                      std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,
                      uint32_t* dest_size);

    FlateModule() = delete;
    FlateModule(const FlateModule&) = delete;
    FlateModule& operator=(const FlateModule&) = delete;
};

} // namespace fxcodec

using FlateModule = fxcodec::FlateModule;

#endif // CORE_FXCODEC_FLTE_FLATEMODULE_H
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_FX_CODEC_DEF_H_
#define CORE_FXCODEC_FX_CODEC_DEF_H_

enum FXCODEC_STATUS {
  FXCODEC_STATUS_ERROR = -1,
  FXCODEC_STATUS_FRAME_READY,
  FXCODEC_STATUS_FRAME_TOBECONTINUE,
  FXCODEC_STATUS_DECODE_READY,
  FXCODEC_STATUS_DECODE_TOBECONTINUE,
  FXCODEC_STATUS_DECODE_FINISH,
#ifdef PDF_ENABLE_XFA
  FXCODEC_STATUS_ERR_MEMORY,
#endif // PDF_ENABLE_XFA
  FXCODEC_STATUS_ERR_READ,
  FXCODEC_STATUS_ERR_FLUSH,
  FXCODEC_STATUS_ERR_FORMAT,
  FXCODEC_STATUS_ERR_PARAMS
};

#ifdef PDF_ENABLE_XFA
enum FXCODEC_IMAGE_TYPE {
  FXCODEC_IMAGE_UNKNOWN = 0,
  FXCODEC_IMAGE_JPG,
#ifdef PDF_ENABLE_XFA_BMP
  FXCODEC_IMAGE_BMP,
#endif // PDF_ENABLE_XFA_BMP
#ifdef PDF_ENABLE_XFA_PNG
  FXCODEC_IMAGE_PNG,
#endif // PDF_ENABLE_XFA_PNG
#ifdef PDF_ENABLE_XFA_GIF
  FXCODEC_IMAGE_GIF,
#endif // PDF_ENABLE_XFA_GIF
#ifdef PDF_ENABLE_XFA_TIFF
  FXCODEC_IMAGE_TIFF,
#endif // PDF_ENABLE_XFA_TIFF
  FXCODEC_IMAGE_MAX
};

enum FXCODEC_RESUNIT {
  FXCODEC_RESUNIT_NONE = 0,
  FXCODEC_RESUNIT_INCH,
  FXCODEC_RESUNIT_CENTIMETER,
  FXCODEC_RESUNIT_METER
};
#endif // PDF_ENABLE_XFA

#endif // CORE_FXCODEC_FX_CODEC_DEF_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_FX_CODEC_H_
#define CORE_FXCODEC_FX_CODEC_H_

#include <map>
#include <memory>
#include <utility>

#include "core/fxcodec/fx_codec_def.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_safe_types.h"
#include "core/fxcrt/fx_string.h"

#ifdef PDF_ENABLE_XFA
#ifdef PDF_ENABLE_XFA_BMP
#include "core/fxcodec/bmp/bmpmodule.h"
#endif // PDF_ENABLE_XFA_BMP

#ifdef PDF_ENABLE_XFA_GIF
#include "core/fxcodec/gif/gifmodule.h"
#endif // PDF_ENABLE_XFA_GIF

#ifdef PDF_ENABLE_XFA_PNG
#include "core/fxcodec/png/pngmodule.h"
#endif // PDF_ENABLE_XFA_PNG

#ifdef PDF_ENABLE_XFA_TIFF
#include "core/fxcodec/tiff/tiffmodule.h"
#endif // PDF_ENABLE_XFA_TIFF
#endif // PDF_ENABLE_XFA

namespace fxcodec {

#ifdef PDF_ENABLE_XFA
class CFX_DIBAttribute {
public:
    CFX_DIBAttribute();
    ~CFX_DIBAttribute();

    int32_t m_nXDPI = -1;
    int32_t m_nYDPI = -1;
    uint16_t m_wDPIUnit = 0;
    std::map<uint32_t, void*> m_Exif;
};
#endif // PDF_ENABLE_XFA

class Jbig2Module;
class JpegModule;
class ProgressiveDecoder;

class ModuleMgr {
public:
    // Per-process singleton managed by callers.
    static void Create();
    static void Destroy();
    static ModuleMgr* GetInstance();

    JpegModule* GetJpegModule() const { return m_pJpegModule.get(); }
};
```

```
Jbig2Module* GetJbig2Module() const { return m_pJbig2Module.get(); }

#ifdef PDF_ENABLE_XFA
    std::unique_ptr<ProgressiveDecoder> CreateProgressiveDecoder();
#endif

#ifdef PDF_ENABLE_XFA_BMP
    BmpModule* GetBmpModule() const { return m_pBmpModule.get(); }
    void SetBmpModule(std::unique_ptr<BmpModule> module) {
        m_pBmpModule = std::move(module);
    }
#endif // PDF_ENABLE_XFA_BMP

#ifdef PDF_ENABLE_XFA_GIF
    GifModule* GetGifModule() const { return m_pGifModule.get(); }
    void SetGifModule(std::unique_ptr<GifModule> module) {
        m_pGifModule = std::move(module);
    }
#endif // PDF_ENABLE_XFA_GIF

#ifdef PDF_ENABLE_XFA_PNG
    PngModule* GetPngModule() const { return m_pPngModule.get(); }
    void SetPngModule(std::unique_ptr<PngModule> module) {
        m_pPngModule = std::move(module);
    }
#endif // PDF_ENABLE_XFA_PNG

#ifdef PDF_ENABLE_XFA_TIFF
    TiffModule* GetTiffModule() const { return m_pTiffModule.get(); }
    void SetTiffModule(std::unique_ptr<TiffModule> module) {
        m_pTiffModule = std::move(module);
    }
#endif // PDF_ENABLE_XFA_TIFF
#endif // PDF_ENABLE_XFA

private:
    ModuleMgr();
    ~ModuleMgr();

    std::unique_ptr<JpegModule> m_pJpegModule;
    std::unique_ptr<Jbig2Module> m_pJbig2Module;

#ifdef PDF_ENABLE_XFA
#ifdef PDF_ENABLE_XFA_BMP
    std::unique_ptr<BmpModule> m_pBmpModule;
#endif // PDF_ENABLE_XFA_BMP

#ifdef PDF_ENABLE_XFA_GIF
    std::unique_ptr<GifModule> m_pGifModule;
#endif // PDF_ENABLE_XFA_GIF

#ifdef PDF_ENABLE_XFA_PNG
    std::unique_ptr<PngModule> m_pPngModule;
#endif // PDF_ENABLE_XFA_PNG

#ifdef PDF_ENABLE_XFA_TIFF
    std::unique_ptr<TiffModule> m_pTiffModule;
#endif // PDF_ENABLE_XFA_TIFF
#endif // PDF_ENABLE_XFA
};

void ReverseRGB(uint8_t* pDestBuf, const uint8_t* pSrcBuf, int pixels);

FX_SAFE_UINT32 CalculatePitch8(uint32_t bpc, uint32_t components, int width);
```

```
FX_SAFE_UINT32 CalculatePitch32(int bpp, int width);
```

```
} // namespace fxcodec
```

```
#ifdef PDF_ENABLE_XFA
```

```
using CFX_DIBAttribute = fxcodec::CFX_DIBAttribute;
```

```
#endif
```

```
#endif // CORE_FXCODEC_FX_CODEC_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FXCODER_GIF_CFX_GIFCONTEXT_H_
#define CORE_FXCODER_GIF_CFX_GIFCONTEXT_H_
```

```
#include <memory>
#include <vector>
```

```
#include "core/fxcodec/gif/cfx_gif.h"
#include "core/fxcodec/gif/cfx_lzwdecompressor.h"
#include "core/fxcodec/gif/gifmodule.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/unowned_ptr.h"
```

```
class CFX_CodecMemory;
```

```
namespace fxcodec {
```

```
class CFX_GifContext : public ModuleIface::Context {
public:
    CFX_GifContext(GifModule* gif_module, GifModule::Delegate* delegate);
    ~CFX_GifContext() override;
```

```
    void RecordCurrentPosition(uint32_t* cur_pos);
    void ReadScanline(int32_t row_num, uint8_t* row_buf);
    bool GetRecordPosition(uint32_t cur_pos,
                           int32_t left,
                           int32_t top,
                           int32_t width,
                           int32_t height,
                           int32_t pal_num,
                           CFX_GifPalette* pal,
                           int32_t delay_time,
                           bool user_input,
                           int32_t trans_index,
                           int32_t disposal_method,
                           bool interlace);
```

```
    CFX_GifDecodeStatus ReadHeader();
    CFX_GifDecodeStatus GetFrame();
    CFX_GifDecodeStatus LoadFrame(int32_t frame_num);
    void SetInputBuffer(RetainPtr<CFX_CodecMemory> codec_memory);
    uint32_t GetAvailInput() const;
    size_t GetFrameNum() const { return images_.size(); }
```

```
    UnownedPtr<GifModule> const gif_module_;
    UnownedPtr<GifModule::Delegate> const delegate_;
    std::vector<CFX_GifPalette> global_palette_;
    uint8_t global_pal_exp_ = 0;
    uint32_t img_row_offset_ = 0;
    uint32_t img_row_avail_size_ = 0;
    int32_t decode_status_ = GIF_D_STATUS_SIG;
    std::unique_ptr<CFX_GifGraphicControlExtension> graphic_control_extension_;
    std::vector<std::unique_ptr<CFX_GifImage>> images_;
    std::unique_ptr<CFX_LZWDecompressor> lzw_decompressor_;
    int width_ = 0;
    int height_ = 0;
    uint8_t bc_index_ = 0;
    uint8_t global_sort_flag_ = 0;
    uint8_t global_color_resolution_ = 0;
```

```
uint8_t img_pass_num_ = 0;
```

```
protected:
```

```
bool ReadAllOrNone(uint8_t* dest, uint32_t size);  
CFX_GifDecodeStatus ReadGifSignature();  
CFX_GifDecodeStatus ReadLogicalScreenDescriptor();
```

```
RetainPtr<CFX_CodecMemory> input_buffer_;
```

```
private:
```

```
void SaveDecodingStatus(int32_t status);  
CFX_GifDecodeStatus DecodeExtension();  
CFX_GifDecodeStatus DecodeImageInfo();  
void DecodingFailureAtTailCleanup(CFX_GifImage* gif_image);  
bool ScanForTerminalMarker();
```

```
};
```

```
} // namespace fxcodec
```

```
#endif // CORE_FXCODER_GIF_CFX_GIFCONTEXT_H
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_GIF_CFX_GIF_H_
#define CORE_FXCODEC_GIF_CFX_GIF_H_

#include <memory>
#include <vector>

#include "core/fxcrt/fx_memory_wrappers.h"

class CFX_GifContext;

extern const char kGifSignature87[];
extern const char kGifSignature89[];

#define GIF_SIG_EXTENSION 0x21
#define GIF_SIG_IMAGE 0x2C
#define GIF_SIG_TRAILER 0x3B
#define GIF_BLOCK_GCE 0xF9
#define GIF_BLOCK_PTE 0x01
#define GIF_BLOCK_CE 0xFE
#define GIF_BLOCK_TERMINAL 0x00
#define GIF_MAX_LZW_EXP 12
#define GIF_MAX_LZW_CODE 4096
#define GIF_D_STATUS_SIG 0x01
#define GIF_D_STATUS_TAIL 0x02
#define GIF_D_STATUS_EXT 0x03
#define GIF_D_STATUS_EXT_CE 0x05
#define GIF_D_STATUS_EXT_GCE 0x06
#define GIF_D_STATUS_EXT_PTE 0x07
#define GIF_D_STATUS_EXT_UNE 0x08
#define GIF_D_STATUS_IMG_INFO 0x09
#define GIF_D_STATUS_IMG_DATA 0x0A

#pragma pack(1)
struct CFX_GifGlobalFlags {
    uint8_t pal_bits : 3;
    uint8_t sort_flag : 1;
    uint8_t color_resolution : 3;
    uint8_t global_pal : 1;
};

struct CFX_GifLocalFlags {
    uint8_t pal_bits : 3;
    uint8_t reserved : 2;
    uint8_t sort_flag : 1;
    uint8_t interlace : 1;
    uint8_t local_pal : 1;
};

struct CFX_GifHeader {
    char signature[6];
};

struct CFX_GifLocalScreenDescriptor {
    uint16_t width;
    uint16_t height;
    CFX_GifGlobalFlags global_flags;
    uint8_t bc_index;
```

```
    uint8_t pixel_aspect;
};

struct CFX_CFX_GifImageInfo {
    uint16_t left;
    uint16_t top;
    uint16_t width;
    uint16_t height;
    CFX_GifLocalFlags local_flags;
};

struct CFX_GifControlExtensionFlags {
    uint8_t transparency : 1;
    uint8_t user_input : 1;
    uint8_t disposal_method : 3;
    uint8_t reserved : 3;
};

struct CFX_GifGraphicControlExtension {
    uint8_t block_size;
    CFX_GifControlExtensionFlags gce_flags;
    uint16_t delay_time;
    uint8_t trans_index;
};

struct CFX_GifPlainTextExtension {
    uint8_t block_size;
    uint16_t grid_left;
    uint16_t grid_top;
    uint16_t grid_width;
    uint16_t grid_height;
    uint8_t char_width;
    uint8_t char_height;
    uint8_t fc_index;
    uint8_t bc_index;
};

struct GifApplicationExtension {
    uint8_t block_size;
    uint8_t app_identify[8];
    uint8_t app_authentication[3];
};

struct CFX_GifPalette {
    uint8_t r;
    uint8_t g;
    uint8_t b;
};
pragma pack()

enum class CFX_GifDecodeStatus {
    Error,
    Success,
    Unfinished,
    InsufficientDestSize, // Only used internally by CGifLZWDecoder::Decode()
};

struct CFX_GifImage {
    CFX_GifImage();
    ~CFX_GifImage();

    std::unique_ptr<CFX_GifGraphicControlExtension> image_GCE;
    std::vector<CFX_GifPalette> local_palettes;
};
```

```
std::vector<uint8_t, FxAllocAllocator<uint8_t>> row_buffer;
CFX_CFX_GifImageInfo image_info;
uint8_t local_palette_exp;
uint8_t code_exp;
uint32_t data_pos;
int32_t row_num;
};

#endif // CORE_FXCODEC_GIF_CFX_GIF_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_GIF_CFX_LZWDECOMPRESSOR_H
#define CORE_FXCODEC_GIF_CFX_LZWDECOMPRESSOR_H

#include <memory>
#include <vector>

#include "core/fxcodec/gif/cfx_gif.h"

class CFX_LZWDecompressor {
public:
    struct CodeEntry {
        uint16_t prefix;
        uint8_t suffix;
    };

    // Returns nullptr on error
    static std::unique_ptr<CFX_LZWDecompressor> Create(uint8_t color_exp,
                                                       uint8_t code_exp);
    ~CFX_LZWDecompressor();

    CFX_GifDecodeStatus Decode(const uint8_t* src_buf,
                              uint32_t src_size,
                              uint8_t* dest_buf,
                              uint32_t* dest_size);

    // Used by unittests, should not be called in production code.
    uint32_t ExtractDataForTest(uint8_t* dest_buf, uint32_t dest_size) {
        return ExtractData(dest_buf, dest_size);
    }

    std::vector<uint8_t>* DecompressedForTest() { return &decompressed_; }
    size_t* DecompressedNextForTest() { return &decompressed_next_; }

private:
    CFX_LZWDecompressor(uint8_t color_exp, uint8_t code_exp);
    void ClearTable();
    void AddCode(uint16_t prefix_code, uint8_t append_char);
    bool DecodeString(uint16_t code);
    uint32_t ExtractData(uint8_t* dest_buf, uint32_t dest_size);

    uint8_t code_size_;
    uint8_t code_size_cur_;
    uint16_t code_color_end_;
    uint16_t code_clear_;
    uint16_t code_end_;
    uint16_t code_next_;
    uint8_t code_first_;
    std::vector<uint8_t> decompressed_;
    size_t decompressed_next_;
    uint16_t code_old_;
    const uint8_t* next_in_;
    uint32_t avail_in_;
    uint8_t bits_left_;
    uint32_t code_store_;
    CodeEntry code_table_[GIF_MAX_LZW_CODE];
};
```

#endif // CORE\_FXCODEC\_GIF\_CFX\_LZWDECOMPRESSOR\_H\_

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_GIF_GIFMODULE_H_
#define CORE_FXCODEC_GIF_GIFMODULE_H_

#include <memory>
#include <utility>

#include "core/fxcodec/codec_module_iface.h"
#include "core/fxcodec/gif/cfx_gif.h"
#include "core/fxcrt/fx_coordinates.h"

namespace fxcodec {

class CFX_DIBAttribute;

class GifModule final : public ModuleIface {
public:
  class Delegate {
public:
    virtual void GifRecordCurrentPosition(uint32_t& cur_pos) = 0;
    virtual bool GifInputRecordPositionBuf(uint32_t rcd_pos,
                                           const FX_RECT& img_rc,
                                           int32_t pal_num,
                                           CFX_GifPalette* pal_ptr,
                                           int32_t delay_time,
                                           bool user_input,
                                           int32_t trans_index,
                                           int32_t disposal_method,
                                           bool interlace) = 0;

    virtual void GifReadScanline(int32_t row_num, uint8_t* row_buf) = 0;
  };

  GifModule();
  ~GifModule() override;

  // ModuleIface:
  FX_FILESIZE GetAvailInput(Context* context) const override;
  bool Input(Context* context,
             RetainPtr<CFX_CodecMemory> codec_memory,
             CFX_DIBAttribute* pAttribute) override;

  std::unique_ptr<Context> Start(Delegate* pDelegate);
  CFX_GifDecodeStatus ReadHeader(Context* context,
                                 int* width,
                                 int* height,
                                 int* pal_num,
                                 CFX_GifPalette** pal_pp,
                                 int* bg_index);
  std::pair<CFX_GifDecodeStatus, size_t> LoadFrameInfo(Context* context);
  CFX_GifDecodeStatus LoadFrame(Context* context, size_t frame_num);
};

} // namespace fxcodec

using GifModule = fxcodec::GifModule;

#endif // CORE_FXCODEC_GIF_GIFMODULE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_ICC_ICCMODULE_H_
#define CORE_FXCODEC_ICC_ICCMODULE_H_

#include <memory>

#include "core/fxcodec/fx_codec_def.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "third_party/base/span.h"

#if defined(USE_SYSTEM_LCMS2)
#include <lcms2.h>
#else
#include "third_party/lcms/include/lcms2.h"
#endif

namespace fxcodec {

class CLcmsCmm {
public:
  CLcmsCmm(cmsHTRANSFORM transform,
           int srcComponents,
           bool bIsLab,
           bool bNormal);
  ~CLcmsCmm();

  cmsHTRANSFORM transform() const { return m_hTransform; }
  int components() const { return m_nSrcComponents; }
  bool IsLab() const { return m_bLab; }
  bool IsNormal() const { return m_bNormal; }

private:
  const cmsHTRANSFORM m_hTransform;
  const int m_nSrcComponents;
  const bool m_bLab;
  const bool m_bNormal;
};

class IccModule {
public:
  static std::unique_ptr<CLcmsCmm> CreateTransformSRGB(
    pdfium::span<const uint8_t> span);
  static void Translate(CLcmsCmm* pTransform,
                       uint32_t nSrcComponents,
                       const float* pSrcValues,
                       float* pDestValues);
  static void TranslateScanline(CLcmsCmm* pTransform,
                                uint8_t* pDest,
                                const uint8_t* pSrc,
                                int pixels);

  IccModule() = delete;
  IccModule(const IccModule&) = delete;
  IccModule& operator=(const IccModule&) = delete;
};

} // namespace fxcodec
```

```
using CLcmsCmm = fxcodec::CLcmsCmm;  
using IccModule = fxcodec::IccModule;  
  
#endif // CORE_FXCODEC_ICC_ICCMODULE_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_ARITHDECODER_H_
#define CORE_FXCODEC_JBIG2_JBIG2_ARITHDECODER_H_

#include <stdint.h>

#include "core/fxcrt/unowned_ptr.h"

class CBig2_BitStream;
struct JBig2ArithQe;

class JBig2ArithCtx {
public:
    struct JBig2ArithQe {
        uint16_t Qe;
        uint8_t NMPS;
        uint8_t NLPS;
        bool bSwitch;
    };

    JBig2ArithCtx();

    int DecodeNLPS(const JBig2ArithQe& qe);
    int DecodeNMPS(const JBig2ArithQe& qe);

    unsigned int MPS() const { return m_MPS ? 1 : 0; }
    unsigned int I() const { return m_I; }

private:
    bool m_MPS = 0;
    unsigned int m_I = 0;
};

class CBig2_ArithDecoder {
public:
    explicit CBig2_ArithDecoder(CBig2_BitStream* pStream);
    ~CBig2_ArithDecoder();

    int Decode(JBig2ArithCtx* pCX);

    bool IsComplete() const { return m_Complete; }

private:
    enum class StreamState : uint8_t {
        kDataAvailable,
        kDecodingFinished,
        kLooping,
    };

    void BYTEIN();
    void ReadValueA();

    bool m_Complete = false;
    StreamState m_State = StreamState::kDataAvailable;
    uint8_t m_B;
    unsigned int m_C;
    unsigned int m_A;
    unsigned int m_CT;
};
```

```
    UnownedPtr<CJBig2_BitStream> const m_pStream;  
};
```

```
#endif // CORE_FXCODEC_JBIG2_JBIG2_ARITHDECODER_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_ARITHINTDECODER_H_
#define CORE_FXCODEC_JBIG2_JBIG2_ARITHINTDECODER_H_

#include <vector>

#include "core/fxcodec/jbig2/JBig2_ArithDecoder.h"
#include "core/fxcrt/fx_system.h"

class CJBIG2_ArithIntDecoder {
public:
  CJBIG2_ArithIntDecoder();
  ~CJBIG2_ArithIntDecoder();

  // Returns true on success, and false when an OOB condition occurs. Many
  // callers can tolerate OOB and do not check the return value.
  bool Decode(CJBIG2_ArithDecoder* pArithDecoder, int* nResult);

private:
  std::vector<JBig2ArithCtx> m_IAX;
};

class CJBIG2_ArithIaidDecoder {
public:
  explicit CJBIG2_ArithIaidDecoder(unsigned char SBSYMCODELENA);
  ~CJBIG2_ArithIaidDecoder();

  void Decode(CJBIG2_ArithDecoder* pArithDecoder, uint32_t* nResult);

private:
  std::vector<JBig2ArithCtx> m_IAID;

  const unsigned char SBSYMCODELEN;
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_ARITHINTDECODER_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_BITSTREAM_H_
#define CORE_FXCODEC_JBIG2_JBIG2_BITSTREAM_H_

#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/span.h"

class CBig2_BitStream {
public:
  CBig2_BitStream(pdfium::span<const uint8_t> pSrcStream, uint32_t dwObjNum);
  CBig2_BitStream(const CBig2_BitStream&) = delete;
  CBig2_BitStream& operator=(const CBig2_BitStream&) = delete;
  ~CBig2_BitStream();

  // TODO(thestig): readFoo() should return bool.
  int32_t readNBits(uint32_t dwBits, uint32_t* dwResult);
  int32_t readNBits(uint32_t dwBits, int32_t* nResult);
  int32_t read1Bit(uint32_t* dwResult);
  int32_t read1Bit(bool* bResult);
  int32_t read1Byte(uint8_t* cResult);
  int32_t readInteger(uint32_t* dwResult);
  int32_t readShortInteger(uint16_t* wResult);
  void alignByte();
  uint8_t getCurByte() const;
  void incByteIdx();
  uint8_t getCurByte_arith() const;
  uint8_t getNextByte_arith() const;
  uint32_t getOffset() const;
  void setOffset(uint32_t dwOffset);
  uint32_t getBitPos() const;
  void setBitPos(uint32_t dwBitPos);
  const uint8_t* getBuf() const;
  uint32_t getLength() const { return m_Span.size(); }
  const uint8_t* getPointer() const;
  void offset(uint32_t dwOffset);
  uint32_t getByteLeft() const;
  uint32_t getObjNum() const;
  bool IsInBounds() const;

private:
  void AdvanceBit();
  uint32_t LengthInBits() const;

  const pdfium::span<const uint8_t> m_Span;
  uint32_t m_dwByteIdx = 0;
  uint32_t m_dwBitIdx = 0;
  const uint32_t m_dwObjNum;
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_BITSTREAM_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODER_JBIG2_JBIG2_CONTEXT_H_
#define CORE_FXCODER_JBIG2_JBIG2_CONTEXT_H_

#include <list>
#include <memory>
#include <utility>
#include <vector>

#include "core/fxcodec/fx_codec_def.h"
#include "core/fxcodec/jbig2/JBig2_Page.h"
#include "core/fxcodec/jbig2/JBig2_Segment.h"
#include "core/fxcrt/fx_safe_types.h"
#include "third_party/base/span.h"

class CJBig2_ArithDecoder;
class CJBig2_GRDProc;
class CPDF_StreamAcc;
class PauseIndicatorIface;

// Cache is keyed by the ObjNum of a stream and an index within the stream.
using CJBig2_CacheKey = std::pair<uint32_t, uint32_t>;
using CJBig2_CachePair =
    std::pair<CJBig2_CacheKey, std::unique_ptr<CJBig2_SymbolDict>>;

#define JBIG2_MIN_SEGMENT_SIZE 11

enum class JBig2_Result { kSuccess, kFailure, kEndReached };

class CJBig2_Context {
public:
    static std::unique_ptr<CJBig2_Context> Create(
        pdfium::span<const uint8_t> pGlobalSpan,
        uint32_t dwGlobalObjNum,
        pdfium::span<const uint8_t> pSrcSpan,
        uint32_t dwSrcObjNum,
        std::list<CJBig2_CachePair>* pSymbolDictCache);

    ~CJBig2_Context();

    static bool HuffmanAssignCode(JBig2HuffmanCode* SBSYMCODES, uint32_t NTEMP);

    bool GetFirstPage(uint8_t* pBuf,
        int32_t width,
        int32_t height,
        int32_t stride,
        PauseIndicatorIface* pPause);

    bool Continue(PauseIndicatorIface* pPause);
    FXCODEC_STATUS GetProcessingStatus() const { return m_ProcessingStatus; }

private:
    CJBig2_Context(pdfium::span<const uint8_t> pSrcSpan,
        uint32_t dwObjNum,
        std::list<CJBig2_CachePair>* pSymbolDictCache,
        bool bIsGlobal);

    JBig2_Result DecodeSequential(PauseIndicatorIface* pPause);

```

```
CJBig2_Segment* FindSegmentByNumber(uint32_t dwNumber);
CJBig2_Segment* FindReferredTableSegmentByIndex(CJBig2_Segment* pSegment,
                                                int32_t nIndex);

JBig2_Result ParseSegmentHeader(CJBig2_Segment* pSegment);
JBig2_Result ParseSegmentData(CJBig2_Segment* pSegment,
                              PauseIndicatorIface* pPause);
JBig2_Result ProcessingParseSegmentData(CJBig2_Segment* pSegment,
                                        PauseIndicatorIface* pPause);

JBig2_Result ParseSymbolDict(CJBig2_Segment* pSegment);
JBig2_Result ParseTextRegion(CJBig2_Segment* pSegment);
JBig2_Result ParsePatternDict(CJBig2_Segment* pSegment,
                              PauseIndicatorIface* pPause);
JBig2_Result ParseHalftoneRegion(CJBig2_Segment* pSegment,
                                 PauseIndicatorIface* pPause);
JBig2_Result ParseGenericRegion(CJBig2_Segment* pSegment,
                                PauseIndicatorIface* pPause);
JBig2_Result ParseGenericRefinementRegion(CJBig2_Segment* pSegment);
JBig2_Result ParseTable(CJBig2_Segment* pSegment);
JBig2_Result ParseRegionInfo(JBig2RegionInfo* pRI);

std::vector<JBig2HuffmanCode> DecodeSymbolIDHuffmanTable(uint32_t SBNUMSYMS);
const CJBig2_HuffmanTable* GetHuffmanTable(size_t idx);

std::unique_ptr<CJBig2_Context> m_pGlobalContext;
std::unique_ptr<CJBig2_BitStream> m_pStream;
std::vector<std::unique_ptr<CJBig2_Segment>> m_SegmentList;
std::vector<std::unique_ptr<JBig2PageInfo>> m_PageInfoList;
std::unique_ptr<CJBig2_Image> m_pPage;
std::vector<std::unique_ptr<CJBig2_HuffmanTable>> m_HuffmanTables;
const bool m_bIsGlobal;
bool m_bInPage = false;
bool m_bBufSpecified = false;
int32_t m_PauseStep = 10;
FXCODEC_STATUS m_ProcessingStatus = FXCODEC_STATUS_FRAME_READY;
std::vector<JBig2ArithCtx> m_gbContext;
std::unique_ptr<CJBig2_ArithDecoder> m_pArithDecoder;
std::unique_ptr<CJBig2_GRDProc> m_pGRD;
std::unique_ptr<CJBig2_Segment> m_pSegment;
FX_SAFE_UINT32 m_dwOffset = 0;
JBig2RegionInfo m_ri;
std::list<CJBig2_CachePair>* const m_pSymbolDictCache;
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_CONTEXT_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_DEFINE_H_
#define CORE_FXCODEC_JBIG2_JBIG2_DEFINE_H_

#include <stdint.h>

#define JBIG2_OOB 1

struct JBig2RegionInfo {
  int32_t width;
  int32_t height;
  int32_t x;
  int32_t y;
  uint8_t flags;
};

struct JBig2HuffmanCode {
  int32_t codelen;
  int32_t code;
};

#define JBIG2_MAX_REFERRED_SEGMENT_COUNT 64
#define JBIG2_MAX_EXPORT_SYMBOLS 65535
#define JBIG2_MAX_NEW_SYMBOLS 65535
#define JBIG2_MAX_PATTERN_INDEX 65535
#define JBIG2_MAX_IMAGE_SIZE 65535

#endif // CORE_FXCODEC_JBIG2_JBIG2_DEFINE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.
//
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_DOCUMENTCONTEXT_H_
#define CORE_FXCODEC_JBIG2_JBIG2_DOCUMENTCONTEXT_H_

#include <list>
#include <memory>
#include <utility>

class CBig2_SymbolDict;

using CBig2_CacheKey = std::pair<uint32_t, uint32_t>;
using CBig2_CachePair =
    std::pair<CBig2_CacheKey, std::unique_ptr<CBig2_SymbolDict>>;

// Holds per-document JBig2 related data.
class JBig2_DocumentContext {
public:
    JBig2_DocumentContext();
    ~JBig2_DocumentContext();

    std::list<CBig2_CachePair>* GetSymbolDictCache() {
        return &m_SymbolDictCache;
    }

private:
    std::list<CBig2_CachePair> m_SymbolDictCache;
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_DOCUMENTCONTEXT_H_
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FXCODEC_JBIG2_JBIG2_GRDPROC_H
```

```
#define CORE_FXCODEC_JBIG2_JBIG2_GRDPROC_H
```

```
#include <memory>
```

```
#include "core/fxcodec/fx_codec_def.h"
```

```
#include "core/fxcrt/fx_coordinates.h"
```

```
#include "core/fxcrt/fx_system.h"
```

```
#include "core/fxcrt/unowned_ptr.h"
```

```
class CBig2_ArithDecoder;
```

```
class CBig2_BitStream;
```

```
class CBig2_Image;
```

```
class JBig2ArithCtx;
```

```
class PauseIndicatorIface;
```

```
class CBig2_GRDProc {
```

```
  public:
```

```
    class ProgressiveArithDecodeState {
```

```
      public:
```

```
        ProgressiveArithDecodeState();
```

```
        ~ProgressiveArithDecodeState();
```

```
        std::unique_ptr<CBig2_Image>* pImage;
```

```
        UnownedPtr<CBig2_ArithDecoder> pArithDecoder;
```

```
        UnownedPtr<JBig2ArithCtx> gbContext;
```

```
        UnownedPtr<PauseIndicatorIface> pPause;
```

```
    };
```

```
  CBig2_GRDProc();
```

```
  ~CBig2_GRDProc();
```

```
  std::unique_ptr<CBig2_Image> DecodeArith(CBig2_ArithDecoder* pArithDecoder,  
                                           JBig2ArithCtx* gbContext);
```

```
  FXCODEC_STATUS StartDecodeArith(ProgressiveArithDecodeState* pState);
```

```
  FXCODEC_STATUS StartDecodeMMR(std::unique_ptr<CBig2_Image>* pImage,  
                                 CBig2_BitStream* pStream);
```

```
  FXCODEC_STATUS ContinueDecode(ProgressiveArithDecodeState* pState);
```

```
  const FX_RECT& GetReplaceRect() const { return m_ReplaceRect; }
```

```
  bool MMR;
```

```
  bool TPGDON;
```

```
  bool USESKIP;
```

```
  uint8_t GBTEMPLATE;
```

```
  uint32_t GBW;
```

```
  uint32_t GBH;
```

```
  UnownedPtr<CBig2_Image> SKIP;
```

```
  int8_t GBAT[8];
```

```
  private:
```

```
    bool UseTemplate0Opt3() const;
```

```
    bool UseTemplate1Opt3() const;
```

```
    bool UseTemplate23Opt3() const;
```

```
  FXCODEC_STATUS ProgressiveDecodeArith(ProgressiveArithDecodeState* pState);
```

```
  FXCODEC_STATUS ProgressiveDecodeArithTemplate0Opt3(
```

```
    ProgressiveArithDecodeState* pState);
FXCODEC_STATUS ProgressiveDecodeArithTemplate0Unopt(
    ProgressiveArithDecodeState* pState);
FXCODEC_STATUS ProgressiveDecodeArithTemplate1Opt3(
    ProgressiveArithDecodeState* pState);
FXCODEC_STATUS ProgressiveDecodeArithTemplate1Unopt(
    ProgressiveArithDecodeState* pState);
FXCODEC_STATUS ProgressiveDecodeArithTemplate2Opt3(
    ProgressiveArithDecodeState* pState);
FXCODEC_STATUS ProgressiveDecodeArithTemplate2Unopt(
    ProgressiveArithDecodeState* pState);
FXCODEC_STATUS ProgressiveDecodeArithTemplate3Opt3(
    ProgressiveArithDecodeState* pState);
FXCODEC_STATUS ProgressiveDecodeArithTemplate3Unopt(
    ProgressiveArithDecodeState* pState);

std::unique_ptr<CJBig2_Image> DecodeArithOpt3(
    CJBig2_ArithDecoder* pArithDecoder,
    JBig2ArithCtx* gbContext,
    int OPT);
std::unique_ptr<CJBig2_Image> DecodeArithTemplateUnopt(
    CJBig2_ArithDecoder* pArithDecoder,
    JBig2ArithCtx* gbContext,
    int UNOPT);
std::unique_ptr<CJBig2_Image> DecodeArithTemplate3Opt3(
    CJBig2_ArithDecoder* pArithDecoder,
    JBig2ArithCtx* gbContext);
std::unique_ptr<CJBig2_Image> DecodeArithTemplate3Unopt(
    CJBig2_ArithDecoder* pArithDecoder,
    JBig2ArithCtx* gbContext);

uint32_t m_loopIndex = 0;
uint8_t* m_pLine = nullptr;
FXCODEC_STATUS m_ProssiveStatus;
uint16_t m_DecodeType = 0;
int m_LTP = 0;
FX_RECT m_ReplaceRect;
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_GRDPROC_H_
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODER_JBIG2_JBIG2_GRRDPROC_H_
#define CORE_FXCODER_JBIG2_JBIG2_GRRDPROC_H_

#include <memory>

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"

class CBig2_ArithDecoder;
class CBig2_Image;
class JBig2ArithCtx;

class CBig2_GRRDProc {
public:
  CBig2_GRRDProc();
  ~CBig2_GRRDProc();

  std::unique_ptr<CBig2_Image> Decode(CBig2_ArithDecoder* pArithDecoder,
                                     JBig2ArithCtx* grContext);

  bool GRTEMPLATE;
  bool TPGRON;
  uint32_t GRW;
  uint32_t GRH;
  int32_t GRREFERENCEDX;
  int32_t GRREFERENCEDY;
  UnownedPtr<CBig2_Image> GRREFERENCE;
  int8_t GRAT[4];

private:
  std::unique_ptr<CBig2_Image> DecodeTemplate0Unopt (
    CBig2_ArithDecoder* pArithDecoder,
    JBig2ArithCtx* grContext);
  uint32_t DecodeTemplate0UnoptCalculateContext(const CBig2_Image& GRREG,
                                               const uint32_t* lines,
                                               uint32_t w,
                                               uint32_t h) const;
  void DecodeTemplate0UnoptSetPixel(CBig2_Image* GRREG,
                                    uint32_t* lines,
                                    uint32_t w,
                                    uint32_t h,
                                    int bVal);

  std::unique_ptr<CBig2_Image> DecodeTemplate0Opt (
    CBig2_ArithDecoder* pArithDecoder,
    JBig2ArithCtx* grContext);

  std::unique_ptr<CBig2_Image> DecodeTemplate1Unopt (
    CBig2_ArithDecoder* pArithDecoder,
    JBig2ArithCtx* grContext);

  std::unique_ptr<CBig2_Image> DecodeTemplate1Opt (
    CBig2_ArithDecoder* pArithDecoder,
    JBig2ArithCtx* grContext);
};

#endif // CORE_FXCODER_JBIG2_JBIG2_GRRDPROC_H_
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_HTRDPROC_H_
#define CORE_FXCODEC_JBIG2_JBIG2_HTRDPROC_H_

#include <memory>
#include <vector>

#include "core/fxcodec/jbig2/JBig2_Image.h"
#include "core/fxcrt/fx_system.h"

class CBig2_ArithDecoder;
class CBig2_BitStream;
class JBig2ArithCtx;
class PauseIndicatorIface;

class CBig2_HTRDProc {
public:
    std::unique_ptr<CBig2_Image> DecodeArith(CBig2_ArithDecoder* pArithDecoder,
                                           JBig2ArithCtx* gbContext,
                                           PauseIndicatorIface* pPause);

    std::unique_ptr<CBig2_Image> DecodeMMR(CBig2_BitStream* pStream);

public:
    uint32_t HBW;
    uint32_t HBH;
    bool HMMR;
    uint8_t HTEMPLATE;
    uint32_t HNUMPATS;
    const std::vector<std::unique_ptr<CBig2_Image>>* HPATS;
    bool HDEFPIXEL;
    JBig2ComposeOp HCOMBOP;
    bool HENABLESKIP;
    uint32_t HGW;
    uint32_t HGH;
    int32_t HGX;
    int32_t HGY;
    uint16_t HRX;
    uint16_t HRY;
    uint8_t HPW;
    uint8_t HPH;

private:
    std::unique_ptr<CBig2_Image> DecodeImage(
        const std::vector<std::unique_ptr<CBig2_Image>>& GSPLANES);
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_HTRDPROC_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODER_JBIG2_JBIG2_HUFFMANDECODER_H_
#define CORE_FXCODER_JBIG2_JBIG2_HUFFMANDECODER_H_

#include "core/fxcodec/jbig2/JBig2_BitStream.h"
#include "core/fxcodec/jbig2/JBig2_HuffmanTable.h"
#include "core/fxcrt/unowned_ptr.h"

class CBig2_HuffmanDecoder {
public:
  explicit CBig2_HuffmanDecoder(CBig2_BitStream* pStream);
  ~CBig2_HuffmanDecoder();

  int DecodeAValue(const CBig2_HuffmanTable* pTable, int* nResult);

private:
  UnownedPtr<CBig2_BitStream> const m_pStream;
};

#endif // CORE_FXCODER_JBIG2_JBIG2_HUFFMANDECODER_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_HUFFMANTABLE_H_
#define CORE_FXCODEC_JBIG2_JBIG2_HUFFMANTABLE_H_

#include <vector>

#include "core/fxcodec/jbig2/JBig2_Define.h"
#include "core/fxcrt/fx_system.h"

class CJBIG2_BitStream;

class CJBIG2_HuffmanTable {
public:
    explicit CJBIG2_HuffmanTable(size_t idx);
    explicit CJBIG2_HuffmanTable(CJBIG2_BitStream* pStream);
    ~CJBIG2_HuffmanTable();

    bool IsHTOOB() const { return HTOOB; }
    uint32_t Size() const { return NTEMP; }
    const std::vector<JBig2HuffmanCode>& GetCODES() const { return CODES; }
    const std::vector<int>& GetRANGELEN() const { return RANGELEN; }
    const std::vector<int>& GetRANGELOW() const { return RANGELOW; }
    bool IsOK() const { return m_bOK; }

    constexpr static size_t kNumHuffmanTables = 16;

private:
    bool ParseFromStandardTable(size_t table_idx);
    bool ParseFromCodedBuffer(CJBIG2_BitStream* pStream);
    void ExtendBuffers(bool increment);

    bool m_bOK;
    bool HTOOB;
    uint32_t NTEMP;
    std::vector<JBig2HuffmanCode> CODES;
    std::vector<int> RANGELEN;
    std::vector<int> RANGELOW;
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_HUFFMANTABLE_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODER_JBIG2_JBIG2_IMAGE_H
#define CORE_FXCODER_JBIG2_JBIG2_IMAGE_H

#include <memory>

#include "core/fxcodec/jbig2/JBig2_Define.h"
#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/maybe_owned.h"

struct FX_RECT;

enum JBig2ComposeOp {
    JBIG2_COMPOSE_OR = 0,
    JBIG2_COMPOSE_AND = 1,
    JBIG2_COMPOSE_XOR = 2,
    JBIG2_COMPOSE_XNOR = 3,
    JBIG2_COMPOSE_REPLACE = 4
};

class CJBig2_Image {
public:
    CJBig2_Image(int32_t w, int32_t h);
    CJBig2_Image(int32_t w, int32_t h, int32_t stride, uint8_t* pBuf);
    CJBig2_Image(const CJBig2_Image& other);
    ~CJBig2_Image();

    static bool IsValidImageSize(int32_t w, int32_t h);

    int32_t width() const { return m_nWidth; }
    int32_t height() const { return m_nHeight; }
    int32_t stride() const { return m_nStride; }

    uint8_t* data() const { return m_pData.Get(); }

    int GetPixel(int32_t x, int32_t y) const;
    void SetPixel(int32_t x, int32_t y, int v);

    uint8_t* GetLineUnsafe(int32_t y) const { return data() + y * m_nStride; }
    uint8_t* GetLine(int32_t y) const {
        return (y >= 0 && y < m_nHeight) ? GetLineUnsafe(y) : nullptr;
    }

    void CopyLine(int32_t hTo, int32_t hFrom);
    void Fill(bool v);

    bool ComposeFrom(int32_t x, int32_t y, CJBig2_Image* pSrc, JBig2ComposeOp op);
    bool ComposeFromWithRect(int32_t x,
                             int32_t y,
                             CJBig2_Image* pSrc,
                             const FX_RECT& rtSrc,
                             JBig2ComposeOp op);

    std::unique_ptr<CJBig2_Image> SubImage(int32_t x,
                                         int32_t y,
                                         int32_t w,
                                         int32_t h);

    void Expand(int32_t h, bool v);
};
```

```
bool ComposeTo(CJBig2_Image* pDst, int32_t x, int32_t y, JBig2ComposeOp op);  
bool ComposeToWithRect(CJBig2_Image* pDst,  
                        int32_t x,  
                        int32_t y,  
                        const FX_RECT& rtSrc,  
                        JBig2ComposeOp op);
```

**private:**

```
void SubImageFast(int32_t x,  
                  int32_t y,  
                  int32_t w,  
                  int32_t h,  
                  CJBig2_Image* pImage);  
void SubImageSlow(int32_t x,  
                  int32_t y,  
                  int32_t w,  
                  int32_t h,  
                  CJBig2_Image* pImage);  
bool ComposeToInternal(CJBig2_Image* pDst,  
                       int32_t x,  
                       int32_t y,  
                       JBig2ComposeOp op,  
                       const FX_RECT& rtSrc);
```

```
MaybeOwned<uint8_t, FxFreeDeleter> m_pData;  
int32_t m_nWidth = 0; // 1-bit pixels  
int32_t m_nHeight = 0; // lines  
int32_t m_nStride = 0; // bytes, must be multiple of 4.  
};
```

```
#endif // CORE_FXCODER_JBIG2_JBIG2_IMAGE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2MODULE_H
#define CORE_FXCODEC_JBIG2_JBIG2MODULE_H

#include <memory>

#include "core/fxcodec/fx_codec_def.h"
#include "third_party/base/span.h"

class CBig2_Context;
class CBig2_Image;
class JBig2_DocumentContext;
class PauseIndicatorIface;

namespace fxcodec {

class Jbig2Context {
public:
    Jbig2Context();
    ~Jbig2Context();

    uint32_t m_width = 0;
    uint32_t m_height = 0;
    uint32_t m_nGlobalObjNum = 0;
    uint32_t m_nSrcObjNum = 0;
    pdfium::span<const uint8_t> m_pGlobalSpan;
    pdfium::span<const uint8_t> m_pSrcSpan;
    uint8_t* m_dest_buf = nullptr;
    uint32_t m_dest_pitch = 0;
    std::unique_ptr<CBig2_Context> m_pContext;
};

class Jbig2Module {
public:
    Jbig2Module();
    ~Jbig2Module();

    FXCODEC_STATUS StartDecode(
        Jbig2Context* pJbig2Context,
        std::unique_ptr<JBig2_DocumentContext>* pContextHolder,
        uint32_t width,
        uint32_t height,
        pdfium::span<const uint8_t> src_span,
        uint32_t src_objnum,
        pdfium::span<const uint8_t> global_span,
        uint32_t global_objnum,
        uint8_t* dest_buf,
        uint32_t dest_pitch,
        PauseIndicatorIface* pPause);

    FXCODEC_STATUS ContinueDecode(Jbig2Context* pJbig2Context,
        PauseIndicatorIface* pPause);

private:
    FXCODEC_STATUS Decode(Jbig2Context* pJbig2Context, bool decode_success);
};

} // namespace fxcodec
```

```
using Jbig2Context = fxcodec::Jbig2Context;  
using Jbig2Module = fxcodec::Jbig2Module;  
  
#endif // CORE_FXCODER_JBIG2_JBIG2MODULE_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_PAGE_H_
#define CORE_FXCODEC_JBIG2_JBIG2_PAGE_H_

#include "core/fxcrt/fx_system.h"

struct JBig2PageInfo {
  uint32_t m_dwWidth;
  uint32_t m_dwHeight;
  uint32_t m_dwResolutionX;
  uint32_t m_dwResolutionY;
  uint8_t m_cFlags;
  bool m_bIsStriped;
  uint16_t m_wMaxStripeSize;
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_PAGE_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODER_JBIG2_JBIG2_PATTERNDICT_H_
#define CORE_FXCODER_JBIG2_JBIG2_PATTERNDICT_H_

#include <memory>
#include <vector>

#include "core/fxcodec/jbig2/JBig2_Define.h"
#include "core/fxcodec/jbig2/JBig2_Image.h"

class CJBIG2_PatternDict {
public:
  explicit CJBIG2_PatternDict(uint32_t dict_size);
  ~CJBIG2_PatternDict();

  uint32_t NUMPATS;
  std::vector<std::unique_ptr<CJBIG2_Image>> HDPATS;
};

#endif // CORE_FXCODER_JBIG2_JBIG2_PATTERNDICT_H_
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_PDDPROC_H_
#define CORE_FXCODEC_JBIG2_JBIG2_PDDPROC_H_

#include <memory>

#include "core/fxcrt/fx_system.h"

class CBig2_ArithDecoder;
class CBig2_BitStream;
class CBig2_GRDProc;
class CBig2_PatternDict;
class JBig2ArithCtx;
class PauseIndicatorIface;

class CBig2_PDDProc {
public:
    std::unique_ptr<CBig2_PatternDict> DecodeArith(
        CBig2_ArithDecoder* pArithDecoder,
        JBig2ArithCtx* gbContext,
        PauseIndicatorIface* pPause);

    std::unique_ptr<CBig2_PatternDict> DecodeMMR(CBig2_BitStream* pStream);

    bool HDMMR;
    uint8_t HDPW;
    uint8_t HDPH;
    uint32_t GRAYMAX;
    uint8_t HDTEMPLATE;

private:
    std::unique_ptr<CBig2_GRDProc> CreateGRDProc();
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_PDDPROC_H_
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_SDDPROC_H_
#define CORE_FXCODEC_JBIG2_JBIG2_SDDPROC_H_

#include <memory>
#include <vector>

#include "core/fxcodec/jbig2/JBig2_ArithDecoder.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"

class CBig2_BitStream;
class CBig2_HuffmanTable;
class CBig2_Image;
class CBig2_SymbolDict;

class CBig2_SDDProc {
public:
  CBig2_SDDProc();
  ~CBig2_SDDProc();

  std::unique_ptr<CBig2_SymbolDict> DecodeArith(
    CBig2_ArithDecoder* pArithDecoder,
    std::vector<JBig2ArithCtx>* gbContext,
    std::vector<JBig2ArithCtx>* grContext);

  std::unique_ptr<CBig2_SymbolDict> DecodeHuffman(
    CBig2_BitStream* pStream,
    std::vector<JBig2ArithCtx>* gbContext,
    std::vector<JBig2ArithCtx>* grContext);

  bool SDHUFF;
  bool SDREFAGG;
  bool SDRTEMPLATE;
  uint8_t SDTEMPLATE;
  uint32_t SDNUMINSYMS;
  uint32_t SDNUMNEWSYMS;
  uint32_t SDNUMEXSYMS;
  CBig2_Image** SDINSYMS;
  UnownedPtr<const CBig2_HuffmanTable> SDHUFFDH;
  UnownedPtr<const CBig2_HuffmanTable> SDHUFFDW;
  UnownedPtr<const CBig2_HuffmanTable> SDHUFFBMSIZE;
  UnownedPtr<const CBig2_HuffmanTable> SDHUFFAGGINST;
  int8_t SDAT[8];
  int8_t SDRAT[4];
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_SDDPROC_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODER_JBIG2_JBIG2_SEGMENT_H_
#define CORE_FXCODER_JBIG2_JBIG2_SEGMENT_H_

#include <memory>
#include <vector>

#include "core/fxcodec/jbig2/JBig2_Define.h"
#include "core/fxcodec/jbig2/JBig2_HuffmanTable.h"
#include "core/fxcodec/jbig2/JBig2_PatternDict.h"
#include "core/fxcodec/jbig2/JBig2_SymbolDict.h"

enum JBig2_SegmentState {
    JBIG2_SEGMENT_HEADER_UNPARSED,
    JBIG2_SEGMENT_DATA_UNPARSED,
    JBIG2_SEGMENT_PARSE_COMPLETE,
    JBIG2_SEGMENT_PAUSED,
    JBIG2_SEGMENT_ERROR
};

enum JBig2_ResultType {
    JBIG2_VOID_POINTER = 0,
    JBIG2_IMAGE_POINTER,
    JBIG2_SYMBOL_DICT_POINTER,
    JBIG2_PATTERN_DICT_POINTER,
    JBIG2_HUFFMAN_TABLE_POINTER
};

class CJBig2_Segment {
public:
    CJBig2_Segment();
    ~CJBig2_Segment();

    uint32_t m_dwNumber;
    union {
        struct {
            uint8_t type : 6;
            uint8_t page_association_size : 1;
            uint8_t deferred_non_retain : 1;
        } s;
        uint8_t c;
    } m_cFlags;
    int32_t m_nReferred_to_segment_count;
    std::vector<uint32_t> m_Referred_to_segment_numbers;
    uint32_t m_dwPage_association;
    uint32_t m_dwData_length;

    uint32_t m_dwHeader_Length;
    uint32_t m_dwObjNum;
    uint32_t m_dwDataOffset;
    JBig2_SegmentState m_State;
    JBig2_ResultType m_nResultType;
    std::unique_ptr<CJBig2_SymbolDict> m_SymbolDict;
    std::unique_ptr<CJBig2_PatternDict> m_PatternDict;
    std::unique_ptr<CJBig2_Image> m_Image;
    std::unique_ptr<CJBig2_HuffmanTable> m_HuffmanTable;
};
```

#endif // CORE\_FXCODEC\_JBIG2\_JBIG2\_SEGMENT\_H\_

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JBIG2_JBIG2_SYMBOLDICT_H_
#define CORE_FXCODEC_JBIG2_JBIG2_SYMBOLDICT_H_

#include <memory>
#include <utility>
#include <vector>

#include "core/fxcodec/jbig2/JBig2_ArithDecoder.h"

class CJBIG2_Image;

class CJBIG2_SymbolDict {
public:
  CJBIG2_SymbolDict();
  ~CJBIG2_SymbolDict();

  std::unique_ptr<CJBIG2_SymbolDict> DeepCopy() const;

  void AddImage(std::unique_ptr<CJBIG2_Image> image) {
    m_SDEXSyms.push_back(std::move(image));
  }

  size_t NumImages() const { return m_SDEXSyms.size(); }
  CJBIG2_Image* GetImage(size_t index) const { return m_SDEXSyms[index].get(); }

  const std::vector<JBig2ArithCtx>& GbContext() const { return m_gbContext; }
  const std::vector<JBig2ArithCtx>& GrContext() const { return m_grContext; }

  void SetGbContext(std::vector<JBig2ArithCtx> gbContext) {
    m_gbContext = std::move(gbContext);
  }
  void SetGrContext(std::vector<JBig2ArithCtx> grContext) {
    m_grContext = std::move(grContext);
  }

private:
  std::vector<JBig2ArithCtx> m_gbContext;
  std::vector<JBig2ArithCtx> m_grContext;
  std::vector<std::unique_ptr<CJBIG2_Image>> m_SDEXSyms;
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_SYMBOLDICT_H_
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODER_JBIG2_JBIG2_TRDPROC_H_
#define CORE_FXCODER_JBIG2_JBIG2_TRDPROC_H_

#include <memory>
#include <vector>

#include "core/fxcodec/jbig2/JBig2_Image.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"

class CBig2_ArithDecoder;
class CBig2_ArithIaidDecoder;
class CBig2_ArithIntDecoder;
class CBig2_BitStream;
class CBig2_HuffmanTable;
class JBig2ArithCtx;
struct JBig2HuffmanCode;

struct JBig2IntDecoderState {
    JBig2IntDecoderState();
    ~JBig2IntDecoderState();

    UnownedPtr<CBig2_ArithIntDecoder> IADT;
    UnownedPtr<CBig2_ArithIntDecoder> IAFS;
    UnownedPtr<CBig2_ArithIntDecoder> IADS;
    UnownedPtr<CBig2_ArithIntDecoder> IAIT;
    UnownedPtr<CBig2_ArithIntDecoder> IARI;
    UnownedPtr<CBig2_ArithIntDecoder> IARDW;
    UnownedPtr<CBig2_ArithIntDecoder> IARDH;
    UnownedPtr<CBig2_ArithIntDecoder> IARDX;
    UnownedPtr<CBig2_ArithIntDecoder> IARDY;
    UnownedPtr<CBig2_ArithIaidDecoder> IAID;
};

enum JBig2Corner {
    JBIG2_CORNER_BOTTOMLEFT = 0,
    JBIG2_CORNER_TOPLEFT = 1,
    JBIG2_CORNER_BOTTOMRIGHT = 2,
    JBIG2_CORNER_TOPRIGHT = 3
};

class CBig2_TRDProc {
public:
    CBig2_TRDProc();
    ~CBig2_TRDProc();

    std::unique_ptr<CBig2_Image> DecodeHuffman(CBig2_BitStream* pStream,
                                              JBig2ArithCtx* grContext);

    std::unique_ptr<CBig2_Image> DecodeArith(CBig2_ArithDecoder* pArithDecoder,
                                           JBig2ArithCtx* grContext,
                                           JBig2IntDecoderState* pIDS);

    bool SBHUFF;
    bool SBREFINE;
    bool SBRTEMPLATE;
    bool TRANSPOSED;
};
```

```
bool SBDEFPIXEL;
int8_t SBDSOFFSET;
uint8_t SBSYMCODELEN;
uint32_t SBW;
uint32_t SBH;
uint32_t SBNUMINSTANCES;
uint32_t SBSTRIPS;
uint32_t SBNUMSYMS;
std::vector<JBig2HuffmanCode> SBSYMCODES;
CJBig2_Image** SBSYMS;
JBig2ComposeOp SBCOMBOP;
JBig2Corner REFCORNER;
UnownedPtr<const CJBig2_HuffmanTable> SBHUFFFS;
UnownedPtr<const CJBig2_HuffmanTable> SBHUFFDS;
UnownedPtr<const CJBig2_HuffmanTable> SBHUFFDT;
UnownedPtr<const CJBig2_HuffmanTable> SBHUFFRDW;
UnownedPtr<const CJBig2_HuffmanTable> SBHUFFRDH;
UnownedPtr<const CJBig2_HuffmanTable> SBHUFFRDY;
UnownedPtr<const CJBig2_HuffmanTable> SBHUFFRDX;
UnownedPtr<const CJBig2_HuffmanTable> SBHUFFRDY;
UnownedPtr<const CJBig2_HuffmanTable> SBHUFFRSIZE;
int8_t SBRAT[4];

private:
struct ComposeData {
    int32_t x;
    int32_t y;
    uint32_t increment = 0;
};
ComposeData GetComposeData(int32_t SI,
                           int32_t TI,
                           uint32_t WI,
                           uint32_t HI) const;
};

#endif // CORE_FXCODEC_JBIG2_JBIG2_TRDPROC_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JPEG_JPEGMODULE_H_
#define CORE_FXCODEC_JPEG_JPEGMODULE_H_

#include <csetjmp>
#include <memory>

#include "build/build_config.h"
#include "core/fxcodec/codec_module_iface.h"
#include "third_party/base/span.h"

class CFX_DIBBase;

namespace fxcodec {

class CFX_DIBAttribute;
class ScanlineDecoder;

class JpegModule final : public ModuleIface {
public:
    std::unique_ptr<ScanlineDecoder> CreateDecoder(
        pdfium::span<const uint8_t> src_span,
        int width,
        int height,
        int nComps,
        bool ColorTransform);

    // ModuleIface:
    FX_FILESIZE GetAvailInput(Context* pContext) const override;
    bool Input(Context* pContext,
        RetainPtr<CFX_CodecMemory> codec_memory,
        CFX_DIBAttribute* pAttribute) override;

    jmp_buf* GetJumpMark(Context* pContext);
    bool LoadInfo(pdfium::span<const uint8_t> src_span,
        int* width,
        int* height,
        int* num_components,
        int* bits_per_components,
        bool* color_transform);

    std::unique_ptr<Context> Start();
};

#ifdef PDF_ENABLE_XFA
    int ReadHeader(Context* pContext,
        int* width,
        int* height,
        int* nComps,
        CFX_DIBAttribute* pAttribute);
#endif // PDF_ENABLE_XFA

    bool StartScanline(Context* pContext, int down_scale);
    bool ReadScanline(Context* pContext, uint8_t* dest_buf);

#ifdef OS_WIN
    static bool JpegEncode(const RetainPtr<CFX_DIBBase>& pSource,
        uint8_t** dest_buf,
        size_t* dest_size);
#endif
};
```

```
#endif // defined(OS_WIN)
```

```
};
```

```
} // namespace fxcodec
```

```
using JpegModule = fxcodec::JpegModule;
```

```
#endif // CORE_FXCODER_JPEG_JPEGMODULE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JPX_CJPX_DECODER_H_
#define CORE_FXCODEC_JPX_CJPX_DECODER_H_

#include <memory>

#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/span.h"

#if defined(USE_SYSTEM_LIBOPENJPEG2)
#include <openjpeg.h>
#else
#include "third_party/libopenjpeg20/openjpeg.h"
#endif

namespace fxcodec {

struct DecodeData;

class CJPX_Decoder {
public:
    enum ColorSpaceOption {
        kNoColorSpace,
        kNormalColorSpace,
        kIndexedColorSpace
    };

    static void Sycc420ToRgbForTesting(opj_image_t* img);

    explicit CJPX_Decoder(ColorSpaceOption option);
    ~CJPX_Decoder();

    bool Init(pdfium::span<const uint8_t> src_data);
    void GetInfo(uint32_t* width, uint32_t* height, uint32_t* components);
    bool StartDecode();
    bool Decode(uint8_t* dest_buf,
                uint32_t pitch,
                pdfium::span<const uint8_t> offsets);

private:
    const ColorSpaceOption m_ColorSpaceOption;
    pdfium::span<const uint8_t> m_SrcData;
    UnownedPtr<opj_image_t> m_Image;
    UnownedPtr<opj_codec_t> m_Codec;
    std::unique_ptr<DecodeData> m_DeclareData;
    UnownedPtr<opj_stream_t> m_Stream;
    opj_dparameters_t m_Parameters;
};

} // namespace fxcodec

using fxcodec::CJPX_Decoder;

#endif // CORE_FXCODEC_JPX_CJPX_DECODER_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JPX_JPX_DECODE_UTILS_H_
#define CORE_FXCODEC_JPX_JPX_DECODE_UTILS_H_

#include <stdint.h>

#if defined(USE_SYSTEM_LIBOPENJPEG2)
#include <openjpeg.h>
#else
#include "third_party/libopenjpeg20/openjpeg.h"
#endif

namespace fxcodec {

struct DecodeData {
  DecodeData(const uint8_t* data, OPJ_SIZE_T size)
    : src_data(data), src_size(size), offset(0) {}

  const uint8_t* src_data;
  OPJ_SIZE_T src_size;
  OPJ_SIZE_T offset;
};

/* Wrappers for C-style callbacks. */
OPJ_SIZE_T opj_read_from_memory(void* p_buffer,
                                OPJ_SIZE_T nb_bytes,
                                void* p_user_data);
OPJ_OFF_T opj_skip_from_memory(OPJ_OFF_T nb_bytes, void* p_user_data);
OPJ_BOOL opj_seek_from_memory(OPJ_OFF_T nb_bytes, void* p_user_data);

} // namespace fxcodec

#endif // CORE_FXCODEC_JPX_JPX_DECODE_UTILS_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_JPX_JPXMODULE_H_
#define CORE_FXCODEC_JPX_JPXMODULE_H_

#include <memory>
#include <vector>

#include "core/fxcodec/jpx/cjpx_decoder.h"
#include "core/fxcrt/fx_system.h"
#include "third_party/base/span.h"

namespace fxcodec {

class JpxModule {
public:
    static std::unique_ptr<CJPX_Decoder> CreateDecoder(
        pdfium::span<const uint8_t> src_span,
        CJPX_Decoder::ColorSpaceOption option);

    JpxModule() = delete;
    JpxModule(const JpxModule&) = delete;
    JpxModule& operator=(const JpxModule&) = delete;
};

} // namespace fxcodec

using JpxModule = fxcodec::JpxModule;

#endif // CORE_FXCODEC_JPX_JPXMODULE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_PNG_PNGMODULE_H_
#define CORE_FXCODEC_PNG_PNGMODULE_H_

#include <memory>

#include "core/fxcodec/codec_module_iface.h"

namespace fxcodec {

class PngModule final : public ModuleIface {
public:
class Delegate {
public:
virtual bool PngReadHeader(int width,
                           int height,
                           int bpc,
                           int pass,
                           int* color_type,
                           double* gamma) = 0;

// Returns true on success. |pSrcBuf| will be set if this succeeds.
// |pSrcBuf| does not take ownership of the buffer.
virtual bool PngAskScanlineBuf(int line, uint8_t** pSrcBuf) = 0;

virtual void PngFillScanlineBufCompleted(int pass, int line) = 0;
};

PngModule();
~PngModule() override;

// ModuleIface:
FX_FILESIZE GetAvailInput(Context* pContext) const override;
bool Input(Context* pContext,
           RetainPtr<CFX_CodecMemory> codec_memory,
           CFX_DIBAttribute* pAttribute) override;

std::unique_ptr<Context> Start(Delegate* pDelegate);
};

} // namespace fxcodec

using PngModule = fxcodec::PngModule;

#endif // CORE_FXCODEC_PNG_PNGMODULE_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODER_PROGRESSIVEDECODER_H_
#define CORE_FXCODER_PROGRESSIVEDECODER_H_

#include <memory>
#include <utility>
#include <vector>

#include "core/fxcodec/fx_codec_def.h"
#include "core/fxcodec/jpeg/jpegmodule.h"
#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/fx_dib.h"

#ifdef PDF_ENABLE_XFA_BMP
#include "core/fxcodec/bmp/bmpmodule.h"
#endif // PDF_ENABLE_XFA_BMP

#ifdef PDF_ENABLE_XFA_GIF
#include "core/fxcodec/gif/gifmodule.h"
#endif // PDF_ENABLE_XFA_GIF

#ifdef PDF_ENABLE_XFA_PNG
#include "core/fxcodec/png/pngmodule.h"
#endif // PDF_ENABLE_XFA_PNG

#ifdef PDF_ENABLE_XFA_TIFF
#include "core/fxcodec/tiff/tiffmodule.h"
#endif // PDF_ENABLE_XFA_TIFF

class CFX_DIBitmap;
class IFX_SeekableReadStream;

namespace fxcodec {

class CFX_DIBAttribute;
class ModuleMgr;

class Dummy {}; // Placeholder to work around C++ syntax issues

class ProgressiveDecoder :
#ifdef PDF_ENABLE_XFA_BMP
    public BmpModule::Delegate,
#endif // PDF_ENABLE_XFA_BMP
#ifdef PDF_ENABLE_XFA_GIF
    public GifModule::Delegate,
#endif // PDF_ENABLE_XFA_GIF
#ifdef PDF_ENABLE_XFA_PNG
    public PngModule::Delegate,
#endif // PDF_ENABLE_XFA_PNG
    public Dummy {
public:
    enum FXCodec_Format {
        FXCodec_Invalid = 0,
        FXCodec_1bppGray = 0x101,
        FXCodec_1bppRgb = 0x001,
    };
};
```



```
    }

    int m_ItemSize;
    std::vector<uint8_t> m_pWeightTables;
};

class CFXCODEC_VertTable {
public:
    CFXCODEC_VertTable();
    ~CFXCODEC_VertTable();

    void Calc(int dest_len, int src_len);
    PixelWeight* GetPixelWeight(int pixel) {
        return reinterpret_cast<PixelWeight*>(m_pWeightTables.data() +
                                             pixel * m_ItemSize);
    }
    int m_ItemSize;
    std::vector<uint8_t> m_pWeightTables;
};

#ifdef PDF_ENABLE_XFA_PNG
    // PngModule::Delegate
    bool PngReadHeader(int width,
                      int height,
                      int bpc,
                      int pass,
                      int* color_type,
                      double* gamma) override;
    bool PngAskScanlineBuf(int line, uint8_t** pSrcBuf) override;
    void PngFillScanlineBufCompleted(int pass, int line) override;
#endif // PDF_ENABLE_XFA_PNG

#ifdef PDF_ENABLE_XFA_GIF
    // GifModule::Delegate
    void GifRecordCurrentPosition(uint32_t& cur_pos) override;
    bool GifInputRecordPositionBuf(uint32_t rcd_pos,
                                   const FX_RECT& img_rc,
                                   int32_t pal_num,
                                   CFX_GifPalette* pal_ptr,
                                   int32_t delay_time,
                                   bool user_input,
                                   int32_t trans_index,
                                   int32_t disposal_method,
                                   bool interlace) override;
    void GifReadScanline(int32_t row_num, uint8_t* row_buf) override;
#endif // PDF_ENABLE_XFA_GIF

#ifdef PDF_ENABLE_XFA_BMP
    // BmpModule::Delegate
    bool BmpInputImagePositionBuf(uint32_t rcd_pos) override;
    void BmpReadScanline(uint32_t row_num,
                         pdfium::span<const uint8_t> row_buf) override;
#endif // PDF_ENABLE_XFA_BMP

private:
#ifdef PDF_ENABLE_XFA_BMP
    bool BmpReadMoreData(BmpModule* pBmpModule,
                        ModuleInterface::Context* pBmpContext,
                        FXCODEC_STATUS& err_status);
    bool BmpDetectImageTypeInBuffer(CFX_DIBAttribute* pAttribute);
    FXCODEC_STATUS BmpStartDecode(const RetainPtr<CFX_DIBitmap>& pDIBitmap);
    FXCODEC_STATUS BmpContinueDecode();
#endif // PDF_ENABLE_XFA_BMP
```

```
#ifndef PDF_ENABLE_XFA_GIF
bool GifReadMoreData(GifModule* pGifModule, FXCODEC_STATUS& err_status);
bool GifDetectImageTypeInBuffer();
FXCODEC_STATUS GifStartDecode(const RetainPtr<CFX_DIBitmap>& pDIBitmap);
FXCODEC_STATUS GifContinueDecode();
void GifDoubleLineResampleVert(const RetainPtr<CFX_DIBitmap>& pDeviceBitmap,
                               double scale_y,
                               int dest_row);
#endif // PDF_ENABLE_XFA_GIF

#ifndef PDF_ENABLE_XFA_PNG
void PngOneOneMapResampleHorz(const RetainPtr<CFX_DIBitmap>& pDeviceBitmap,
                              int32_t dest_line,
                              uint8_t* src_scan,
                              FXCodec_Format src_format);
bool PngDetectImageTypeInBuffer(CFX_DIBAttribute* pAttribute);
FXCODEC_STATUS PngStartDecode(const RetainPtr<CFX_DIBitmap>& pDIBitmap);
FXCODEC_STATUS PngContinueDecode();
#endif // PDF_ENABLE_XFA_PNG

#ifndef PDF_ENABLE_XFA_TIFF
bool TiffDetectImageTypeFromFile(CFX_DIBAttribute* pAttribute);
FXCODEC_STATUS TiffContinueDecode();
#endif // PDF_ENABLE_XFA_TIFF

bool JpegReadMoreData(JpegModule* pJpegModule, FXCODEC_STATUS& err_status);
bool JpegDetectImageTypeInBuffer(CFX_DIBAttribute* pAttribute);
FXCODEC_STATUS JpegStartDecode(const RetainPtr<CFX_DIBitmap>& pDIBitmap);
FXCODEC_STATUS JpegContinueDecode();

bool DetectImageType(FXCODEC_IMAGE_TYPE imageType,
                    CFX_DIBAttribute* pAttribute);
bool ReadMoreData(ModuleIface* pModule,
                 ModuleIface::Context* pContext,
                 bool invalidate_buffer,
                 FXCODEC_STATUS& err_status);

void GetDownScale(int& down_scale);
void GetTransMethod(FXDIB_Format dest_format, FXCodec_Format src_format);

void ReSampleScanline(const RetainPtr<CFX_DIBitmap>& pDeviceBitmap,
                     int32_t dest_line,
                     uint8_t* src_scan,
                     FXCodec_Format src_format);
void Resample(const RetainPtr<CFX_DIBitmap>& pDeviceBitmap,
             int32_t src_line,
             uint8_t* src_scan,
             FXCodec_Format src_format);
void ResampleVert(const RetainPtr<CFX_DIBitmap>& pDeviceBitmap,
                 double scale_y,
                 int dest_row);
void ResampleVertBT(const RetainPtr<CFX_DIBitmap>& pDeviceBitmap,
                   double scale_y,
                   int dest_row);

FXCODEC_STATUS m_status = FXCODEC_STATUS_DECODE_FINISH;
FXCODEC_IMAGE_TYPE m_imageType = FXCODEC_IMAGE_UNKNOWN;
RetainPtr<IFX_SeekableReadStream> m_pFile;
RetainPtr<CFX_DIBitmap> m_pDeviceBitmap;
UnownedPtr<ModuleMgr> m_pCodecMgr;
RetainPtr<CFX_CodecMemory> m_pCodecMemory;
std::unique_ptr<uint8_t, FxFreeDeleter> m_pDecodeBuf;
```

```
    std::unique_ptr<FX_ARGB, FxFreeDeleter> m_pSrcPalette;
    std::unique_ptr<ModuleIface::Context> m_pJpegContext;
#ifdef PDF_ENABLE_XFA_BMP
    std::unique_ptr<ModuleIface::Context> m_pBmpContext;
#endif // PDF_ENABLE_XFA_BMP
#ifdef PDF_ENABLE_XFA_GIF
    std::unique_ptr<ModuleIface::Context> m_pGifContext;
#endif // PDF_ENABLE_XFA_GIF
#ifdef PDF_ENABLE_XFA_PNG
    std::unique_ptr<ModuleIface::Context> m_pPngContext;
#endif // PDF_ENABLE_XFA_PNG
#ifdef PDF_ENABLE_XFA_TIFF
    std::unique_ptr<ModuleIface::Context> m_pTiffContext;
#endif // PDF_ENABLE_XFA_TIFF
    uint32_t m_offSet = 0;
    int m_ScanlineSize = 0;
    CFXCODDEC_WeightTable m_WeightHorz;
    CFXCODDEC_VertTable m_WeightVert;
    CFXCODDEC_HorzTable m_WeightHorzOO;
    int m_SrcWidth = 0;
    int m_SrcHeight = 0;
    int m_SrcComponents = 0;
    int m_SrcBPC = 0;
    FX_RECT m_clipBox;
    int m_startX = 0;
    int m_startY = 0;
    int m_sizeX = 0;
    int m_sizeY = 0;
    int m_TransMethod = -1;
    int m_SrcPaletteNumber = 0;
    int m_SrcRow = 0;
    FXCodec_Format m_SrcFormat = FXCodec_Invalid;
    int m_SrcPassNumber = 0;
    size_t m_FrameNumber = 0;
    size_t m_FrameCur = 0;
#ifdef PDF_ENABLE_XFA_GIF
    int m_GifBgIndex = 0;
    CFX_GifPalette* m_pGifPalette = nullptr;
    int32_t m_GifPltNumber = 0;
    int m_GifTransIndex = -1;
    FX_RECT m_GifFrameRect;
    bool m_InvalidGifBuffer = true;
#endif // PDF_ENABLE_XFA_GIF
#ifdef PDF_ENABLE_XFA_BMP
    bool m_BmpIsTopBottom = false;
#endif // PDF_ENABLE_XFA_BMP
};

} // namespace fxcodec

using ProgressiveDecoder = fxcodec::ProgressiveDecoder;

#endif // CORE_FXCODDEC_PROGRESSIVEDECODER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_SCANLINEDECODER_H_
#define CORE_FXCODEC_SCANLINEDECODER_H_

#include "core/fxcrt/fx_system.h"

class PauseIndicatorIface;

namespace fxcodec {

class ScanlineDecoder {
public:
  ScanlineDecoder();
  ScanlineDecoder(int nOrigWidth,
                  int nOrigHeight,
                  int nOutputWidth,
                  int nOutputHeight,
                  int nComps,
                  int nBpc,
                  uint32_t nPitch);
  virtual ~ScanlineDecoder();

  const uint8_t* GetScanline(int line);
  bool SkipToScanline(int line, PauseIndicatorIface* pPause);

  int GetWidth() const { return m_OutputWidth; }
  int GetHeight() const { return m_OutputHeight; }
  int CountComps() const { return m_nComps; }
  int GetBPC() const { return m_bpc; }

  virtual uint32_t GetSrcOffset() = 0;

protected:
  virtual bool v_Rewind() = 0;
  virtual uint8_t* v_GetNextLine() = 0;

  uint8_t* ReadNextLine();

  int m_OrigWidth;
  int m_OrigHeight;
  int m_OutputWidth;
  int m_OutputHeight;
  int m_nComps;
  int m_bpc;
  uint32_t m_Pitch;
  int m_NextLine = -1;
  uint8_t* m_pLastScanline = nullptr;
};

} // namespace fxcodec

using ScanlineDecoder = fxcodec::ScanlineDecoder;

#endif // CORE_FXCODEC_SCANLINEDECODER_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCODEC_TIFF_TIFFMODULE_H_
#define CORE_FXCODEC_TIFF_TIFFMODULE_H_

#include <memory>

#include "core/fxcodec/codec_module_iface.h"

class CFX_DIBitmap;
class IFX_SeekableReadStream;

namespace fxcodec {

class CFX_DIBAttribute;

class TiffModule final : public ModuleIface {
public:
    std::unique_ptr<Context> CreateDecoder(
        const RetainPtr<IFX_SeekableReadStream>& file_ptr);

    // ModuleIface:
    FX_FILESIZE GetAvailInput(Context* pContext) const override;
    bool Input(Context* pContext,
        RetainPtr<CFX_CodecMemory> codec_memory,
        CFX_DIBAttribute* pAttribute) override;

    bool LoadFrameInfo(Context* ctx,
        int32_t frame,
        int32_t* width,
        int32_t* height,
        int32_t* comps,
        int32_t* bpc,
        CFX_DIBAttribute* pAttribute);
    bool Decode(Context* ctx, const RetainPtr<CFX_DIBitmap>& pDIBitmap);
};

} // namespace fxcodec

using TiffModule = fxcodec::TiffModule;

#endif // CORE_FXCODEC_TIFF_TIFFMODULE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FXCRT_AUTORESTORER_H_  
#define CORE_FXCRT_AUTORESTORER_H_  
  
namespace fxcrt {  
  
template <typename T>  
class AutoRestorer {  
public:  
    explicit AutoRestorer(T* location)  
        : m_Location(location), m_OldValue(*location) {}  
    ~AutoRestorer() {  
        if (m_Location)  
            *m_Location = m_OldValue;  
    }  
    void AbandonRestoration() { m_Location = nullptr; }  
  
private:  
    T* m_Location;  
    const T m_OldValue;  
};  
  
} // namespace fxcrt  
  
using fxcrt::AutoRestorer;  
  
#endif // CORE_FXCRT_AUTORESTORER_H_
```

```
// Copyright 2019 The PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

#ifndef CORE_FXCRT_BYTEORDER_H_
#define CORE_FXCRT_BYTEORDER_H_

#include "build/build_config.h"
#include "third_party/base/sys_byteorder.h"

namespace fxcrt {

// Converts the bytes in |x| from host order (endianness) to little endian, and
// returns the result.
inline uint16_t ByteSwapToLE16(uint16_t x) {
    return pdfium::base::ByteSwapToLE16(x);
}

inline uint32_t ByteSwapToLE32(uint32_t x) {
    return pdfium::base::ByteSwapToLE32(x);
}

// Converts the bytes in |x| from host order (endianness) to big endian, and
// returns the result.
inline uint16_t ByteSwapToBE16(uint16_t x) {
    #if defined(ARCH_CPU_LITTLE_ENDIAN)
        return pdfium::base::ByteSwap(x);
    #else
        return x;
    #endif
}

inline uint32_t ByteSwapToBE32(uint32_t x) {
    #if defined(ARCH_CPU_LITTLE_ENDIAN)
        return pdfium::base::ByteSwap(x);
    #else
        return x;
    #endif
}

} // namespace fxcrt

#endif // CORE_FXCRT_BYTEORDER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_BYTESTRING_H_
#define CORE_FXCRT_BYTESTRING_H_

#include <functional>
#include <iterator>
#include <ostream>
#include <sstream>
#include <utility>

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/string_data_template.h"
#include "core/fxcrt/string_view_template.h"
#include "third_party/base/logging.h"
#include "third_party/base/optional.h"
#include "third_party/base/span.h"

namespace fxcrt {

class ByteString_Assign_Test;
class ByteString_Concat_Test;
class StringPool_ByteString_Test;

// A mutable string with shared buffers using copy-on-write semantics that
// avoids the cost of std::string's iterator stability guarantees.
class ByteString {
public:
    using CharType = char;
    using const_iterator = const CharType*;
    using const_reverse_iterator = std::reverse_iterator<const_iterator>;

    static ByteString FormatInteger(int i) WARN_UNUSED_RESULT;
    static ByteString FormatFloat(float f) WARN_UNUSED_RESULT;
    static ByteString Format(const char* pFormat, ...) WARN_UNUSED_RESULT;
    static ByteString FormatV(const char* pFormat,
                              va_list argList) WARN_UNUSED_RESULT;

    ByteString();
    ByteString(const ByteString& other);
    ByteString(ByteString&& other) noexcept;

    // Deliberately implicit to avoid calling on every string literal.
    // NOLINTNEXTLINE(runtime/explicit)
    ByteString(char ch);
    // NOLINTNEXTLINE(runtime/explicit)
    ByteString(const char* ptr);

    // No implicit conversions from wide strings.
    // NOLINTNEXTLINE(runtime/explicit)
    ByteString(wchar_t) = delete;

    ByteString(const char* pStr, size_t len);
    ByteString(const uint8_t* pStr, size_t len);

    explicit ByteString(ByteStringView bstrc);
    ByteString(ByteStringView str1, ByteStringView str2);
    ByteString(const std::initializer_list<ByteStringView>& list);
};
```

```
explicit ByteString(const std::ostream& outStream);

~ByteString();

void clear() { m_pData.Reset(); }

// Explicit conversion to C-style string.
// Note: Any subsequent modification of |this| will invalidate the result.
const char* c_str() const { return m_pData ? m_pData->m_String : ""; }

// Explicit conversion to uint8_t*.
// Note: Any subsequent modification of |this| will invalidate the result.
const uint8_t* raw_str() const {
    return m_pData ? reinterpret_cast<const uint8_t*>(m_pData->m_String)
        : nullptr;
}

// Explicit conversion to ByteStringView.
// Note: Any subsequent modification of |this| will invalidate the result.
ByteStringView AsStringView() const {
    return ByteStringView(raw_str(), GetLength());
}

// Explicit conversion to span.
// Note: Any subsequent modification of |this| will invalidate the result.
pdfium::span<const char> span() const {
    return pdfium::make_span(m_pData ? m_pData->m_String : nullptr,
        GetLength());
}

pdfium::span<const uint8_t> raw_span() const {
    return pdfium::make_span(raw_str(), GetLength());
}

// Note: Any subsequent modification of |this| will invalidate iterators.
const_iterator begin() const { return m_pData ? m_pData->m_String : nullptr; }
const_iterator end() const {
    return m_pData ? m_pData->m_String + m_pData->m_nDataLength : nullptr;
}

// Note: Any subsequent modification of |this| will invalidate iterators.
const_reverse_iterator rbegin() const {
    return const_reverse_iterator(end());
}
const_reverse_iterator rend() const {
    return const_reverse_iterator(begin());
}

size_t GetLength() const { return m_pData ? m_pData->m_nDataLength : 0; }
size_t GetStringLength() const {
    return m_pData ? strlen(m_pData->m_String) : 0;
}

bool IsEmpty() const { return !GetLength(); }
bool IsValidIndex(size_t index) const { return index < GetLength(); }
bool IsValidLength(size_t length) const { return length <= GetLength(); }

int Compare(ByteStringView str) const;
bool EqualNoCase(ByteStringView str) const;

bool operator==(const char* ptr) const;
bool operator==(ByteStringView str) const;
bool operator==(const ByteString& other) const;

bool operator!=(const char* ptr) const { return !(*this == ptr); }
```

```
bool operator!=(ByteStringView str) const { return !(*this == str); }
bool operator!=(const ByteString& other) const { return !(*this == other); }

bool operator<(const char* ptr) const;
bool operator<(ByteStringView str) const;
bool operator<(const ByteString& other) const;

ByteString& operator=(const char* str);
ByteString& operator=(ByteStringView str);
ByteString& operator=(const ByteString& that);
ByteString& operator=(ByteString&& that);

ByteString& operator+=(char ch);
ByteString& operator+=(const char* str);
ByteString& operator+=(const ByteString& str);
ByteString& operator+=(ByteStringView str);

CharType operator[](const size_t index) const {
    CHECK(IsValidIndex(index));
    return m_pData->m_String[index];
}

CharType First() const { return GetLength() ? (*this)[0] : 0; }
CharType Last() const { return GetLength() ? (*this)[GetLength() - 1] : 0; }

void SetAt(size_t index, char c);

size_t Insert(size_t index, char ch);
size_t InsertAtFront(char ch) { return Insert(0, ch); }
size_t InsertAtBack(char ch) { return Insert(GetLength(), ch); }
size_t Delete(size_t index, size_t count = 1);

void Reserve(size_t len);

// Note: any modification of the string (including ReleaseBuffer()) may
// invalidate the span, which must not outlive its buffer.
pdfium::span<char> GetBuffer(size_t nMinBufLength);
void ReleaseBuffer(size_t nNewLength);

ByteString Mid(size_t first, size_t count) const;
ByteString Left(size_t count) const;
ByteString Right(size_t count) const;

Optional<size_t> Find(ByteStringView subStr, size_t start = 0) const;
Optional<size_t> Find(char ch, size_t start = 0) const;
Optional<size_t> ReverseFind(char ch) const;

bool Contains(ByteStringView lpszSub, size_t start = 0) const {
    return Find(lpszSub, start).has_value();
}

bool Contains(char ch, size_t start = 0) const {
    return Find(ch, start).has_value();
}

void MakeLower();
void MakeUpper();

void Trim();
void Trim(char target);
void Trim(ByteStringView targets);

void TrimLeft();
```

```

void TrimLeft(char target);
void TrimLeft(ByteStringView targets);

void TrimRight();
void TrimRight(char target);
void TrimRight(ByteStringView targets);

size_t Replace(ByteStringView pOld, ByteStringView pNew);
size_t Remove(char ch);

uint32_t GetID() const { return AsStringView().GetID(); }

protected:
using StringData = StringDataTemplate<char>;

void ReallocBeforeWrite(size_t nNewLen);
void AllocBeforeWrite(size_t nNewLen);
void AllocCopy(ByteString& dest, size_t nCopyLen, size_t nCopyIndex) const;
void AssignCopy(const char* pSrcData, size_t nSrcLen);
void Concat(const char* pSrcData, size_t nSrcLen);
intptr_t ReferenceCountForTesting() const;

RetainPtr<StringData> m_pData;

friend ByteString_Assign_Test;
friend ByteString_Concat_Test;
friend class StringPool_ByteString_Test;
};

inline bool operator==(const char* lhs, const ByteString& rhs) {
    return rhs == lhs;
}
inline bool operator==(ByteStringView lhs, const ByteString& rhs) {
    return rhs == lhs;
}
inline bool operator!=(const char* lhs, const ByteString& rhs) {
    return rhs != lhs;
}
inline bool operator!=(ByteStringView lhs, const ByteString& rhs) {
    return rhs != lhs;
}
inline bool operator<(const char* lhs, const ByteString& rhs) {
    return rhs.Compare(lhs) > 0;
}

inline ByteString operator+(ByteStringView str1, ByteStringView str2) {
    return ByteString(str1, str2);
}
inline ByteString operator+(ByteStringView str1, const char* str2) {
    return ByteString(str1, str2);
}
inline ByteString operator+(const char* str1, ByteStringView str2) {
    return ByteString(str1, str2);
}
inline ByteString operator+(ByteStringView str1, char ch) {
    return ByteString(str1, ByteStringView(ch));
}
inline ByteString operator+(char ch, ByteStringView str2) {
    return ByteString(ch, str2);
}
inline ByteString operator+(const ByteString& str1, const ByteString& str2) {
    return ByteString(str1.AsStringView(), str2.AsStringView());
}

```

```
inline ByteString operator+(const ByteString& str1, char ch) {
    return ByteString(str1.AsStringView(), ByteStringView(ch));
}
inline ByteString operator+(char ch, const ByteString& str2) {
    return ByteString(ch, str2.AsStringView());
}
inline ByteString operator+(const ByteString& str1, const char* str2) {
    return ByteString(str1.AsStringView(), str2);
}
inline ByteString operator+(const char* str1, const ByteString& str2) {
    return ByteString(str1, str2.AsStringView());
}
inline ByteString operator+(const ByteString& str1, ByteStringView str2) {
    return ByteString(str1.AsStringView(), str2);
}
inline ByteString operator+(ByteStringView str1, const ByteString& str2) {
    return ByteString(str1, str2.AsStringView());
}

std::ostream& operator<<(std::ostream& os, const ByteString& str);
std::ostream& operator<<(std::ostream& os, ByteStringView str);

} // namespace fxcrt

using ByteString = fxcrt::ByteString;

uint32_t FX_HashCode_GetA(ByteStringView str, bool bIgnoreCase);
uint32_t FX_HashCode_GetAsIfW(ByteStringView str, bool bIgnoreCase);

namespace std {

template <>
struct hash<ByteString> {
    std::size_t operator()(const ByteString& str) const {
        return FX_HashCode_GetA(str.AsStringView(), false);
    }
};

} // namespace std

extern template struct std::hash<ByteString>;

#endif // CORE_FXCRT_BYTESTRING_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_BINARYBUF_H_
#define CORE_FXCRT_CFX_BINARYBUF_H_

#include <memory>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "third_party/base/span.h"

class CFX_BinaryBuf {
public:
    CFX_BinaryBuf();
    virtual ~CFX_BinaryBuf();

    pdfium::span<uint8_t> GetSpan();
    pdfium::span<const uint8_t> GetSpan() const;
    uint8_t* GetBuffer() const { return m_pBuffer.get(); }
    size_t GetSize() const { return m_DataSize; }
    virtual size_t GetLength() const;
    bool IsEmpty() const { return GetLength() == 0; }

    void Clear();
    void SetAllocStep(size_t step) { m_AllocStep = step; }
    void EstimateSize(size_t size);
    void AppendSpan(pdfium::span<const uint8_t> span);
    void AppendBlock(const void* pBuf, size_t size);
    void AppendString(const ByteString& str) {
        AppendBlock(str.c_str(), str.GetLength());
    }

    void AppendByte(uint8_t byte) {
        ExpandBuf(1);
        m_pBuffer.get()[m_DataSize++] = byte;
    }

    void Delete(size_t start_index, size_t count);

    // Releases ownership of |m_pBuffer| and returns it.
    std::unique_ptr<uint8_t, FxFreeDeleter> DetachBuffer();

protected:
    void ExpandBuf(size_t size);

    size_t m_AllocStep = 0;
    size_t m_AllocSize = 0;
    size_t m_DataSize = 0;
    std::unique_ptr<uint8_t, FxFreeDeleter> m_pBuffer;
};

#endif // CORE_FXCRT_CFX_BINARYBUF_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_BITSTREAM_H_
#define CORE_FXCRT_CFX_BITSTREAM_H_

#include <stdint.h>

#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/span.h"

class CFX_BitStream {
public:
  explicit CFX_BitStream(pdfium::span<const uint8_t> pData);
  ~CFX_BitStream();

  void ByteAlign();

  bool IsEOF() const { return m_BitPos >= m_BitSize; }
  uint32_t GetPos() const { return m_BitPos; }
  uint32_t GetBits(uint32_t nBits);

  void SkipBits(uint32_t nBits) { m_BitPos += nBits; }
  void Rewind() { m_BitPos = 0; }

  uint32_t BitsRemaining() const {
    return m_BitSize >= m_BitPos ? m_BitSize - m_BitPos : 0;
  }

private:
  uint32_t m_BitPos;
  uint32_t m_BitSize;
  UnownedPtr<const uint8_t> m_pData;
};

#endif // CORE_FXCRT_CFX_BITSTREAM_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_DATETIME_H_
#define CORE_FXCRT_CFX_DATETIME_H_

#include "core/fxcrt/fx_system.h"

bool FX_IsLeapYear(int32_t iYear);
uint8_t FX_DaysInMonth(int32_t iYear, uint8_t iMonth);

class CFX_DateTime {
public:
    static CFX_DateTime Now();

    CFX_DateTime()
        : year_(0),
          month_(0),
          day_(0),
          hour_(0),
          minute_(0),
          second_(0),
          millisecond_(0) {}
    CFX_DateTime(int32_t year,
                 uint8_t month,
                 uint8_t day,
                 uint8_t hour,
                 uint8_t minute,
                 uint8_t second,
                 uint16_t millisecond)
        : year_(year),
          month_(month),
          day_(day),
          hour_(hour),
          minute_(minute),
          second_(second),
          millisecond_(millisecond) {}

    void Reset() {
        year_ = 0;
        month_ = 0;
        day_ = 0;
        hour_ = 0;
        minute_ = 0;
        second_ = 0;
        millisecond_ = 0;
    }

    bool IsSet() const {
        return year_ != 0 || month_ != 0 || day_ != 0 || hour_ != 0 ||
            minute_ != 0 || second_ != 0 || millisecond_ != 0;
    }

    void SetDate(int32_t year, uint8_t month, uint8_t day) {
        year_ = year;
        month_ = month;
        day_ = day;
    }

    void SetTime(uint8_t hour,
```

```
        uint8_t minute,
        uint8_t second,
        uint16_t millisecond) {
    hour_ = hour;
    minute_ = minute;
    second_ = second;
    millisecond_ = millisecond;
}

int32_t GetYear() const { return year_; }
uint8_t GetMonth() const { return month_; }
uint8_t GetDay() const { return day_; }
uint8_t GetHour() const { return hour_; }
uint8_t GetMinute() const { return minute_; }
uint8_t GetSecond() const { return second_; }
uint16_t GetMillisecond() const { return millisecond_; }
int32_t GetDayOfWeek() const;

bool operator==(const CFX_DateTime& other) const;

private:
    int32_t year_;
    uint8_t month_;
    uint8_t day_;
    uint8_t hour_;
    uint8_t minute_;
    uint8_t second_;
    uint16_t millisecond_;
};

struct FX_TIMEZONE {
    int8_t tzHour;
    uint8_t tzMinute;
};

#endif // CORE_FXCRT_CFX_DATETIME_H
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_FILEACCESS_POSIX_H_
#define CORE_FXCRT_CFX_FILEACCESS_POSIX_H_

#include "build/build_config.h"
#include "core/fxcrt/fileaccess_iface.h"
#include "core/fxcrt/fx_system.h"

#if _FX_PLATFORM_ != _FX_PLATFORM_LINUX_ && !defined(OS_MACOSX) && \
    !defined(OS_ANDROID)
#error "Included on the wrong platform"
#endif

class CFX_FileAccess_Posix final : public FileAccessIface {
public:
    CFX_FileAccess_Posix();
    ~CFX_FileAccess_Posix() override;

    // FileAccessIface:
    bool Open(ByteStringView fileName, uint32_t dwMode) override;
    bool Open(WideStringView fileName, uint32_t dwMode) override;
    void Close() override;
    FX_FILESIZE GetSize() const override;
    FX_FILESIZE GetPosition() const override;
    FX_FILESIZE SetPosition(FX_FILESIZE pos) override;
    size_t Read(void* pBuffer, size_t szBuffer) override;
    size_t Write(const void* pBuffer, size_t szBuffer) override;
    size_t ReadPos(void* pBuffer, size_t szBuffer, FX_FILESIZE pos) override;
    size_t WritePos(const void* pBuffer,
                   size_t szBuffer,
                   FX_FILESIZE pos) override;
    bool Flush() override;
    bool Truncate(FX_FILESIZE szFile) override;

private:
    int32_t m_nFD;
};

#endif // CORE_FXCRT_CFX_FILEACCESS_POSIX_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_FILEACCESS_WINDOWS_H_
#define CORE_FXCRT_CFX_FILEACCESS_WINDOWS_H_

#include "build/build_config.h"
#include "core/fxcrt/fileaccess_iface.h"
#include "core/fxcrt/fx_system.h"

#if !defined(OS_WIN)
#error "Included on the wrong platform"
#endif

class CFX_FileAccess_Windows final : public FileAccessIface {
public:
  CFX_FileAccess_Windows();
  ~CFX_FileAccess_Windows() override;

  // FileAccessIface
  bool Open(ByteStringView fileName, uint32_t dwMode) override;
  bool Open(WideStringView fileName, uint32_t dwMode) override;
  void Close() override;
  FX_FILESIZE GetSize() const override;
  FX_FILESIZE GetPosition() const override;
  FX_FILESIZE SetPosition(FX_FILESIZE pos) override;
  size_t Read(void* pBuffer, size_t szBuffer) override;
  size_t Write(const void* pBuffer, size_t szBuffer) override;
  size_t ReadPos(void* pBuffer, size_t szBuffer, FX_FILESIZE pos) override;
  size_t WritePos(const void* pBuffer,
                  size_t szBuffer,
                  FX_FILESIZE pos) override;
  bool Flush() override;
  bool Truncate(FX_FILESIZE szFile) override;

private:
  void* m_hFile;
};

#endif // CORE_FXCRT_CFX_FILEACCESS_WINDOWS_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_FIXEDBUFGROW_H_
#define CORE_FXCRT_CFX_FIXEDBUFGROW_H_

#include <memory>

#include "core/fxcrt/fx_memory_wrappers.h"

template <class DataType, size_t FixedSize>
class CFX_FixedBufGrow {
public:
  explicit CFX_FixedBufGrow(size_t data_size) {
    if (data_size > FixedSize) {
      m_pGrowData.reset(FX_Alloc(DataType, data_size));
      return;
    }
    memset(m_FixedData, 0, sizeof(DataType) * FixedSize);
  }
  operator DataType*() { return m_pGrowData ? m_pGrowData.get() : m_FixedData; }

private:
  std::unique_ptr<DataType, FxFreeDeleter> m_pGrowData;
  DataType m_FixedData[FixedSize];
};

#endif // CORE_FXCRT_CFX_FIXEDBUFGROW_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_MEMORYSTREAM_H_
#define CORE_FXCRT_CFX_MEMORYSTREAM_H_

#include <memory>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_stream.h"
#include "core/fxcrt/retain_ptr.h"

class CFX_MemoryStream final : public IFX_SeekableStream {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    // IFX_SeekableStream
    FX_FILESIZE GetSize() override;
    FX_FILESIZE GetPosition() override;
    bool IsEOF() override;
    bool ReadBlockAtOffset(void* buffer,
                           FX_FILESIZE offset,
                           size_t size) override;
    size_t ReadBlock(void* buffer, size_t size) override;
    bool WriteBlockAtOffset(const void* buffer,
                            FX_FILESIZE offset,
                            size_t size) override;

    bool Flush() override;

    const uint8_t* GetBuffer() const { return m_data.get(); }

private:
    CFX_MemoryStream();
    CFX_MemoryStream(std::unique_ptr<uint8_t, FxFreeDeleter> pBuffer,
                    size_t nSize);
    ~CFX_MemoryStream() override;

    std::unique_ptr<uint8_t, FxFreeDeleter> m_data;
    size_t m_nTotalSize;
    size_t m_nCurSize;
    size_t m_nCurPos = 0;
};

#endif // CORE_FXCRT_CFX_MEMORYSTREAM_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_READONLYMEMORYSTREAM_H_
#define CORE_FXCRT_CFX_READONLYMEMORYSTREAM_H_

#include <memory>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_stream.h"
#include "core/fxcrt/retain_ptr.h"
#include "third_party/base/span.h"

class CFX_ReadOnlyMemoryStream final : public IFX_SeekableReadStream {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    // IFX_SeekableReadStream:
    FX_FILESIZE GetSize() override;
    bool ReadBlockAtOffset(void* buffer,
                           FX_FILESIZE offset,
                           size_t size) override;

private:
    CFX_ReadOnlyMemoryStream(std::unique_ptr<uint8_t, FxFreeDeleter> data,
                             size_t size);
    explicit CFX_ReadOnlyMemoryStream(pdfium::span<const uint8_t> span);
    ~CFX_ReadOnlyMemoryStream() override;

    std::unique_ptr<uint8_t, FxFreeDeleter> m_data;
    const pdfium::span<const uint8_t> m_span;
};

#endif // CORE_FXCRT_CFX_READONLYMEMORYSTREAM_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_SEEKABLESTREAMPROXY_H_
#define CORE_FXCRT_CFX_SEEKABLESTREAMPROXY_H_

#include "core/fxcrt/fx_stream.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

class CFX_SeekableStreamProxy final : public Retainable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    // Unlike IFX_SeekableStreamProxy, buffers and sizes are always in terms
    // of the number of wchar_t elementss, not bytes.
    FX_FILESIZE GetSize(); // Estimate under worst possible expansion.
    bool IsEOF();
    size_t ReadBlock(wchar_t* pStr, size_t size);

    uint16_t GetCodePage() const { return m_wCodePage; }
    void SetCodePage(uint16_t wCodePage);

private:
    enum class From {
        Begin = 0,
        Current,
    };

    explicit CFX_SeekableStreamProxy(
        const RetainPtr<IFX_SeekableReadStream>& stream);
    ~CFX_SeekableStreamProxy() override;

    FX_FILESIZE GetPosition();
    void Seek(From eSeek, FX_FILESIZE iOffset);
    size_t ReadData(uint8_t* pBuffer, size_t iBufferSize);

    uint16_t m_wCodePage;
    size_t m_wBOMLength;
    FX_FILESIZE m_iPosition;
    RetainPtr<IFX_SeekableReadStream> m_pStream;
};

#endif // CORE_FXCRT_CFX_SEEKABLESTREAMPROXY_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_TIMER_H_
#define CORE_FXCRT_CFX_TIMER_H_

#include "core/fxcrt/timerhandler_iface.h"
#include "core/fxcrt/unowned_ptr.h"

class CFX_TimerHandler;

class CFX_Timer {
public:
  class CallbackIface {
public:
    virtual ~CallbackIface() = default;
    virtual void OnTimerFired() = 0;
  };

  CFX_Timer(TimerHandlerIface* pTimerHandler,
            CallbackIface* pCallbackIface,
            int32_t nInterval);
  ~CFX_Timer();

  bool HasValidID() const {
    return m_nTimerID != TimerHandlerIface::kInvalidTimerID;
  }

private:
  static void TimerProc(int32_t idEvent);

  const int32_t m_nTimerID;
  UnownedPtr<TimerHandlerIface> const m_pTimerHandler;
  UnownedPtr<CallbackIface> const m_pCallbackIface;
};

#endif // CORE_FXCRT_CFX_TIMER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_UTF8DECODER_H_
#define CORE_FXCRT_CFX_UTF8DECODER_H_

#include "core/fxcrt/cfx_widetextbuf.h"

class CFX_UTF8Decoder {
public:
  CFX_UTF8Decoder();
  ~CFX_UTF8Decoder();

  void Input(uint8_t byte);
  void AppendCodePoint(uint32_t ch);
  void ClearStatus() { m_PendingBytes = 0; }
  WideStringView GetResult() const { return m_Buffer.AsStringView(); }

private:
  int m_PendingBytes = 0;
  uint32_t m_PendingChar = 0;
  CFX_WideTextBuf m_Buffer;
};

#endif // CORE_FXCRT_CFX_UTF8DECODER_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_UTF8ENCODER_H_
#define CORE_FXCRT_CFX_UTF8ENCODER_H_

#include <vector>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_string.h"

class CFX_UTF8Encoder {
public:
  CFX_UTF8Encoder();
  ~CFX_UTF8Encoder();

  void Input(wchar_t unicodeAsWchar);

  // The data returned by GetResult() is invalidated when this is modified by
  // appending any data.
  ByteStringView GetResult() const {
    return ByteStringView(m_Buffer.data(), m_Buffer.size());
  }

private:
  std::vector<uint8_t, FxAllocAllocator<uint8_t>> m_Buffer;
};

#endif // CORE_FXCRT_CFX_UTF8ENCODER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CFX_WIDETEXTBUF_H_
#define CORE_FXCRT_CFX_WIDETEXTBUF_H_

#include "core/fxcrt/cfx_binarybuf.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"

class CFX_WideTextBuf final : public CFX_BinaryBuf {
public:
    void AppendChar(wchar_t wch);
    size_t GetLength() const override;
    wchar_t* GetBuffer() const {
        return reinterpret_cast<wchar_t*>(m_pBuffer.get());
    }

    WideStringView AsStringView() const {
        return WideStringView(reinterpret_cast<const wchar_t*>(m_pBuffer.get()),
                               m_DataSize / sizeof(wchar_t));
    }
    WideString MakeString() const {
        return WideString(reinterpret_cast<const wchar_t*>(m_pBuffer.get()),
                           m_DataSize / sizeof(wchar_t));
    }

    void Delete(int start_index, int count) {
        CFX_BinaryBuf::Delete(start_index * sizeof(wchar_t),
                              count * sizeof(wchar_t));
    }

    CFX_WideTextBuf& operator<<(int i);
    CFX_WideTextBuf& operator<<(double f);
    CFX_WideTextBuf& operator<<(ByteStringView ascii);
    CFX_WideTextBuf& operator<<(const wchar_t* lpsz);
    CFX_WideTextBuf& operator<<(WideStringView str);
    CFX_WideTextBuf& operator<<(const WideString& str);
    CFX_WideTextBuf& operator<<(const CFX_WideTextBuf& buf);
};

#endif // CORE_FXCRT_CFX_WIDETEXTBUF_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSCOLORVALUE_H_
#define CORE_FXCRT_CSS_CFX_CSSCOLORVALUE_H_

#include "core/fxcrt/css/cfx_cssvalue.h"
#include "core/fxge/fx_dib.h"

class CFX_CSSColorValue final : public CFX_CSSValue {
public:
  explicit CFX_CSSColorValue(FX_ARGB color);
  ~CFX_CSSColorValue() override;

  FX_ARGB Value() const { return value_; }

private:
  FX_ARGB value_;
};

#endif // CORE_FXCRT_CSS_CFX_CSSCOLORVALUE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSCOMPUTEDSTYLE_H_
#define CORE_FXCRT_CSS_CFX_CSSCOMPUTEDSTYLE_H_

#include <vector>

#include "core/fxcrt/css/cfx_css.h"
#include "core/fxcrt/css/cfx_csscustomproperty.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxge/fx_dib.h"

class CFX_CSSValueList;

class CFX_CSSComputedStyle final : public Retainable {
public:
  class InheritedData {
public:
    InheritedData();
    ~InheritedData();

    CFX_CSSLength m_LetterSpacing;
    CFX_CSSLength m_WordSpacing;
    CFX_CSSLength m_TextIndent;
    RetainPtr<CFX_CSSValueList> m_pFontFamily;
    float m_fFontSize;
    float m_fLineHeight;
    FX_ARGB m_dwFontColor;
    uint16_t m_wFontWeight;
    CFX_CSSFontVariant m_eFontVariant;
    CFX_CSSFontStyle m_eFontStyle;
    CFX_CSSTextAlign m_eTextAlign;
  };

  class NonInheritedData {
public:
    NonInheritedData();

    CFX_CSSRect m_MarginWidth;
    CFX_CSSRect m_BorderWidth;
    CFX_CSSRect m_PaddingWidth;
    CFX_CSSLength m_Top;
    CFX_CSSLength m_Bottom;
    CFX_CSSLength m_Left;
    CFX_CSSLength m_Right;
    float m_fVerticalAlign;
    CFX_CSSDisplay m_eDisplay;
    CFX_CSSVerticalAlign m_eVerticalAlign;
    uint8_t m_dwTextDecoration;
    bool m_bHasMargin;
    bool m_bHasBorder;
    bool m_bHasPadding;
  };

  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  int32_t CountFontFamilies() const;
  const WideString GetFontFamily(int32_t index) const;
};
```

```
uint16_t GetFontWeight() const;
CFX_CSSFontVariant GetFontVariant() const;
CFX_CSSFontStyle GetFontStyle() const;
float GetFontSize() const;
FX_ARGB GetColor() const;
void SetFontWeight(uint16_t wFontWeight);
void SetFontVariant(CFX_CSSFontVariant eFontVariant);
void SetFontStyle(CFX_CSSFontStyle eFontStyle);
void SetFontSize(float fFontSize);
void SetColor(FX_ARGB dwFontColor);

const CFX_CSSRect* GetBorderWidth() const;
const CFX_CSSRect* GetMarginWidth() const;
const CFX_CSSRect* GetPaddingWidth() const;
void SetMarginWidth(const CFX_CSSRect& rect);
void SetPaddingWidth(const CFX_CSSRect& rect);

CFX_CSSDisplay GetDisplay() const;

float GetLineHeight() const;
const CFX_CSSLength& GetTextIndent() const;
CFX_CSSTextAlign GetTextAlign() const;
CFX_CSSVerticalAlign GetVerticalAlign() const;
float GetNumberVerticalAlign() const;
uint32_t GetTextDecoration() const;
const CFX_CSSLength& GetLetterSpacing() const;
void SetLineHeight(const CFX_CSSLength& textIndent);
void SetTextIndent(const CFX_CSSLength& textIndent);
void SetTextAlign(CFX_CSSTextAlign eTextAlign);
void SetNumberVerticalAlign(float fAlign);
void SetTextDecoration(uint32_t dwTextDecoration);
void SetLetterSpacing(const CFX_CSSLength& letterSpacing);
void AddCustomStyle(const CFX_CSSCustomProperty& prop);

bool GetCustomStyle(const WideString& wsName, WideString* pValue) const;

InheritedData m_InheritedData;
NonInheritedData m_NonInheritedData;

private:
CFX_CSSComputedStyle();
~CFX_CSSComputedStyle() override;

std::vector<CFX_CSSCustomProperty> m_CustomProperties;
};

#endif // CORE_FXCRT_CSS_CFX_CSSCOMPUTEDSTYLE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSCUSTOMPROPERTY_H_
#define CORE_FXCRT_CSS_CFX_CSSCUSTOMPROPERTY_H_

#include "core/fxcrt/fx_string.h"

class CFX_CSSCustomProperty {
public:
  CFX_CSSCustomProperty(const WideString& name, const WideString& value);
  CFX_CSSCustomProperty(const CFX_CSSCustomProperty& prop);
  ~CFX_CSSCustomProperty();

  WideString name() const { return name_; }
  WideString value() const { return value_; }

private:
  WideString name_;
  WideString value_;
};

#endif // CORE_FXCRT_CSS_CFX_CSSCUSTOMPROPERTY_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSDATA_H_
#define CORE_FXCRT_CSS_CFX_CSSDATA_H_

#include "core/fxcrt/css/cfx_css.h"
#include "core/fxcrt/css/cfx_cssnumbervalue.h"
#include "core/fxcrt/css/cfx_cssvalue.h"
#include "core/fxcrt/string_view_template.h"
#include "core/fxge/fx_dib.h"

class CFX_CSSData {
public:
    struct Property {
        CFX_CSSProperty eName;
        uint32_t dwHash; // Hashed as wide string.
        uint32_t dwType;
    };

    struct PropertyValue {
        CFX_CSSPropertyValue eName;
        uint32_t dwHash; // Hashed as wide string.
    };

    struct LengthUnit {
        const wchar_t* value;
        CFX_CSSNumberType type;
    };

    struct Color {
        const wchar_t* name;
        FX_ARGB value;
    };

    static const Property* GetPropertyByName(WideStringView name);
    static const Property* GetPropertyByEnum(CFX_CSSProperty property);
    static const PropertyValue* GetPropertyValueByName(WideStringView wsName);
    static const LengthUnit* GetLengthUnitByName(WideStringView wsName);
    static const Color* GetColorByName(WideStringView wsName);
};

#endif // CORE_FXCRT_CSS_CFX_CSSDATA_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSDECLARATION_H_
#define CORE_FXCRT_CSS_CFX_CSSDECLARATION_H_

#include <memory>
#include <vector>

#include "core/fxcrt/css/cfx_cssdata.h"

class CFX_CSSPropertyHolder;
class CFX_CSSCustomProperty;

class CFX_CSSDeclaration {
public:
    using const_prop_iterator =
        std::vector<std::unique_ptr<CFX_CSSPropertyHolder>>::const_iterator;
    using const_custom_iterator =
        std::vector<std::unique_ptr<CFX_CSSCustomProperty>>::const_iterator;

    static bool ParseCSSString(const wchar_t* pszValue,
                              int32_t iValueLen,
                              int32_t iOffset,
                              int32_t* iLength);
    static bool ParseCSSColor(const wchar_t* pszValue,
                              int32_t iValueLen,
                              FX_ARGB* dwColor);

    CFX_CSSDeclaration();
    ~CFX_CSSDeclaration();

    RetainPtr<CFX_CSSValue> GetProperty(CFX_CSSProperty eProperty,
                                       bool* bImportant) const;

    const_prop_iterator begin() const { return properties_.begin(); }
    const_prop_iterator end() const { return properties_.end(); }

    const_custom_iterator custom_begin() const {
        return custom_properties_.begin();
    }
    const_custom_iterator custom_end() const { return custom_properties_.end(); }

    bool empty() const { return properties_.empty(); }

    void AddProperty(const CFX_CSSData::Property* property, WideStringView value);
    void AddProperty(const WideString& prop, const WideString& value);

    size_t PropertyCountForTesting() const;

    FX_ARGB ParseColorForTest(const wchar_t* pszValue,
                              int32_t iValueLen,
                              FX_ARGB* dwColor) const;

private:
    void ParseFontProperty(const wchar_t* pszValue,
                          int32_t iValueLen,
                          bool bImportant);
    bool ParseBorderProperty(const wchar_t* pszValue,
                            int32_t iValueLen,
```

```
        RetainPtr<CFX_CSSValue>& pWidth) const;
void ParseValueListProperty(const CFX_CSSData::Property* pProperty,
        const wchar_t* pszValue,
        int32_t iValueLen,
        bool bImportant);
void Add4ValuesProperty(const std::vector<RetainPtr<CFX_CSSValue>>& list,
        bool bImportant,
        CFX_CSSProperty eLeft,
        CFX_CSSProperty eTop,
        CFX_CSSProperty eRight,
        CFX_CSSProperty eBottom);
RetainPtr<CFX_CSSValue> ParseNumber(const wchar_t* pszValue,
        int32_t iValueLen);
RetainPtr<CFX_CSSValue> ParseEnum(const wchar_t* pszValue, int32_t iValueLen);
RetainPtr<CFX_CSSValue> ParseColor(const wchar_t* pszValue,
        int32_t iValueLen);
RetainPtr<CFX_CSSValue> ParseString(const wchar_t* pszValue,
        int32_t iValueLen);
void AddPropertyHolder(CFX_CSSProperty eProperty,
        RetainPtr<CFX_CSSValue> pValue,
        bool bImportant);

    std::vector<std::unique_ptr<CFX_CSSPropertyHolder>> properties_;
    std::vector<std::unique_ptr<CFX_CSSCustomProperty>> custom_properties_;
};

#endif // CORE_FXCRT_CSS_CFX_CSSDECLARATION_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSENUMVALUE_H_
#define CORE_FXCRT_CSS_CFX_CSSENUMVALUE_H_

#include "core/fxcrt/css/cfx_cssvalue.h"

class CFX_CSSEnumValue final : public CFX_CSSValue {
public:
  explicit CFX_CSSEnumValue(CFX_CSSPropertyValue value);
  ~CFX_CSSEnumValue() override;

  CFX_CSSPropertyValue Value() const { return value_; }

private:
  CFX_CSSPropertyValue value_;
};

#endif // CORE_FXCRT_CSS_CFX_CSSENUMVALUE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSEXTTEXTBUF_H_
#define CORE_FXCRT_CSS_CFX_CSSEXTTEXTBUF_H_

#include "core/fxcrt/fx_system.h"

class CFX_CSSExtTextBuf {
public:
  CFX_CSSExtTextBuf();
  ~CFX_CSSExtTextBuf();

  void AttachBuffer(const wchar_t* pBuffer, int32_t iBufLen);

  bool IsEOF() const { return m_iDatPos >= m_iDatLen; }

  wchar_t GetChar() const { return m_pExtBuffer[m_iDatPos]; }
  wchar_t GetNextChar() const {
    return (m_iDatPos + 1 >= m_iDatLen) ? 0 : m_pExtBuffer[m_iDatPos + 1];
  }

  void MoveNext() { m_iDatPos++; }

protected:
  const wchar_t* m_pExtBuffer;
  int32_t m_iDatLen;
  int32_t m_iDatPos;
};

#endif // CORE_FXCRT_CSS_CFX_CSSEXTTEXTBUF_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSS_H
#define CORE_FXCRT_CSS_CFX_CSS_H

#include <stdint.h>

enum CFX_CSSVALUETYPE {
    CFX_CSSVALUETYPE_Primitive = 1 << 0,
    CFX_CSSVALUETYPE_List = 1 << 1,
    CFX_CSSVALUETYPE_Shorthand = 1 << 2,
    // Note the values below this comment must be > 0x0F so we can mask the above.
    CFX_CSSVALUETYPE_MaybeNumber = 1 << 4,
    CFX_CSSVALUETYPE_MaybeEnum = 1 << 5,
    CFX_CSSVALUETYPE_MaybeString = 1 << 7,
    CFX_CSSVALUETYPE_MaybeColor = 1 << 8
};

enum class CFX_CSSPrimitiveType : uint8_t {
    Unknown = 0,
    Number,
    String,
    RGB,
    Enum,
    Function,
    List,
};

// Any entries added/removed here, will need to be mirrored in
// propertyValueTable, in core/fxcrt/css/cfx_cssdata.cpp.
enum class CFX_CSSPropertyValue : uint8_t {
    Bolder = 0,
    None,
    Dot,
    Sub,
    Top,
    Right,
    Normal,
    Auto,
    Text,
    XSmall,
    Thin,
    Small,
    Bottom,
    Underline,
    Double,
    Lighter,
    Oblique,
    Super,
    Center,
    XxLarge,
    Smaller,
    Baseline,
    Thick,
    Justify,
    Middle,
    Medium,
    ListItem,
    XxSmall,
```

```
Bold,  
SmallCaps,  
Inline,  
Overline,  
TextBottom,  
Larger,  
InlineTable,  
InlineBlock,  
Blink,  
Block,  
Italic,  
LineThrough,  
XLarge,  
Large,  
Left,  
TextTop,  
};  
  
// Any entries added/removed here, will need to be mirrored in  
// propertyTable, in core/fxcrt/css/cfx_cssdata.cpp.  
enum class CFX_CSSProperty : uint8_t {  
    BorderLeft = 0,  
    Top,  
    Margin,  
    TextIndent,  
    Right,  
    PaddingLeft,  
    MarginLeft,  
    Border,  
    BorderTop,  
    Bottom,  
    PaddingRight,  
    BorderBottom,  
    FontFamily,  
    FontWeight,  
    Color,  
    LetterSpacing,  
    TextAlign,  
    BorderRightWidth,  
    VerticalAlign,  
    PaddingTop,  
    FontVariant,  
    BorderWidth,  
    BorderBottomWidth,  
    BorderRight,  
    FontSize,  
    BorderSpacing,  
    FontStyle,  
    Font,  
    LineHeight,  
    MarginRight,  
    BorderLeftWidth,  
    Display,  
    PaddingBottom,  
    BorderTopWidth,  
    WordSpacing,  
    Left,  
    TextDecoration,  
    Padding,  
    MarginBottom,  
    MarginTop,  
};
```

```
enum class CFX_CSSSelectorType : uint8_t { Element = 0, Descendant };
```

```
enum class CFX_CSSLengthUnit : uint8_t {  
    Auto,  
    None,  
    Normal,  
    Point,  
    Percent,  
};
```

```
enum class CFX_CSSDisplay : uint8_t {  
    None,  
    ListItem,  
    Block,  
    Inline,  
    InlineBlock,  
    InlineTable,  
};
```

```
enum class CFX_CSSFontStyle : uint8_t {  
    Normal,  
    Italic,  
};
```

```
enum class CFX_CSSTextAlign : uint8_t {  
    Left,  
    Right,  
    Center,  
    Justify,  
    JustifyAll,  
};
```

```
enum class CFX_CSSVerticalAlign : uint8_t {  
    Baseline,  
    Sub,  
    Super,  
    Top,  
    TextTop,  
    Middle,  
    Bottom,  
    TextBottom,  
    Number,  
};
```

```
enum class CFX_CSSFontVariant : uint8_t {  
    Normal,  
    SmallCaps,  
};
```

```
enum CFX_CSSTEXTDECORATION {  
    CFX_CSSTEXTDECORATION_None = 0,  
    CFX_CSSTEXTDECORATION_Underline = 1 << 0,  
    CFX_CSSTEXTDECORATION_Overline = 1 << 1,  
    CFX_CSSTEXTDECORATION_LineThrough = 1 << 2,  
    CFX_CSSTEXTDECORATION_Blink = 1 << 3,  
    CFX_CSSTEXTDECORATION_Double = 1 << 4,  
};
```

```
class CFX_CSSLength {  
public:  
    CFX_CSSLength() {}  
  
    CFX_CSSLength(CFX_CSSLengthUnit eUnit, float fValue)
```

```
    : m_unit(eUnit), m_fValue(fValue) {}

CFX_CSSLength& Set(CFX_CSSLengthUnit eUnit) {
    m_unit = eUnit;
    return *this;
}

CFX_CSSLength& Set(CFX_CSSLengthUnit eUnit, float fValue) {
    m_unit = eUnit;
    m_fValue = fValue;
    return *this;
}

CFX_CSSLengthUnit GetUnit() const { return m_unit; }

float GetValue() const { return m_fValue; }
bool NonZero() const { return static_cast<int>(m_fValue) != 0; }

private:
    CFX_CSSLengthUnit m_unit;
    float m_fValue;
};

class CFX_CSSRect {
public:
    CFX_CSSRect() {}

    CFX_CSSRect(CFX_CSSLengthUnit eUnit, float val)
        : left(eUnit, val),
          top(eUnit, val),
          right(eUnit, val),
          bottom(eUnit, val) {}

    CFX_CSSRect& Set(CFX_CSSLengthUnit eUnit) {
        left.Set(eUnit);
        top.Set(eUnit);
        right.Set(eUnit);
        bottom.Set(eUnit);
        return *this;
    }

    CFX_CSSRect& Set(CFX_CSSLengthUnit eUnit, float fValue) {
        left.Set(eUnit, fValue);
        top.Set(eUnit, fValue);
        right.Set(eUnit, fValue);
        bottom.Set(eUnit, fValue);
        return *this;
    }

    CFX_CSSLength left;
    CFX_CSSLength top;
    CFX_CSSLength right;
    CFX_CSSLength bottom;
};

#endif // CORE_FXCRT_CSS_CFX_CSS_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSNUMBERVALUE_H_
#define CORE_FXCRT_CSS_CFX_CSSNUMBERVALUE_H_

#include "core/fxcrt/css/cfx_cssvalue.h"
#include "core/fxcrt/fx_system.h"

enum class CFX_CSSNumberType {
    Number,
    Percent,
    EMS,
    EXS,
    Pixels,
    CentiMeters,
    MilliMeters,
    Inches,
    Points,
    Picas,
};

class CFX_CSSNumberValue final : public CFX_CSSValue {
public:
    CFX_CSSNumberValue(CFX_CSSNumberType type, float value);
    ~CFX_CSSNumberValue() override;

    float Value() const { return value_; }
    CFX_CSSNumberType Kind() const { return type_; }

    float Apply(float percentBase) const;

private:
    CFX_CSSNumberType type_;
    float value_;
};

#endif // CORE_FXCRT_CSS_CFX_CSSNUMBERVALUE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSPROPERTYHOLDER_H_
#define CORE_FXCRT_CSS_CFX_CSSPROPERTYHOLDER_H_

#include "core/fxcrt/css/cfx_css.h"
#include "core/fxcrt/css/cfx_cssvalue.h"
#include "core/fxcrt/retain_ptr.h"

class CFX_CSSPropertyHolder {
public:
  CFX_CSSPropertyHolder();
  ~CFX_CSSPropertyHolder();

  CFX_CSSProperty eProperty;
  bool bImportant;
  RetainPtr<CFX_CSSValue> pValue;
};

#endif // CORE_FXCRT_CSS_CFX_CSSPROPERTYHOLDER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSRULECOLLECTION_H_
#define CORE_FXCRT_CSS_CFX_CSSRULECOLLECTION_H_

#include <map>
#include <memory>
#include <vector>

#include "core/fxcrt/fx_string.h"

class CFX_CSSDeclaration;
class CFX_CSSSelector;
class CFX_CSSStyleRule;
class CFX_CSSStyleSheet;

class CFX_CSSRuleCollection {
public:
  class Data {
public:
    Data(CFX_CSSSelector* pSel, CFX_CSSDeclaration* pDecl);

    CFX_CSSSelector* const pSelector;
    CFX_CSSDeclaration* const pDeclaration;
  };

  CFX_CSSRuleCollection();
  ~CFX_CSSRuleCollection();

  void AddRulesFrom(const CFX_CSSStyleSheet* sheet);
  void Clear();
  int32_t CountSelectors() const { return m_iSelectors; }

  const std::vector<std::unique_ptr<Data>>* GetTagRuleData(
    const WideString& tagname) const;

private:
  void AddRulesFrom(const CFX_CSSStyleSheet* pStyleSheet,
    CFX_CSSStyleRule* pRule);

  std::map<uint32_t, std::vector<std::unique_ptr<Data>>> m_TagRules;
  int32_t m_iSelectors;
};

#endif // CORE_FXCRT_CSS_CFX_CSSRULECOLLECTION_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSSELECTOR_H_
#define CORE_FXCRT_CSS_CFX_CSSSELECTOR_H_

#include <memory>
#include <utility>

#include "core/fxcrt/css/cfx_css.h"
#include "core/fxcrt/fx_string.h"

class CFX_CSSSelector {
public:
    static std::unique_ptr<CFX_CSSSelector> FromString(WideStringView str);

    CFX_CSSSelector(CFX_CSSSelectorType eType,
                   const wchar_t* psz,
                   int32_t iLen,
                   bool bIgnoreCase);
    ~CFX_CSSSelector();

    CFX_CSSSelectorType GetType() const;
    uint32_t GetNameHash() const;
    CFX_CSSSelector* GetNextSelector() const;

    void SetNext(std::unique_ptr<CFX_CSSSelector> pNext) {
        m_pNext = std::move(pNext);
    }

private:
    void SetType(CFX_CSSSelectorType eType) { m_eType = eType; }

    CFX_CSSSelectorType m_eType;
    uint32_t m_dwHash;
    std::unique_ptr<CFX_CSSSelector> m_pNext;
};

#endif // CORE_FXCRT_CSS_CFX_CSSSELECTOR_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSSTRINGVALUE_H_
#define CORE_FXCRT_CSS_CFX_CSSSTRINGVALUE_H_

#include "core/fxcrt/css/cfx_cssvalue.h"
#include "core/fxcrt/widestring.h"

class CFX_CSSStringValue final : public CFX_CSSValue {
public:
  explicit CFX_CSSStringValue(const WideString& value);
  ~CFX_CSSStringValue() override;

  const WideString Value() const { return value_; }

private:
  const WideString value_;
};

#endif // CORE_FXCRT_CSS_CFX_CSSSTRINGVALUE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSSTYLERULE_H_
#define CORE_FXCRT_CSS_CFX_CSSSTYLERULE_H_

#include <memory>
#include <vector>

#include "core/fxcrt/css/cfx_cssdeclaration.h"
#include "core/fxcrt/css/cfx_cssselector.h"

class CFX_CSSStyleRule {
public:
    CFX_CSSStyleRule();
    ~CFX_CSSStyleRule();

    size_t CountSelectorLists() const;
    CFX_CSSSelector* GetSelectorList(int32_t index) const;
    CFX_CSSDeclaration* GetDeclaration();

    void SetSelector(std::vector<std::unique_ptr<CFX_CSSSelector>>* list);

private:
    CFX_CSSDeclaration m_Declaration;
    std::vector<std::unique_ptr<CFX_CSSSelector>> m_ppSelector;
};

#endif // CORE_FXCRT_CSS_CFX_CSSSTYLERULE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSSTYLESELECTOR_H_
#define CORE_FXCRT_CSS_CFX_CSSSTYLESELECTOR_H_

#include <memory>
#include <vector>

#include "core/fxcrt/css/cfx_css.h"
#include "core/fxcrt/css/cfx_cssrulecollection.h"
#include "core/fxcrt/fx_system.h"

class CFX_CSSComputedStyle;
class CFX_CSSCustomProperty;
class CFX_CSSDeclaration;
class CFX_CSSPropertyHolder;
class CFX_CSSSelector;
class CFX_CSSStyleSheet;
class CFX_CSSValue;
class CFX_CSSValueList;

class CFX_CSSStyleSelector {
public:
    CFX_CSSStyleSelector();
    ~CFX_CSSStyleSelector();

    void SetDefFontSize(float fFontSize);
    void SetUAStyleSheet(std::unique_ptr<CFX_CSSStyleSheet> pSheet);
    void UpdateStyleIndex();

    RetainPtr<CFX_CSSComputedStyle> CreateComputedStyle(
        CFX_CSSComputedStyle* pParentStyle);

    // Note, the dest style has to be an out param because the CXFA_TextParser
    // adds non-inherited data from the parent style. Attempting to copy
    // internally will fail as you'll lose the non-inherited data.
    void ComputeStyle(const std::vector<const CFX_CSSDeclaration*>& declArray,
        const WideString& styleString,
        const WideString& alignString,
        CFX_CSSComputedStyle* pDestStyle);

    std::vector<const CFX_CSSDeclaration*> MatchDeclarations(
        const WideString& tagname);

private:
    bool MatchSelector(const WideString& tagname, CFX_CSSSelector* pSel);

    void AppendInlineStyle(CFX_CSSDeclaration* pDecl, const WideString& style);
    void ApplyDeclarations(
        const std::vector<const CFX_CSSDeclaration*>& declArray,
        const CFX_CSSDeclaration* extraDecl,
        CFX_CSSComputedStyle* pDestStyle);
    void ApplyProperty(CFX_CSSProperty eProperty,
        const RetainPtr<CFX_CSSValue>& pValue,
        CFX_CSSComputedStyle* pComputedStyle);
    void ExtractValues(const CFX_CSSDeclaration* decl,
        std::vector<const CFX_CSSPropertyHolder*>* importants,
        std::vector<const CFX_CSSPropertyHolder*>* normals,
        std::vector<const CFX_CSSCustomProperty*>* custom);
};
```

```
bool SetLengthWithPercent(CFX_CSSLength& width,
                          CFX_CSSPrimitiveType eType,
                          const RetainPtr<CFX_CSSValue>& pValue,
                          float fFontSize);
float ToFontSize(CFX_CSSPropertyValue eValue, float fCurFontSize);
CFX_CSSDisplay ToDisplay(CFX_CSSPropertyValue eValue);
CFX_CSSTextAlign ToTextAlign(CFX_CSSPropertyValue eValue);
uint16_t ToFontWeight(CFX_CSSPropertyValue eValue);
CFX_CSSFontStyle ToFontStyle(CFX_CSSPropertyValue eValue);
CFX_CSSVerticalAlign ToVerticalAlign(CFX_CSSPropertyValue eValue);
uint32_t ToTextDecoration(const RetainPtr<CFX_CSSValueList>& pList);
CFX_CSSFontVariant ToFontVariant(CFX_CSSPropertyValue eValue);

float m_fDefFontSize;
std::unique_ptr<CFX_CSSStyleSheet> m_UAStyles;
CFX_CSSRuleCollection m_UARules;
};

#endif // CORE_FXCRT_CSS_CFX_CSSSTYLESELECTOR_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSSTYLE SHEET_H_
#define CORE_FXCRT_CSS_CFX_CSSSTYLE SHEET_H_

#include <map>
#include <memory>
#include <vector>

#include "core/fxcrt/css/cfx_csssyntaxparser.h"
#include "core/fxcrt/fx_string.h"

class CFX_CSSStyleRule;

class CFX_CSSStyleSheet {
public:
    CFX_CSSStyleSheet();
    ~CFX_CSSStyleSheet();

    bool LoadBuffer(const wchar_t* pBuffer, int32_t iBufSize);

    int32_t CountRules() const;
    CFX_CSSStyleRule* GetRule(int32_t index) const;

private:
    void Reset();
    CFX_CSSSyntaxStatus LoadStyleRule(
        CFX_CSSSyntaxParser* pSyntax,
        std::vector<std::unique_ptr<CFX_CSSStyleRule>>* ruleArray);
    void SkipRuleSet(CFX_CSSSyntaxParser* pSyntax);

    std::vector<std::unique_ptr<CFX_CSSStyleRule>> m_RuleArray;
    std::map<uint32_t, wchar_t*> m_StringCache;
};

#endif // CORE_FXCRT_CSS_CFX_CSSSTYLE SHEET_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FXCRT_CSS_CFX_CSSSYNTAXPARSER_H  
#define CORE_FXCRT_CSS_CFX_CSSSYNTAXPARSER_H
```

```
#include <stack>
```

```
#include "core/fxcrt/css/cfx_cssextextbuf.h"  
#include "core/fxcrt/css/cfx_csstextbuf.h"  
#include "core/fxcrt/fx_string.h"
```

```
#define CFX_CSSSYNTAXCHECK_AllowCharset 1  
#define CFX_CSSSYNTAXCHECK_AllowImport 2
```

```
enum class CFX_CSSSyntaxMode {  
    RuleSet,  
    Comment,  
    UnknownRule,  
    Selector,  
    PropertyName,  
    PropertyValue,  
};
```

```
enum class CFX_CSSSyntaxStatus : uint8_t {  
    Error,  
    EOS,  
    None,  
    StyleRule,  
    Selector,  
    DeclOpen,  
    DeclClose,  
    PropertyName,  
    PropertyValue,  
};
```

```
class CFX_CSSSyntaxParser {  
public:  
    CFX_CSSSyntaxParser(const wchar_t* pBuffer, int32_t iBufferSize);  
    CFX_CSSSyntaxParser(const wchar_t* pBuffer,  
                        int32_t iBufferSize,  
                        int32_t iTextDatSize,  
                        bool bOnlyDeclaration);  
    ~CFX_CSSSyntaxParser();  
  
    CFX_CSSSyntaxStatus DoSyntaxParse();  
    WideStringView GetCurrentString() const;  
  
protected:  
    void SwitchMode(CFX_CSSSyntaxMode eMode);  
    int32_t SwitchToComment();  
  
    bool RestoreMode();  
    bool AppendChar(wchar_t wch);  
    int32_t SaveTextData();  
    bool IsCharsetEnabled() const {  
        return (m_dwCheck & CFX_CSSSYNTAXCHECK_AllowCharset) != 0;  
    }  
    void DisableCharset() { m_dwCheck = CFX_CSSSYNTAXCHECK_AllowImport; }  
    bool IsImportEnabled() const;
```

```
void DisableImport() { m_dwCheck = 0; }

CFX_CSSTextBuf m_TextData;
CFX_CSSExtTextBuf m_TextPlane;
int32_t m_iTextDataLen;
uint32_t m_dwCheck;
CFX_CSSSyntaxMode m_eMode;
CFX_CSSSyntaxStatus m_eStatus;
std::stack<CFX_CSSSyntaxMode> m_ModeStack;
};

#endif // CORE_FXCRT_CSS_CFX_CSSSYNTAXPARSER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSTEXTBUF_H_
#define CORE_FXCRT_CSS_CFX_CSSTEXTBUF_H_

#include "core/fxcrt/fx_system.h"

class CFX_CSSTextBuf {
public:
  CFX_CSSTextBuf();
  ~CFX_CSSTextBuf();

  void InitWithSize(int32_t iAllocSize);
  void AppendChar(wchar_t wch);

  void Clear() { m_iDatLen = 0; }

  int32_t TrimEnd();

  int32_t GetLength() const { return m_iDatLen; }
  const wchar_t* GetBuffer() const { return m_pBuffer; }

protected:
  void ExpandBuf(int32_t iDesiredSize);

  wchar_t* m_pBuffer;
  int32_t m_iBufLen;
  int32_t m_iDatLen;
};

#endif // CORE_FXCRT_CSS_CFX_CSSTEXTBUF_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSVALUE_H_
#define CORE_FXCRT_CSS_CFX_CSSVALUE_H_

#include "core/fxcrt/css/cfx_css.h"
#include "core/fxcrt/retain_ptr.h"

class CFX_CSSValue : public Retainable {
public:
    CFX_CSSPrimitiveType GetType() const { return m_value; }

protected:
    explicit CFX_CSSValue(CFX_CSSPrimitiveType type);

private:
    CFX_CSSPrimitiveType m_value;
};

#endif // CORE_FXCRT_CSS_CFX_CSSVALUE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSVALUELIST_H_
#define CORE_FXCRT_CSS_CFX_CSSVALUELIST_H_

#include <vector>

#include "core/fxcrt/css/cfx_cssvalue.h"

class CFX_CSSValueList final : public CFX_CSSValue {
public:
  explicit CFX_CSSValueList(std::vector<RetainPtr<CFX_CSSValue>>& list);
  ~CFX_CSSValueList() override;

  int32_t CountValues() const;
  RetainPtr<CFX_CSSValue> GetValue(int32_t index) const;

private:
  std::vector<RetainPtr<CFX_CSSValue>> m_ppList;
};

#endif // CORE_FXCRT_CSS_CFX_CSSVALUELIST_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_CSS_CFX_CSSVALUELISTPARSER_H_
#define CORE_FXCRT_CSS_CFX_CSSVALUELISTPARSER_H_

#include "core/fxcrt/css/cfx_css.h"
#include "core/fxcrt/fx_system.h"

class CFX_CSSValueListParser {
public:
  CFX_CSSValueListParser(const wchar_t* psz, int32_t iLen, wchar_t separator);

  bool NextValue(CFX_CSSPrimitiveType* eType,
                const wchar_t** pStart,
                int32_t* iLength);
  void UseCommaSeparator() { m_Separator = ','; }

private:
  int32_t SkipTo(wchar_t wch, bool breakOnSpace, bool matchBrackets);

  wchar_t m_Separator;
  const wchar_t* m_pCur;
  const wchar_t* m_pEnd;
};

#endif // CORE_FXCRT_CSS_CFX_CSSVALUELISTPARSER_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_FILEACCESS_IFACE_H_
#define CORE_FXCRT_FILEACCESS_IFACE_H_

#include <memory>

#include "core/fxcrt/fx_safe_types.h"
#include "core/fxcrt/fx_string.h"

class FileAccessIface {
public:
    static std::unique_ptr<FileAccessIface> Create();
    virtual ~FileAccessIface() = default;

    virtual bool Open(ByteStringView fileName, uint32_t dwMode) = 0;
    virtual bool Open(WideStringView fileName, uint32_t dwMode) = 0;
    virtual void Close() = 0;
    virtual FX_FILESIZE GetSize() const = 0;
    virtual FX_FILESIZE GetPosition() const = 0;
    virtual FX_FILESIZE SetPosition(FX_FILESIZE pos) = 0;
    virtual size_t Read(void* pBuffer, size_t szBuffer) = 0;
    virtual size_t Write(const void* pBuffer, size_t szBuffer) = 0;
    virtual size_t ReadPos(void* pBuffer, size_t szBuffer, FX_FILESIZE pos) = 0;
    virtual size_t WritePos(const void* pBuffer,
                            size_t szBuffer,
                            FX_FILESIZE pos) = 0;
    virtual bool Flush() = 0;
    virtual bool Truncate(FX_FILESIZE szFile) = 0;
};

#endif // CORE_FXCRT_FILEACCESS_IFACE_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_FX_BIDI_H_
#define CORE_FXCRT_FX_BIDI_H_

#include <vector>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"

// Processes characters and group them into segments based on text direction.
class CFX_BidiChar {
public:
    enum Direction { NEUTRAL, LEFT, RIGHT };
    struct Segment {
        int32_t start;           // Start position.
        int32_t count;          // Character count.
        Direction direction;     // Segment direction.
    };

    CFX_BidiChar();

    // Append a character and classify it as left, right, or neutral.
    // Returns true if the character has a different direction than the
    // existing direction to indicate there is a segment to process.
    bool AppendChar(wchar_t wch);

    // Call this after the last character has been appended. AppendChar()
    // must not be called after this.
    // Returns true if there is still a segment to process.
    bool EndChar();

    // Call after a change in direction is indicated by the above to get
    // information about the segment to process.
    const Segment& GetSegmentInfo() const { return m_LastSegment; }

private:
    void StartNewSegment(CFX_BidiChar::Direction direction);

    Segment m_CurrentSegment;
    Segment m_LastSegment;
};

class CFX_BidiString {
public:
    using const_iterator = std::vector<CFX_BidiChar::Segment>::const_iterator;

    explicit CFX_BidiString(const WideString& str);
    ~CFX_BidiString();

    // Overall direction is always LEFT or RIGHT, never NEUTRAL.
    CFX_BidiChar::Direction OverallDirection() const;

    // Force the overall direction to be R2L regardless of what was detected.
    void SetOverallDirectionRight();

    const_iterator begin() const { return m_Order.begin(); }
    const_iterator end() const { return m_Order.end(); }
};
```

```
private:
  const WideString& m_Str;
  std::vector<CFX_BidiChar::Segment> m_Order;
  CFX_BidiChar::Direction m_OverallDirection = CFX_BidiChar::LEFT;
};

#endif // CORE_FXCRT_FX_BIDI_H
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FXCRT_FX_CODEPAGE_H
```

```
#define CORE_FXCRT_FX_CODEPAGE_H
```

```
#include <stdint.h>
```

```
#define FX_CODEPAGE_DefANSI 0
```

```
#define FX_CODEPAGE_Symbol 42
```

```
#define FX_CODEPAGE_MSDOS_US 437
```

```
#define FX_CODEPAGE_Arabic_ASMO708 708
```

```
#define FX_CODEPAGE_MSDOS_Greek1 737
```

```
#define FX_CODEPAGE_MSDOS_Baltic 775
```

```
#define FX_CODEPAGE_MSDOS_WesternEuropean 850
```

```
#define FX_CODEPAGE_MSDOS_EasternEuropean 852
```

```
#define FX_CODEPAGE_MSDOS_Cyrillic 855
```

```
#define FX_CODEPAGE_MSDOS_Turkish 857
```

```
#define FX_CODEPAGE_MSDOS_Portuguese 860
```

```
#define FX_CODEPAGE_MSDOS_Icelandic 861
```

```
#define FX_CODEPAGE_MSDOS_Hebrew 862
```

```
#define FX_CODEPAGE_MSDOS_FrenchCanadian 863
```

```
#define FX_CODEPAGE_MSDOS_Arabic 864
```

```
#define FX_CODEPAGE_MSDOS_Norwegian 865
```

```
#define FX_CODEPAGE_MSDOS_Russian 866
```

```
#define FX_CODEPAGE_MSDOS_Greek2 869
```

```
#define FX_CODEPAGE_MSDOS_Thai 874
```

```
#define FX_CODEPAGE_ShiftJIS 932
```

```
#define FX_CODEPAGE_ChineseSimplified 936
```

```
#define FX_CODEPAGE_Hangul 949
```

```
#define FX_CODEPAGE_ChineseTraditional 950
```

```
#define FX_CODEPAGE_UTF16LE 1200
```

```
#define FX_CODEPAGE_UTF16BE 1201
```

```
#define FX_CODEPAGE_MSWin_EasternEuropean 1250
```

```
#define FX_CODEPAGE_MSWin_Cyrillic 1251
```

```
#define FX_CODEPAGE_MSWin_WesternEuropean 1252
```

```
#define FX_CODEPAGE_MSWin_Greek 1253
```

```
#define FX_CODEPAGE_MSWin_Turkish 1254
```

```
#define FX_CODEPAGE_MSWin_Hebrew 1255
```

```
#define FX_CODEPAGE_MSWin_Arabic 1256
```

```
#define FX_CODEPAGE_MSWin_Baltic 1257
```

```
#define FX_CODEPAGE_MSWin_Vietnamese 1258
```

```
#define FX_CODEPAGE_Johab 1361
```

```
#define FX_CODEPAGE_MAC_Roman 10000
```

```
#define FX_CODEPAGE_MAC_ShiftJIS 10001
```

```
#define FX_CODEPAGE_MAC_ChineseTraditional 10002
```

```
#define FX_CODEPAGE_MAC_Korean 10003
```

```
#define FX_CODEPAGE_MAC_Arabic 10004
```

```
#define FX_CODEPAGE_MAC_Hebrew 10005
```

```
#define FX_CODEPAGE_MAC_Greek 10006
```

```
#define FX_CODEPAGE_MAC_Cyrillic 10007
```

```
#define FX_CODEPAGE_MAC_ChineseSimplified 10008
```

```
#define FX_CODEPAGE_MAC_Thai 10021
```

```
#define FX_CODEPAGE_MAC_EasternEuropean 10029
```

```
#define FX_CODEPAGE_MAC_Turkish 10081
```

```
#define FX_CODEPAGE_UTF8 65001
```

```
#define FX_CHARSET_ANSI 0
```

```
#define FX_CHARSET_Default 1
```

```
#define FX_CHARSET_Symbol 2
```

```
#define FX_CHARSET_MAC_Roman 77
#define FX_CHARSET_MAC_ShiftJIS 78
#define FX_CHARSET_MAC_Korean 79
#define FX_CHARSET_MAC_ChineseSimplified 80
#define FX_CHARSET_MAC_ChineseTraditional 81
#define FX_CHARSET_MAC_Hebrew 83
#define FX_CHARSET_MAC_Arabic 84
#define FX_CHARSET_MAC_Greek 85
#define FX_CHARSET_MAC_Turkish 86
#define FX_CHARSET_MAC_Thai 87
#define FX_CHARSET_MAC_EasternEuropean 88
#define FX_CHARSET_MAC_Cyrillic 89
#define FX_CHARSET_ShiftJIS 128
#define FX_CHARSET_Hangul 129
#define FX_CHARSET_Johab 130
#define FX_CHARSET_ChineseSimplified 134
#define FX_CHARSET_ChineseTraditional 136
#define FX_CHARSET_MSWin_Greek 161
#define FX_CHARSET_MSWin_Turkish 162
#define FX_CHARSET_MSWin_Vietnamese 163
#define FX_CHARSET_MSWin_Hebrew 177
#define FX_CHARSET_MSWin_Arabic 178
#define FX_CHARSET_MSWin_Baltic 186
#define FX_CHARSET_MSWin_Cyrillic 204
#define FX_CHARSET_Thai 222
#define FX_CHARSET_MSWin_EasternEuropean 238
#define FX_CHARSET_US 254
#define FX_CHARSET_OEM 255

// Hi-bytes to unicode codepoint mapping for various code pages.
struct FX_CharsetUnicodes {
    uint8_t m_Charset;
    const uint16_t* m_pUnicodes;
};

extern const FX_CharsetUnicodes g_FX_CharsetUnicodes[8];

uint16_t FX_GetCodePageFromCharset(uint8_t charset);
uint8_t FX_GetCharsetFromCodePage(uint16_t codepage);
bool FX_CharSetIsCJK(uint8_t uCharset);

#endif // CORE_FXCRT_FX_CODEPAGE_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_FX_COORDINATES_H_
#define CORE_FXCRT_FX_COORDINATES_H_

#include <algorithm>

#include "core/fxcrt/fx_system.h"
#include "third_party/base/numerics/safe_math.h"

#ifndef NDEBUG
#include <ostream>
#endif

template <class BaseType>
class CFX_PTemplate {
public:
    CFX_PTemplate() : x(0), y(0) {}
    CFX_PTemplate(BaseType new_x, BaseType new_y) : x(new_x), y(new_y) {}
    CFX_PTemplate(const CFX_PTemplate& other) : x(other.x), y(other.y) {}

    CFX_PTemplate& operator=(const CFX_PTemplate& other) {
        if (this != &other) {
            x = other.x;
            y = other.y;
        }
        return *this;
    }
    bool operator==(const CFX_PTemplate& other) const {
        return x == other.x && y == other.y;
    }
    bool operator!=(const CFX_PTemplate& other) const {
        return !(*this == other);
    }
    CFX_PTemplate& operator+=(const CFX_PTemplate<BaseType>& obj) {
        x += obj.x;
        y += obj.y;
        return *this;
    }
    CFX_PTemplate& operator-=(const CFX_PTemplate<BaseType>& obj) {
        x -= obj.x;
        y -= obj.y;
        return *this;
    }
    CFX_PTemplate operator+(const CFX_PTemplate& other) const {
        return CFX_PTemplate(x + other.x, y + other.y);
    }
    CFX_PTemplate operator-(const CFX_PTemplate& other) const {
        return CFX_PTemplate(x - other.x, y - other.y);
    }
};

BaseType x;
BaseType y;

using CFX_Point = CFX_PTemplate<int32_t>;
using CFX_PointF = CFX_PTemplate<float>;
```

```
template <class BaseType>
class CFX_STemplate {
```

```
public:
    CFX_STemplate() : width(0), height(0) {}

    CFX_STemplate(BaseType new_width, BaseType new_height)
        : width(new_width), height(new_height) {}

    CFX_STemplate(const CFX_STemplate& other)
        : width(other.width), height(other.height) {}

    template <typename OtherType>
    CFX_STemplate<OtherType> As() const {
        return CFX_STemplate<OtherType>(static_cast<OtherType>(width),
                                         static_cast<OtherType>(height));
    }

    void clear() {
        width = 0;
        height = 0;
    }

    CFX_STemplate& operator=(const CFX_STemplate& other) {
        if (this != &other) {
            width = other.width;
            height = other.height;
        }
        return *this;
    }

    bool operator==(const CFX_STemplate& other) const {
        return width == other.width && height == other.height;
    }

    bool operator!=(const CFX_STemplate& other) const {
        return !(*this == other);
    }

    CFX_STemplate& operator+=(const CFX_STemplate<BaseType>& obj) {
        width += obj.width;
        height += obj.height;
        return *this;
    }

    CFX_STemplate& operator-=(const CFX_STemplate<BaseType>& obj) {
        width -= obj.width;
        height -= obj.height;
        return *this;
    }

    CFX_STemplate& operator*=(BaseType factor) {
        width *= factor;
        height *= factor;
        return *this;
    }

    CFX_STemplate& operator/=(BaseType divisor) {
        width /= divisor;
        height /= divisor;
        return *this;
    }

    CFX_STemplate operator+(const CFX_STemplate& other) const {
        return CFX_STemplate(width + other.width, height + other.height);
    }

    CFX_STemplate operator-(const CFX_STemplate& other) const {
        return CFX_STemplate(width - other.width, height - other.height);
    }

    CFX_STemplate operator*(BaseType factor) const {
        return CFX_STemplate(width * factor, height * factor);
    }

    CFX_STemplate operator/(BaseType divisor) const {
        return CFX_STemplate(width / divisor, height / divisor);
    }
};
```

```

    }

    BaseType width;
    BaseType height;
};
using CFX_Size = CFX_STemplate<int32_t>;
using CFX_SizeF = CFX_STemplate<float>;

template <class BaseType>
class CFX_VTemplate final : public CFX_PTemplate<BaseType> {
public:
    using CFX_PTemplate<BaseType>::x;
    using CFX_PTemplate<BaseType>::y;

    CFX_VTemplate() : CFX_PTemplate<BaseType>() {}
    CFX_VTemplate(BaseType new_x, BaseType new_y)
        : CFX_PTemplate<BaseType>(new_x, new_y) {}

    CFX_VTemplate(const CFX_VTemplate& other) : CFX_PTemplate<BaseType>(other) {}

    CFX_VTemplate(const CFX_PTemplate<BaseType>& point1,
                 const CFX_PTemplate<BaseType>& point2)
        : CFX_PTemplate<BaseType>(point2.x - point1.x, point2.y - point1.y) {}

    float Length() const { return sqrt(x * x + y * y); }
    void Normalize() {
        float fLen = Length();
        if (fLen < 0.0001f)
            return;

        x /= fLen;
        y /= fLen;
    }
    void Translate(BaseType dx, BaseType dy) {
        x += dx;
        y += dy;
    }
    void Scale(BaseType sx, BaseType sy) {
        x *= sx;
        y *= sy;
    }
    void Rotate(float fRadian) {
        float cosValue = cos(fRadian);
        float sinValue = sin(fRadian);
        x = x * cosValue - y * sinValue;
        y = x * sinValue + y * cosValue;
    }
};
using CFX_Vector = CFX_VTemplate<int32_t>;
using CFX_VectorF = CFX_VTemplate<float>;

// Rectangles.
// TODO(tsepez): Consolidate all these different rectangle classes.

// LTRB rectangles (y-axis runs downwards).
// Struct layout is compatible with win32 RECT.
struct FX_RECT {
    FX_RECT() = default;
    FX_RECT(int l, int t, int r, int b) : left(l), top(t), right(r), bottom(b) {}

    int Width() const { return right - left; }
    int Height() const { return bottom - top; }
    bool IsEmpty() const { return right <= left || bottom <= top; }
};

```

```

bool Valid() const {
    pdfium::base::CheckedNumeric<int> w = right;
    pdfium::base::CheckedNumeric<int> h = bottom;
    w -= left;
    h -= top;
    return w.IsValid() && h.IsValid();
}

void Normalize();
void Intersect(const FX_RECT& src);
void Intersect(int l, int t, int r, int b) { Intersect(FX_RECT(l, t, r, b)); }

void Offset(int dx, int dy) {
    left += dx;
    right += dx;
    top += dy;
    bottom += dy;
}

bool operator==(const FX_RECT& src) const {
    return left == src.left && right == src.right && top == src.top &&
           bottom == src.bottom;
}

bool Contains(int x, int y) const {
    return x >= left && x < right && y >= top && y < bottom;
}

int32_t left = 0;
int32_t top = 0;
int32_t right = 0;
int32_t bottom = 0;
};

// LTRB rectangles (y-axis runs upwards).
class CFX_FloatRect {
public:
    constexpr CFX_FloatRect() = default;
    constexpr CFX_FloatRect(float l, float b, float r, float t)
        : left(l), bottom(b), right(r), top(t) {}

    explicit CFX_FloatRect(const float* pArray)
        : CFX_FloatRect(pArray[0], pArray[1], pArray[2], pArray[3]) {}

    explicit CFX_FloatRect(const FX_RECT& rect);

    static CFX_FloatRect GetBBox(const CFX_PointF* pPoints, int nPoints);

    void Normalize();

    bool IsEmpty() const { return left >= right || bottom >= top; }
    bool Contains(const CFX_PointF& point) const;
    bool Contains(const CFX_FloatRect& other_rect) const;

    void Intersect(const CFX_FloatRect& other_rect);
    void Union(const CFX_FloatRect& other_rect);

    // These may be better at rounding than ToFxRect() and friends.
    //
    // Returned rect has bounds rounded up/down such that it is contained in the
    // original.
    FX_RECT GetInnerRect() const;

```

```
// Returned rect has bounds rounded up/down such that the original is
// contained in it.
FX_RECT GetOuterRect() const;

// Returned rect has bounds rounded up/down such that the dimensions are
// rounded up and the sum of the error in the bounds is minimized.
FX_RECT GetClosestRect() const;

CFX_FloatRect GetCenterSquare() const;

void InitRect(const CFX_PointF& point) {
    left = point.x;
    right = point.x;
    bottom = point.y;
    top = point.y;
}
void UpdateRect(const CFX_PointF& point);

float Width() const { return right - left; }
float Height() const { return top - bottom; }
float Left() const { return left; }
float Bottom() const { return bottom; }
float Right() const { return right; }
float Top() const { return top; }

void Inflate(float x, float y);
void Inflate(float other_left,
             float other_bottom,
             float other_right,
             float other_top);
void Inflate(const CFX_FloatRect& rt);

void Deflate(float x, float y);
void Deflate(float other_left,
             float other_bottom,
             float other_right,
             float other_top);
void Deflate(const CFX_FloatRect& rt);

CFX_FloatRect GetDeflated(float x, float y) const;

void Translate(float e, float f);

void Scale(float fScale);
void ScaleFromCenterPoint(float fScale);

// GetInnerRect() and friends may be better at rounding than these methods.
// Unlike the methods above, these two blindly floor / round the LBRT values.
// Doing so may introduce rounding errors that are visible to users as
// off-by-one pixels/lines.
//
// Floors LBRT values.
FX_RECT ToFxRect() const;

// Rounds LBRT values.
FX_RECT ToRoundedFxRect() const;

float left = 0.0f;
float bottom = 0.0f;
float right = 0.0f;
float top = 0.0f;
};
```

```

#ifndef NDEBUG
std::ostream& operator<<(std::ostream& os, const CFX_FloatRect& rect);
#endif

// LTWH rectangles (y-axis runs downwards).
class CFX_RectF {
public:
    using PointType = CFX_PointF;
    using SizeType = CFX_SizeF;

    CFX_RectF() = default;
    CFX_RectF(float dst_left, float dst_top, float dst_width, float dst_height)
        : left(dst_left), top(dst_top), width(dst_width), height(dst_height) {}
    CFX_RectF(float dst_left, float dst_top, const SizeType& dst_size)
        : left(dst_left),
          top(dst_top),
          width(dst_size.width),
          height(dst_size.height) {}
    CFX_RectF(const PointType& p, float dst_width, float dst_height)
        : left(p.x), top(p.y), width(dst_width), height(dst_height) {}
    CFX_RectF(const PointType& p1, const SizeType& s2)
        : left(p1.x), top(p1.y), width(s2.width), height(s2.height) {}
    explicit CFX_RectF(const FX_RECT& that)
        : left(static_cast<float>(that.left)),
          top(static_cast<float>(that.top)),
          width(static_cast<float>(that.Width())),
          height(static_cast<float>(that.Height())) {}

    // NOLINTNEXTLINE(runtime/explicit)
    CFX_RectF(const CFX_RectF& other) = default;

    CFX_RectF& operator+=(const PointType& p) {
        left += p.x;
        top += p.y;
        return *this;
    }
    CFX_RectF& operator--=(const PointType& p) {
        left -= p.x;
        top -= p.y;
        return *this;
    }
    float right() const { return left + width; }
    float bottom() const { return top + height; }
    void Normalize() {
        if (width < 0) {
            left += width;
            width = -width;
        }
        if (height < 0) {
            top += height;
            height = -height;
        }
    }
    void Offset(float dx, float dy) {
        left += dx;
        top += dy;
    }
    void Inflate(float x, float y) {
        left -= x;
        width += x * 2;
        top -= y;
        height += y * 2;
    }

```

```
}  
void Inflate(const PointType& p) { Inflate(p.x, p.y); }  
void Inflate(float off_left,  
            float off_top,  
            float off_right,  
            float off_bottom) {  
    left -= off_left;  
    top -= off_top;  
    width += off_left + off_right;  
    height += off_top + off_bottom;  
}  
void Inflate(const CFX_RectF& rt) {  
    Inflate(rt.left, rt.top, rt.left + rt.width, rt.top + rt.height);  
}  
void Deflate(float x, float y) {  
    left += x;  
    width -= x * 2;  
    top += y;  
    height -= y * 2;  
}  
void Deflate(const PointType& p) { Deflate(p.x, p.y); }  
void Deflate(float off_left,  
            float off_top,  
            float off_right,  
            float off_bottom) {  
    left += off_left;  
    top += off_top;  
    width -= off_left + off_right;  
    height -= off_top + off_bottom;  
}  
void Deflate(const CFX_RectF& rt) {  
    Deflate(rt.left, rt.top, rt.top + rt.width, rt.top + rt.height);  
}  
bool IsEmpty() const { return width <= 0 || height <= 0; }  
bool IsEmpty(float fEpsilon) const {  
    return width <= fEpsilon || height <= fEpsilon;  
}  
void Empty() { width = height = 0; }  
bool Contains(const PointType& p) const {  
    return p.x >= left && p.x < left + width && p.y >= top &&  
        p.y < top + height;  
}  
bool Contains(const CFX_RectF& rt) const {  
    return rt.left >= left && rt.right() <= right() && rt.top >= top &&  
        rt.bottom() <= bottom();  
}  
float Left() const { return left; }  
float Top() const { return top; }  
float Width() const { return width; }  
float Height() const { return height; }  
SizeType Size() const { return SizeType(width, height); }  
PointType TopLeft() const { return PointType(left, top); }  
PointType TopRight() const { return PointType(left + width, top); }  
PointType BottomLeft() const { return PointType(left, top + height); }  
PointType BottomRight() const {  
    return PointType(left + width, top + height);  
}  
PointType Center() const {  
    return PointType(left + width / 2, top + height / 2);  
}  
void Union(float x, float y) {  
    float r = right();  
    float b = bottom();
```

```

    left = std::min(left, x);
    top = std::min(top, y);
    r = std::max(r, x);
    b = std::max(b, y);

    width = r - left;
    height = b - top;
}
void Union(const PointType& p) { Union(p.x, p.y); }
void Union(const CFX_RectF& rt) {
    float r = right();
    float b = bottom();

    left = std::min(left, rt.left);
    top = std::min(top, rt.top);
    r = std::max(r, rt.right());
    b = std::max(b, rt.bottom());

    width = r - left;
    height = b - top;
}
void Intersect(const CFX_RectF& rt) {
    float r = right();
    float b = bottom();

    left = std::max(left, rt.left);
    top = std::max(top, rt.top);
    r = std::min(r, rt.right());
    b = std::min(b, rt.bottom());

    width = r - left;
    height = b - top;
}
bool IntersectWith(const CFX_RectF& rt) const {
    CFX_RectF rect = rt;
    rect.Intersect(*this);
    return !rect.IsEmpty();
}
bool IntersectWith(const CFX_RectF& rt, float fEpsilon) const {
    CFX_RectF rect = rt;
    rect.Intersect(*this);
    return !rect.IsEmpty(fEpsilon);
}
friend bool operator==(const CFX_RectF& rc1, const CFX_RectF& rc2) {
    return rc1.left == rc2.left && rc1.top == rc2.top &&
        rc1.width == rc2.width && rc1.height == rc2.height;
}
friend bool operator!=(const CFX_RectF& rc1, const CFX_RectF& rc2) {
    return !(rc1 == rc2);
}

CFX_FloatRect ToFloatRect() const {
    // Note, we flip top/bottom here because the CFX_FloatRect has the
    // y-axis running in the opposite direction.
    return CFX_FloatRect(left, top, right(), bottom());
}

// Returned rect has bounds rounded up/down such that the original is
// contained in it.
FX_RECT GetOuterRect() const;

float left = 0.0f;

```

```

    float top = 0.0f;
    float width = 0.0f;
    float height = 0.0f;
};

#ifdef NDEBUG
std::ostream& operator<<(std::ostream& os, const CFX_RectF& rect);
#endif // NDEBUG

// The matrix is of the form:
// | a  b  0 |
// | c  d  0 |
// | e  f  1 |
// See PDF spec 1.7 Section 4.2.3.
//
class CFX_Matrix {
public:
    CFX_Matrix() = default;

    explicit CFX_Matrix(const float n[6])
        : a(n[0]), b(n[1]), c(n[2]), d(n[3]), e(n[4]), f(n[5]) {}

    CFX_Matrix(float a1, float b1, float c1, float d1, float e1, float f1)
        : a(a1), b(b1), c(c1), d(d1), e(e1), f(f1) {}

    CFX_Matrix(const CFX_Matrix& other) = default;

    CFX_Matrix& operator=(const CFX_Matrix& other) = default;

    bool operator==(const CFX_Matrix& other) const {
        return a == other.a && b == other.b && c == other.c && d == other.d &&
            e == other.e && f == other.f;
    }
    bool operator!=(const CFX_Matrix& other) const { return !(*this == other); }

    CFX_Matrix operator*(const CFX_Matrix& right) const {
        return CFX_Matrix(a * right.a + b * right.c, a * right.b + b * right.d,
            c * right.a + d * right.c, c * right.b + d * right.d,
            e * right.a + f * right.c + right.e,
            e * right.b + f * right.d + right.f);
    }
    CFX_Matrix& operator*=(const CFX_Matrix& other) {
        *this = *this * other;
        return *this;
    }

    bool IsIdentity() const { return *this == CFX_Matrix(); }
    CFX_Matrix GetInverse() const;

    bool Is90Rotated() const;
    bool IsScaled() const;
    bool WillScale() const { return a != 1.0f || b != 0 || c != 0 || d != 1.0f; }

    void Concat(const CFX_Matrix& right) { *this *= right; }
    void Translate(float x, float y);
    void TranslatePrepend(float x, float y);
    void Translate(int32_t x, int32_t y) {
        Translate(static_cast<float>(x), static_cast<float>(y));
    }
    void TranslatePrepend(int32_t x, int32_t y) {
        TranslatePrepend(static_cast<float>(x), static_cast<float>(y));
    }
}

```

```
void Scale(float sx, float sy);
void Rotate(float fRadian);
void Shear(float fAlphaRadian, float fBetaRadian);

void MatchRect(const CFX_FloatRect& dest, const CFX_FloatRect& src);

float GetXUnit() const;
float GetYUnit() const;
CFX_FloatRect GetUnitRect() const;

float TransformXDistance(float dx) const;
float TransformDistance(float distance) const;

CFX_PointF Transform(const CFX_PointF& point) const;

CFX_RectF TransformRect(const CFX_RectF& rect) const;
CFX_FloatRect TransformRect(const CFX_FloatRect& rect) const;

float a = 1.0f;
float b = 0.0f;
float c = 0.0f;
float d = 1.0f;
float e = 0.0f;
float f = 0.0f;
};

#endif // CORE_FXCRT_FX_COORDINATES_H
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_FX_EXTENSION_H_
#define CORE_FXCRT_FX_EXTENSION_H_

#include <time.h>

#include <cctype>
#include <cmath>
#include <cwctype>
#include <memory>

#include "core/fxcrt/fx_string.h"

#if defined(USE_SYSTEM_ICUUC)
#include <unicode/uchar.h>
#else
#include "third_party/icu/source/common/unicode/uchar.h"
#endif

#define FX_INVALID_OFFSET static_cast<uint32_t>(-1)

#ifdef PDF_ENABLE_XFA
#define FX_IsOdd(a) ((a)&1)
#endif // PDF_ENABLE_XFA

float FXSYS_wcstof(const wchar_t* pwsStr, int32_t iLength, int32_t* pUsedLen);
wchar_t* FXSYS_wcsncpy(wchar_t* dstStr, const wchar_t* srcStr, size_t count);
int32_t FXSYS_wcsnicmp(const wchar_t* s1, const wchar_t* s2, size_t count);

inline bool FXSYS_iswlower(int32_t c) {
    return u_islower(c);
}

inline bool FXSYS_iswupper(int32_t c) {
    return u_isupper(c);
}

inline int32_t FXSYS_towlower(wchar_t c) {
    return u_tolower(c);
}

inline int32_t FXSYS_towupper(wchar_t c) {
    return u_toupper(c);
}

inline char FXSYS_ToUpperASCII(char c) {
    return (c >= 'a' && c <= 'z') ? (c + ('A' - 'a')) : c;
}

inline bool FXSYS_iswalpha(wchar_t c) {
    return u_isalpha(c);
}

inline bool FXSYS_iswalnum(wchar_t c) {
    return u_isalnum(c);
}

inline bool FXSYS_isspace(wchar_t c) {
```

```
    return u_isspace(c);
}

inline bool FXSYS_IsOctalDigit(char c) {
    return c >= '0' && c <= '7';
}

inline bool FXSYS_IsHexDigit(char c) {
    return !((c & 0x80) || !std::isxdigit(c));
}

inline bool FXSYS_IsWideHexDigit(wchar_t c) {
    return !((c & 0xFFFFF80) || !std::isxdigit(c));
}

inline int FXSYS_HexCharToInt(char c) {
    if (!FXSYS_IsHexDigit(c))
        return 0;
    char upchar = FXSYS_ToUpperASCII(c);
    return upchar > '9' ? upchar - 'A' + 10 : upchar - '0';
}

inline int FXSYS_WideHexCharToInt(wchar_t c) {
    if (!FXSYS_IsWideHexDigit(c))
        return 0;
    char upchar = std::toupper(static_cast<char>(c));
    return upchar > '9' ? upchar - 'A' + 10 : upchar - '0';
}

inline bool FXSYS_IsDecimalDigit(char c) {
    return !((c & 0x80) || !std::isdigit(c));
}

inline bool FXSYS_IsDecimalDigit(wchar_t c) {
    return !((c & 0xFFFFF80) || !std::iswdigit(c));
}

inline int FXSYS_DecimalCharToInt(char c) {
    return FXSYS_IsDecimalDigit(c) ? c - '0' : 0;
}

inline int FXSYS_DecimalCharToInt(wchar_t c) {
    return FXSYS_IsDecimalDigit(c) ? c - L'0' : 0;
}

void FXSYS_IntToTwoHexChars(uint8_t n, char* buf);
void FXSYS_IntToFourHexChars(uint16_t n, char* buf);

size_t FXSYS_ToUTF16BE(uint32_t unicode, char* buf);

// Strict order over floating types where NaNs may be present.
template <typename T>
bool FXSYS_SafeEQ(const T& lhs, const T& rhs) {
    return (std::isnan(lhs) && std::isnan(rhs)) ||
        (!std::isnan(lhs) && !std::isnan(rhs) && lhs == rhs);
}

template <typename T>
bool FXSYS_SafeLT(const T& lhs, const T& rhs) {
    if (std::isnan(lhs) && std::isnan(rhs))
        return false;
    if (std::isnan(lhs) || std::isnan(rhs))
        return std::isnan(lhs) < std::isnan(rhs);
}
```

```
    return lhs < rhs;
}

// Override time/localtime functions for test consistency.
void FXSYS_SetTimeFunction(time_t (*func)());
void FXSYS_SetLocaltimeFunction(struct tm* (*func)(const time_t*));

// Replacements for time/localtime that respect overrides.
time_t FXSYS_time(time_t* tloc);
struct tm* FXSYS_localtime(const time_t* tp);

#endif // CORE_FXCRT_FX_EXTENSION_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_FX_MEMORY_H
#define CORE_FXCRT_FX_MEMORY_H

#include <stddef.h>

#ifdef __cplusplus
extern "C" {
#endif

// For external C libraries to malloc through PDFium. These may return nullptr.
void* FXMEM_DefaultAlloc(size_t byte_size);
void* FXMEM_DefaultCalloc(size_t num_elems, size_t byte_size);
void* FXMEM_DefaultRealloc(void* pointer, size_t new_size);
void FXMEM_DefaultFree(void* pointer);

#ifdef __cplusplus
} // extern "C"
#endif

#include "third_party/base/allocator/partition_allocator/partition_alloc.h"

pdfium::base::PartitionAllocatorGeneric& GetArrayBufferPartitionAllocator();
pdfium::base::PartitionAllocatorGeneric& GetGeneralPartitionAllocator();
pdfium::base::PartitionAllocatorGeneric& GetStringPartitionAllocator();

void FXMEM_InitializePartitionAlloc();
NOINLINE void FX_OutOfMemoryTerminate();

// These never return nullptr, and must return cleared memory.
#define FX_Alloc(type, size) \
    static_cast<type*>(FX_AllocOrDie(size, sizeof(type)))
#define FX_Alloc2D(type, w, h) \
    static_cast<type*>(FX_AllocOrDie2D(w, h, sizeof(type)))
#define FX_Realloc(type, ptr, size) \
    static_cast<type*>(FX_ReallocOrDie(ptr, size, sizeof(type)))

// May return nullptr, but returns cleared memory otherwise.
#define FX_TryAlloc(type, size) \
    static_cast<type*>(FX_SafeAlloc(size, sizeof(type)))
#define FX_TryRealloc(type, ptr, size) \
    static_cast<type*>(FX_SafeRealloc(ptr, size, sizeof(type)))

void* FX_SafeAlloc(size_t num_members, size_t member_size);
void* FX_SafeRealloc(void* ptr, size_t num_members, size_t member_size);
void* FX_AllocOrDie(size_t num_members, size_t member_size);
void* FX_AllocOrDie2D(size_t w, size_t h, size_t member_size);
void* FX_ReallocOrDie(void* ptr, size_t num_members, size_t member_size);
void FX_Free(void* ptr);

// The FX_ArraySize(arr) macro returns the # of elements in an array arr.
// The expression is a compile-time constant, and therefore can be
// used in defining new arrays, for example. If you use FX_ArraySize on
// a pointer by mistake, you will get a compile-time error.
//
// One caveat is that FX_ArraySize() doesn't accept any array of an
// anonymous type or a type defined inside a function.
#define FX_ArraySize(array) (sizeof(ArraySizeHelper(array)))
```

```
// This template function declaration is used in defining FX_ArraySize.
// Note that the function doesn't need an implementation, as we only
// use its type.
template <typename T, size_t N>
char (&ArraySizeHelper(T (&array)[N]))[N];

// Round up to the power-of-two boundary N.
template <int N, typename T>
inline T FxAlignToBoundary(T size) {
    static_assert(N > 0 && (N & (N - 1)) == 0, "Not non-zero power of two");
    return (size + (N - 1)) & ~(N - 1);
}

#endif // __cplusplus

#endif // CORE_FXCRT_FX_MEMORY_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FXCRT_FX_MEMORY_WRAPPERS_H_  
#define CORE_FXCRT_FX_MEMORY_WRAPPERS_H_
```

```
#include <limits>  
#include <type_traits>  
#include <utility>
```

```
#include "core/fxcrt/fx_memory.h"
```

```
// Used with std::unique_ptr to FX_Free raw memory.
```

```
struct FxFreeDeleter {  
    inline void operator()(void* ptr) const { FX_Free(ptr); }  
};
```

```
// Used with std::vector<> to put purely numeric vectors into  
// the same "general" partition used by FX_Alloc(). Otherwise,  
// replacing FX_Alloc/FX_Free pairs with std::vector<> may undo  
// some of the nice segregation that we get from partition alloc.
```

```
template <class T>  
struct FxAllocAllocator {  
public:  
    static_assert(std::is_arithmetic<T>::value,  
        "Only numeric types allowed in this partition");
```

```
    using value_type = T;  
    using pointer = T*;  
    using const_pointer = const T*;  
    using reference = T&;  
    using const_reference = const T&;  
    using size_type = size_t;  
    using difference_type = ptrdiff_t;
```

```
template <class U>  
struct rebind {  
    using other = FxAllocAllocator<U>;  
};
```

```
FxAllocAllocator() noexcept = default;  
FxAllocAllocator(const FxAllocAllocator& other) noexcept = default;  
~FxAllocAllocator() = default;
```

```
template <typename U>  
FxAllocAllocator(const FxAllocAllocator<U>& other) noexcept {}
```

```
pointer address(reference x) const noexcept { return &x; }  
const_pointer address(const_reference x) const noexcept { return &x; }  
pointer allocate(size_type n, const void* hint = 0) {  
    return static_cast<pointer>(FX_AllocOrDie(n, sizeof(value_type)));  
}  
void deallocate(pointer p, size_type n) { FX_Free(p); }  
size_type max_size() const noexcept {  
    return std::numeric_limits<size_type>::max() / sizeof(value_type);  
}
```

```
template <class U, class... Args>  
void construct(U* p, Args&&... args) {  
    new (reinterpret_cast<void*>(p)) U(std::forward<Args>(args)...);  
}
```

```
template <class U>
void destroy(U* p) {
    p->~U();
}

// There's no state, so they are all the same,
bool operator==(const FxAllocAllocator& that) { return true; }
bool operator!=(const FxAllocAllocator& that) { return false; }
};

#endif // CORE_FXCRT_FX_MEMORY_WRAPPERS_H_
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_FX_NUMBER_H
#define CORE_FXCRT_FX_NUMBER_H

#include <stdint.h>

#include "core/fxcrt/bytestring.h"

class FX_Number {
public:
    FX_Number();
    explicit FX_Number(uint32_t value) = delete;
    explicit FX_Number(int32_t value);
    explicit FX_Number(float value);
    explicit FX_Number(ByteStringView str);

    bool IsInteger() const { return m_bInteger; }
    bool IsSigned() const { return m_bSigned; }

    int32_t GetSigned() const; // Underflow/Overflow possible.
    float GetFloat() const;

private:
    bool m_bInteger; // One of the two integers vs. float type.
    bool m_bSigned; // Only valid if |m_bInteger|.
    union {
        uint32_t m_UnsignedValue;
        int32_t m_SignedValue;
        float m_FloatValue;
    };
};

#endif // CORE_FXCRT_FX_NUMBER_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_FX_RANDOM_H_
#define CORE_FXCRT_FX_RANDOM_H_

#include <stdint.h>

void* FX_Random_MT_Start(uint32_t dwSeed);
void FX_Random_MT_Close(void* pContext);
uint32_t FX_Random_MT_Generate(void* pContext);

void FX_Random_GenerateMT(uint32_t* pBuffer, int32_t iCount);

#endif // CORE_FXCRT_FX_RANDOM_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FXCRT_FX_SAFE_TYPES_H_  
#define CORE_FXCRT_FX_SAFE_TYPES_H_
```

```
#include <stdlib.h> // For size_t.
```

```
#include "core/fxcrt/fx_system.h"
```

```
#include "third_party/base/numerics/safe_math.h"
```

```
typedef pdfium::base::CheckedNumeric<uint32_t> FX_SAFE_UINT32;
```

```
typedef pdfium::base::CheckedNumeric<int32_t> FX_SAFE_INT32;
```

```
typedef pdfium::base::CheckedNumeric<int64_t> FX_SAFE_INT64;
```

```
typedef pdfium::base::CheckedNumeric<size_t> FX_SAFE_SIZE_T;
```

```
typedef pdfium::base::CheckedNumeric<FX_FILESIZE> FX_SAFE_FILESIZE;
```

```
#endif // CORE_FXCRT_FX_SAFE_TYPES_H_
```



```
        size_t size) = 0;
};

class IFX_SeekableReadStream : virtual public Retainable,
                               virtual public IFX_StreamWithSize {
public:
    static RetainPtr<IFX_SeekableReadStream> CreateFromFilename(
        const char* filename);

    virtual bool IsEOF();
    virtual FX_FILESIZE GetPosition();
    virtual size_t ReadBlock(void* buffer, size_t size);

    virtual bool ReadBlockAtOffset(void* buffer,
                                   FX_FILESIZE offset,
                                   size_t size) WARN_UNUSED_RESULT = 0;
};

class IFX_SeekableStream : public IFX_SeekableReadStream,
                           public IFX_SeekableWriteStream {
public:
    static RetainPtr<IFX_SeekableStream> CreateFromFilename(const char* filename,
                                                            uint32_t dwModes);

    static RetainPtr<IFX_SeekableStream> CreateFromFilename(
        const wchar_t* filename,
        uint32_t dwModes);

    // IFX_SeekableWriteStream:
    bool WriteBlock(const void* buffer, size_t size) override;
    bool WriteString(ByteStringView str) override;
};

#endif // CORE_FXCRT_FX_STREAM_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_FX_STRING_H_
#define CORE_FXCRT_FX_STRING_H_

#include <stdint.h>

#include <vector>

#include "core/fxcrt/bytestring.h"
#include "core/fxcrt/widestring.h"

#define FXBSTR_ID(c1, c2, c3, c4) \
  (((uint32_t)c1 << 24) | ((uint32_t)c2 << 16) | ((uint32_t)c3 << 8) | \
  ((uint32_t)c4))

ByteString FX_UTF8Encode(WideStringView wsStr);
WideString FX_UTF8Decode(ByteStringView bsStr);

float StringToFloat(ByteStringView str);
float StringToFloat(WideStringView wsStr);
size_t FloatToString(float f, char* buf);

double StringToDouble(ByteStringView str);
double StringToDouble(WideStringView wsStr);
size_t DoubleToString(double d, char* buf);

namespace fxcrt {

template <typename StrType>
std::vector<StrType> Split(const StrType& that, typename StrType::CharType ch) {
  std::vector<StrType> result;
  StringViewTemplate<typename StrType::CharType> remaining(that.span());
  while (1) {
    Optional<size_t> index = remaining.Find(ch);
    if (!index.has_value())
      break;
    result.emplace_back(remaining.Left(index.value()));
    remaining = remaining.Right(remaining.GetLength() - index.value() - 1);
  }
  result.emplace_back(remaining);
  return result;
}

} // namespace fxcrt

#endif // CORE_FXCRT_FX_STRING_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_FX_SYSTEM_H
#define CORE_FXCRT_FX_SYSTEM_H

#include <assert.h>
#include <math.h>
#include <stdarg.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wchar.h>

// _FX_PLATFORM_ values;
#define _FX_PLATFORM_WINDOWS_ 1
#define _FX_PLATFORM_LINUX_ 2
#define _FX_PLATFORM_APPLE_ 3

#if defined(_WIN32)
#define _FX_PLATFORM_ _FX_PLATFORM_WINDOWS_
#elif defined(_WIN64)
#define _FX_PLATFORM_ _FX_PLATFORM_WINDOWS_
#elif defined(__linux__)
#define _FX_PLATFORM_ _FX_PLATFORM_LINUX_
#elif defined(__APPLE__)
#define _FX_PLATFORM_ _FX_PLATFORM_APPLE_
#elif defined(__asmjs__) || defined(__wasm__)
#define _FX_PLATFORM_ _FX_PLATFORM_LINUX_
#endif

#if defined(_MSC_VER) && _MSC_VER < 1900
#error Sorry, VC++ 2015 or later is required to compile PDFium.
#endif // defined(_MSC_VER) && _MSC_VER < 1900

#if defined(__wasm__) && defined(PDF_ENABLE_V8)
#error Cannot compile v8 with wasm.
#endif // PDF_ENABLE_V8

#if _FX_PLATFORM_ == _FX_PLATFORM_WINDOWS_
#include <windows.h>
#include <sal.h>
#endif // _FX_PLATFORM_ == _FX_PLATFORM_WINDOWS_

#if _FX_PLATFORM_ == _FX_PLATFORM_APPLE_
#include <Carbon/Carbon.h>
#include <libkern/OSAtomic.h>
#endif // _FX_PLATFORM_ == _FX_PLATFORM_APPLE_

#ifdef __cplusplus
extern "C" {
#endif // __cplusplus

#define IsFloatZero(f) ((f) < 0.0001 && (f) > -0.0001)
#define IsFloatBigger(fa, fb) ((fa) > (fb) && !IsFloatZero((fa) - (fb)))
#define IsFloatSmaller(fa, fb) ((fa) < (fb) && !IsFloatZero((fa) - (fb)))
#define IsFloatEqual(fa, fb) IsFloatZero((fa) - (fb))
```

```
// PDFium file sizes match the platform, but PDFium itself does not support
// files larger than 2GB even if the platform does. The value must be signed
// to support -1 error returns.
// TODO(tsepez): support larger files.
#if _FX_PLATFORM_ == _FX_PLATFORM_WINDOWS_
#define FX_FILESIZE int32_t
#else // _FX_PLATFORM_ == _FX_PLATFORM_WINDOWS_
#define FX_FILESIZE off_t
#endif // _FX_PLATFORM_ == _FX_PLATFORM_WINDOWS_

#ifndef ASSERT
#ifndef NDEBUG
#define ASSERT assert
#else
#define ASSERT(a)
#endif // NDEBUG
#endif // ASSERT

// M_PI not universally present on all platforms.
#define FX_PI 3.1415926535897932384626433832795f
#define FX_BEZIER 0.5522847498308f

// NOTE: prevent use of the return value from sprintf() since some platforms
// have different return values.
#define FXSYS_sprintf (void)sprintf
#define FXSYS_vsprintf (void)vsprintf
#define FXSYS_sprintf DO_NOT_USE_SPRINTF_DIE_DIE_DIE
#define FXSYS_vsprintf DO_NOT_USE_VSPRINTF_DIE_DIE_DIE

#ifdef __cplusplus
} // extern "C"

#include "third_party/base/numerics/safe_conversions.h"

// Overloaded functions for C++ templates
inline size_t FXSYS_len(const char* ptr) {
    return strlen(ptr);
}

inline size_t FXSYS_len(const wchar_t* ptr) {
    return wcslen(ptr);
}

inline int FXSYS_cmp(const char* ptr1, const char* ptr2, size_t len) {
    return memcmp(ptr1, ptr2, len);
}

inline int FXSYS_cmp(const wchar_t* ptr1, const wchar_t* ptr2, size_t len) {
    return wmemcmp(ptr1, ptr2, len);
}

inline const char* FXSYS_chr(const char* ptr, char ch, size_t len) {
    return reinterpret_cast<const char*>(memchr(ptr, ch, len));
}

inline const wchar_t* FXSYS_chr(const wchar_t* ptr, wchar_t ch, size_t len) {
    return wmemchr(ptr, ch, len);
}

extern "C" {
#endif // __cplusplus

#if _FX_PLATFORM_ == _FX_PLATFORM_WINDOWS_
```

```
#define FXSYS_GetACP GetACP
#define FXSYS_itoa _itoa
#define FXSYS_WideCharToMultiByte WideCharToMultiByte
#define FXSYS_MultiByteToWideChar MultiByteToWideChar
#define FXSYS_strlwr _strlwr
#define FXSYS_strupr _strupr
#define FXSYS_stricmp _stricmp
#define FXSYS_wcsicmp _wcsicmp
#define FXSYS_wcslwr _wcslwr
#define FXSYS_wcsupr _wcsupr
#define FXSYS_pow(a, b) (float)powf(a, b)
size_t FXSYS_wcsftime(wchar_t* strDest,
                    size_t maxsize,
                    const wchar_t* format,
                    const struct tm* timeptr);

#define FXSYS_SetLastError SetLastError
#define FXSYS_GetLastError GetLastError
#else // _FX_PLATFORM_ == _FX_PLATFORM_WINDOWS_
int FXSYS_GetACP();
char* FXSYS_itoa(int value, char* str, int radix);
int FXSYS_WideCharToMultiByte(uint32_t codepage,
                              uint32_t dwFlags,
                              const wchar_t* wstr,
                              int wlen,
                              char* buf,
                              int buflen,
                              const char* default_str,
                              int* pUseDefault);
int FXSYS_MultiByteToWideChar(uint32_t codepage,
                              uint32_t dwFlags,
                              const char* bstr,
                              int blen,
                              wchar_t* buf,
                              int buflen);

char* FXSYS_strlwr(char* str);
char* FXSYS_strupr(char* str);
int FXSYS_stricmp(const char* str1, const char* str2);
int FXSYS_wcsicmp(const wchar_t* str1, const wchar_t* str2);
wchar_t* FXSYS_wcslwr(wchar_t* str);
wchar_t* FXSYS_wcsupr(wchar_t* str);
#define FXSYS_pow(a, b) (float)powf(a, b)
#define FXSYS_wcsftime wcsftime
void FXSYS_SetLastError(uint32_t err);
uint32_t FXSYS_GetLastError();
#endif // _FX_PLATFORM_ == _FX_PLATFORM_WINDOWS_

#define FXWORD_GET_LSBFIRST(p) \
    (static_cast<uint16_t>((static_cast<uint16_t>(p[1]) << 8) | \
                          (static_cast<uint16_t>(p[0]))))
#define FXWORD_GET_MSBFIRST(p) \
    (static_cast<uint16_t>((static_cast<uint16_t>(p[0]) << 8) | \
                          (static_cast<uint16_t>(p[1]))))
#define FXDWORD_GET_LSBFIRST(p) \
    ((static_cast<uint32_t>(p[3]) << 24) | (static_cast<uint32_t>(p[2]) << 16) | \
     (static_cast<uint32_t>(p[1]) << 8) | (static_cast<uint32_t>(p[0])))
#define FXDWORD_GET_MSBFIRST(p) \
    ((static_cast<uint32_t>(p[0]) << 24) | (static_cast<uint32_t>(p[1]) << 16) | \
     (static_cast<uint32_t>(p[2]) << 8) | (static_cast<uint32_t>(p[3])))
int32_t FXSYS_atoi(const char* str);
uint32_t FXSYS_atoui(const char* str);
int32_t FXSYS_wtoi(const wchar_t* str);
int64_t FXSYS_atoi64(const char* str);
const char* FXSYS_i64toa(int64_t value, char* str, int radix);
```

```
int FXSYS_roundf(float f);
int FXSYS_round(double d);
#define FXSYS_sqrt2(a, b) (float)sqrt((a) * (a) + (b) * (b))
#ifdef __cplusplus
} // extern C
#endif // __cplusplus

// To print a size_t value in a portable way:
// size_t size;
// printf("xyz: %" PRIuS, size);
// The "u" in the macro corresponds to %u, and S is for "size".
#if _FX_PLATFORM_ != _FX_PLATFORM_WINDOWS_

#if (defined(_INTTYPES_H) || defined(_INTTYPES_H_)) && !defined(PRIu64)
#error "inttypes.h has already been included before this header file, but "
#error "without __STDC_FORMAT_MACROS defined."
#endif

#if !defined(__STDC_FORMAT_MACROS)
#define __STDC_FORMAT_MACROS
#endif

#include <inttypes.h>

#if !defined(PRIuS)
#define PRIuS "zu"
#endif

#else // _FX_PLATFORM_ != _FX_PLATFORM_WINDOWS_

#if !defined(PRIuS)
#define PRIuS "Iu"
#endif

#endif // _FX_PLATFORM_ != _FX_PLATFORM_WINDOWS_

#endif // CORE_FXCRT_FX_SYSTEM_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FXCRT_FX_UNICODE_H_
#define CORE_FXCRT_FX_UNICODE_H_
```

```
#include "core/fxcrt/fx_system.h"
```

```
// NOTE: Order matters, less-than/greater-than comparisons are used.
```

```
enum class FX_BIDICLASS : uint8_t {
    kON = 0,      // Other Neutral
    kL = 1,      // Left Letter
    kR = 2,      // Right Letter
    kAN = 3,     // Arabic Number
    kEN = 4,     // European Number
    kAL = 5,     // Arabic Letter
    kNSM = 6,    // Non-spacing Mark
    kCS = 7,     // Common Number Separator
    kES = 8,     // European Separator
    kET = 9,     // European Number Terminator
    kBN = 10,    // Boundary Neutral
    kS = 11,    // Segment Separator
    kWS = 12,    // Whitespace
    kB = 13,     // Paragraph Separator
    kRLO = 14,  // Right-to-Left Override
    kRLE = 15,  // Right-to-Left Embedding
    kLRO = 16,  // Left-to-Right Override
    kLRE = 17,  // Left-to-Right Embedding
    kPDF = 18,  // Pop Directional Format
    kN = kON,
};
```

```
wchar_t FX_GetMirrorChar(wchar_t wch);
FX_BIDICLASS FX_GetBidiClass(wchar_t wch);
```

```
#ifdef PDF_ENABLE_XFA
```

```
// As defined in http://www.unicode.org/reports/tr14
```

```
enum class FX_BREAKPROPERTY : uint8_t {
    kOP = 0,
    kCL = 1,
    kQU = 2,
    kGL = 3,
    kNS = 4,
    kEX = 5,
    kSY = 6,
    kIS = 7,
    kPR = 8,
    kPO = 9,
    kNU = 10,
    kAL = 11,
    kID = 12,
    kIN = 13,
    kHY = 14,
    kBA = 15,
    kBB = 16,
    kB2 = 17,
    kZW = 18,
    kCM = 19,
    kWJ = 20,
    kH2 = 21,
```

```
kH3 = 22,  
kJL = 23,  
kJV = 24,  
kJT = 25,  
kBK = 26,  
kCR = 27,  
kLF = 28,  
kNL = 29,  
kSA = 30,  
kSG = 31,  
kCB = 32,  
kXX = 33,  
kAI = 34,  
kSP = 35,  
kNONE = 36,  
kTB = 37,  
};  
  
enum class FX_CHARTYPE : uint8_t {  
    kUnknown = 0,  
    kTab,  
    kSpace,  
    kControl,  
    kCombination,  
    kNumeric,  
    kNormal,  
    kArabicAlef,  
    kArabicSpecial,  
    kArabicDistortion,  
    kArabicNormal,  
    kArabicForm,  
    kArabic,  
};  
  
FX_CHARTYPE FX_GetCharType(wchar_t wch);  
  
// Analagous to ULineBreak in icu's uchar.h, but permuted order, and a  
// subset lacking some more recent additions.  
FX_BREAKPROPERTY FX_GetBreakProperty(wchar_t wch);  
#endif // PDF_ENABLE_XFA  
  
#endif // CORE_FXCRT_FX_UNICODE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

#ifdef CORE_FXCRT_MAYBE_OWNED_H
#define CORE_FXCRT_MAYBE_OWNED_H

#include <memory>
#include <utility>

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"

namespace fxcrt {

// A template that can hold either owned or unowned references, and cleans up
// appropriately. Possibly the most pernicious anti-pattern imaginable, but
// it crops up throughout the codebase due to a desire to avoid copying-in
// objects or data.
template <typename T, typename D = std::default_delete<T>>
class MaybeOwned {
public:
    MaybeOwned() = default;
    explicit MaybeOwned(T* ptr) : m_pObj(ptr) {}
    explicit MaybeOwned(const UnownedPtr<T>& ptr) : m_pObj(ptr.Get()) {}
    explicit MaybeOwned(std::unique_ptr<T, D> ptr)
        : m_pOwnedObj(std::move(ptr)), m_pObj(m_pOwnedObj.get()) {}

    MaybeOwned(const MaybeOwned& that) = delete;
    MaybeOwned(MaybeOwned&& that) noexcept
        : m_pOwnedObj(that.m_pOwnedObj.release()), m_pObj(that.m_pObj) {
        that.m_pObj = nullptr;
    }

    void Reset(std::unique_ptr<T, D> ptr) {
        m_pObj = ptr.get();
        m_pOwnedObj = std::move(ptr);
    }
    void Reset(T* ptr = nullptr) {
        m_pObj = ptr;
        m_pOwnedObj.reset();
    }
    // Helpful for untangling a collection of intertwined MaybeOwned<>.
    void ResetIfUnowned() {
        if (!IsOwned())
            Reset();
    }

    T* Get() const { return m_pObj.Get(); }
    bool IsOwned() const { return !!m_pOwnedObj; }

    // Downgrades to unowned, caller takes ownership.
    std::unique_ptr<T, D> Release() {
        ASSERT(IsOwned());
        return std::move(m_pOwnedObj);
    }

    // Downgrades to empty, caller takes ownership.
    std::unique_ptr<T, D> ReleaseAndClear() {
        ASSERT(IsOwned());
        m_pObj = nullptr;
        return std::move(m_pOwnedObj);
    }
};
```

```
MaybeOwned& operator=(const MaybeOwned& that) = delete;
MaybeOwned& operator=(MaybeOwned&& that) {
    m_pObj = that.m_pObj;
    m_pOwnedObj = std::move(that.m_pOwnedObj);
    that.m_pObj = nullptr;
    return *this;
}
MaybeOwned& operator=(T* ptr) {
    Reset(ptr);
    return *this;
}
MaybeOwned& operator=(const UnownedPtr<T>& ptr) {
    Reset(ptr.Get());
    return *this;
}
MaybeOwned& operator=(std::unique_ptr<T, D> ptr) {
    Reset(std::move(ptr));
    return *this;
}

bool operator==(const MaybeOwned& that) const { return Get() == that.Get(); }
bool operator==(const std::unique_ptr<T, D>& ptr) const {
    return Get() == ptr.get();
}
bool operator==(T* ptr) const { return Get() == ptr; }

bool operator!=(const MaybeOwned& that) const { return !(*this == that); }
bool operator!=(const std::unique_ptr<T, D> ptr) const {
    return !(*this == ptr);
}
bool operator!=(T* ptr) const { return !(*this == ptr); }

explicit operator bool() const { return !!m_pObj; }
T& operator*() const { return *m_pObj; }
T* operator->() const { return m_pObj.Get(); }

private:
    std::unique_ptr<T, D> m_pOwnedObj;
    UnownedPtr<T> m_pObj;
};

} // namespace fxcrt

using fxcrt::MaybeOwned;

#endif // CORE_FXCRT_MAYBE_OWNED_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FXCRT_OBSERVED_PTR_H_  
#define CORE_FXCRT_OBSERVED_PTR_H_
```

```
#include <set>
```

```
#include "core/fxcrt/fx_system.h"  
#include "third_party/base/stl_util.h"
```

```
namespace fxcrt {
```

```
class Observable {
```

```
public:
```

```
    // General-purpose interface for more complicated cleanup.
```

```
    class ObserverIface {
```

```
    public:
```

```
        virtual ~ObserverIface() = default;
```

```
        virtual void OnObservableDestroyed() = 0;
```

```
};
```

```
Observable();
```

```
Observable(const Observable& that) = delete;
```

```
~Observable();
```

```
void AddObserver(ObserverIface* pObserver) {
```

```
    ASSERT(!pdfium::ContainsKey(m_Observers, pObserver));
```

```
    m_Observers.insert(pObserver);
```

```
}
```

```
void RemoveObserver(ObserverIface* pObserver) {
```

```
    ASSERT(pdfium::ContainsKey(m_Observers, pObserver));
```

```
    m_Observers.erase(pObserver);
```

```
}
```

```
void NotifyObservers() {
```

```
    for (auto* pObserver : m_Observers)
```

```
        pObserver->OnObservableDestroyed();
```

```
    m_Observers.clear();
```

```
}
```

```
Observable& operator=(const Observable& that) = delete;
```

```
protected:
```

```
    size_t ActiveObserversForTesting() const { return m_Observers.size(); }
```

```
private:
```

```
    std::set<ObserverIface*> m_Observers;
```

```
};
```

```
// Simple case of a self-nulling pointer.
```

```
template <typename T>
```

```
class ObservedPtr final : public Observable::ObserverIface {
```

```
public:
```

```
    ObservedPtr() = default;
```

```
    explicit ObservedPtr(T* pObservable) : m_pObservable(pObservable) {
```

```
        if (m_pObservable)
```

```
            m_pObservable->AddObserver(this);
```

```
    }
```

```
    ObservedPtr(const ObservedPtr& that) : ObservedPtr(that.Get()) {}
```

```
    ~ObservedPtr() override {
```

```
        if (m_pObservable)
```

```
            m_pObservable->RemoveObserver(this);
```

```
    }
```

```
    void Reset(T* pObservable = nullptr) {
```

```

    if (m_pObservable)
        m_pObservable->RemoveObserver(this);
    m_pObservable = pObservable;
    if (m_pObservable)
        m_pObservable->AddObserver(this);
}
void OnObservableDestroyed() override {
    ASSERT(m_pObservable);
    m_pObservable = nullptr;
}
bool HasObservable() const { return !!m_pObservable; }
ObservedPtr& operator=(const ObservedPtr& that) {
    Reset(that.Get());
    return *this;
}
bool operator==(const ObservedPtr& that) const {
    return m_pObservable == that.m_pObservable;
}
bool operator!=(const ObservedPtr& that) const { return !(*this == that); }

template <typename U>
bool operator==(const U* that) const {
    return Get() == that;
}

template <typename U>
bool operator!=(const U* that) const {
    return !(*this == that);
}

explicit operator bool() const { return HasObservable(); }
T* Get() const { return m_pObservable; }
T& operator*() const { return *m_pObservable; }
T* operator->() const { return m_pObservable; }

private:
    T* m_pObservable = nullptr;
};

template <typename T, typename U>
inline bool operator==(const U* lhs, const ObservedPtr<T>& rhs) {
    return rhs == lhs;
}

template <typename T, typename U>
inline bool operator!=(const U* lhs, const ObservedPtr<T>& rhs) {
    return rhs != lhs;
}

} // namespace fxcrt

using fxcrt::Observable;
using fxcrt::ObservedPtr;

#endif // CORE_FXCRT_OBSERVED_PTR_H_

```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_PAUSEINDICATOR_IFACE_H_
#define CORE_FXCRT_PAUSEINDICATOR_IFACE_H_

class PauseIndicatorIface {
public:
  virtual ~PauseIndicatorIface() = default;
  virtual bool NeedToPauseNow() = 0;
};

#endif // CORE_FXCRT_PAUSEINDICATOR_IFACE_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

#ifndef CORE_FXCRT_RETAINED_TREE_NODE_H_
#define CORE_FXCRT_RETAINED_TREE_NODE_H_

#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/tree_node.h"
#include "third_party/base/logging.h"

namespace fxcrt {

// For DOM/XML-ish trees, where references outside the tree are RetainPtr<T>,
// and the parent node also "retains" its children but doesn't always have
// a direct pointer to them.
template <typename T>
class RetainedTreeNode : public TreeNode<T> {
public:
    template <typename U, typename... Args>
    friend RetainPtr<U> pdfium::MakeRetain(Args&&... args);

    void AppendFirstChild(const RetainPtr<T>& child) {
        TreeNode<T>::AppendFirstChild(child.Get());
    }

    void AppendLastChild(const RetainPtr<T>& child) {
        TreeNode<T>::AppendLastChild(child.Get());
    }

    void InsertBefore(const RetainPtr<T>& child, T* other) {
        TreeNode<T>::InsertBefore(child.Get(), other);
    }

    void InsertAfter(const RetainPtr<T>& child, T* other) {
        TreeNode<T>::InsertAfter(child.Get(), other);
    }

    void RemoveChild(const RetainPtr<T>& child) {
        TreeNode<T>::RemoveChild(child.Get());
    }

    void RemoveSelfIfParented() {
        if (T* parent = TreeNode<T>::GetParent()) {
            parent->TreeNode<T>::RemoveChild(
                pdfium::WrapRetain(static_cast<T*>(this)).Get());
        }
    }

protected:
    RetainedTreeNode() = default;
    ~RetainedTreeNode() override {
        while (auto* pChild = TreeNode<T>::GetFirstChild())
            RemoveChild(pdfium::WrapRetain(pChild));
    }

private:
    template <typename U>
    friend struct ReleaseDeleter;

    template <typename U>
    friend class RetainPtr;
};
```

```
RetainedTreeNode(const RetainedTreeNode& that) = delete;
RetainedTreeNode& operator=(const RetainedTreeNode& that) = delete;

void Retain() { ++m_nRefCount; }
void Release() {
  ASSERT(m_nRefCount > 0);
  if (--m_nRefCount == 0 && !TreeNode<T>::GetParent())
    delete this;
}

intptr_t m_nRefCount = 0;
};

} // namespace fxcrt

using fxcrt::RetainedTreeNode;

#endif // CORE_FXCRT_RETAINED_TREE_NODE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FXCRT_RETAIN_PTR_H_  
#define CORE_FXCRT_RETAIN_PTR_H_
```

```
#include <functional>  
#include <memory>  
#include <utility>
```

```
#include "core/fxcrt/fx_system.h"  
#include "core/fxcrt/unowned_ptr.h"
```

```
namespace fxcrt {
```

```
// Used with std::unique_ptr to Release() objects that can't be deleted.
```

```
template <class T>  
struct ReleaseDeleter {  
    inline void operator()(T* ptr) const { ptr->Release(); }  
};
```

```
// Analogous to base's scoped_refptr.
```

```
template <class T>  
class RetainPtr {  
public:  
    explicit RetainPtr(T* pObj) : m_pObj(pObj) {  
        if (m_pObj)  
            m_pObj->Retain();  
    }  
};
```

```
RetainPtr() = default;
```

```
RetainPtr(const RetainPtr& that) : RetainPtr(that.Get()) {}
```

```
RetainPtr(RetainPtr&& that) noexcept { Swap(that); }
```

```
// Deliberately implicit to allow returning nullptrs.
```

```
// NOLINTNEXTLINE(runtime/explicit)
```

```
RetainPtr(std::nullptr_t ptr) {}
```

```
template <class U>
```

```
RetainPtr(const RetainPtr<U>& that) : RetainPtr(that.Get()) {}
```

```
template <class U>
```

```
RetainPtr<U> As() const {  
    return RetainPtr<U>(static_cast<U*>(Get()));  
}
```

```
void Reset(T* obj = nullptr) {
```

```
    if (obj)  
        obj->Retain();  
    m_pObj.reset(obj);  
}
```

```
T* Get() const { return m_pObj.get(); }
```

```
UnownedPtr<T> BackPointer() const { return UnownedPtr<T>(Get()); }
```

```
void Swap(RetainPtr& that) { m_pObj.swap(that.m_pObj); }
```

```
// Useful for passing notion of object ownership across a C API.
```

```
T* Leak() { return m_pObj.release(); }
```

```
void Unleak(T* ptr) { m_pObj.reset(ptr); }
```

```
RetainPtr& operator=(const RetainPtr& that) {  
    if (*this != that)
```

```

    Reset(that.Get());
    return *this;
}

RetainPtr& operator=(RetainPtr&& that) {
    m_pObj.reset(that.Leak());
    return *this;
}

// Assignment from raw pointers is intentionally not provided to make
// reference count churn more visible where possible.

bool operator==(const RetainPtr& that) const { return Get() == that.Get(); }
bool operator!=(const RetainPtr& that) const { return !(*this == that); }

template <typename U>
bool operator==(const U& that) const {
    return Get() == that;
}

template <typename U>
bool operator!=(const U& that) const {
    return !(*this == that);
}

bool operator<(const RetainPtr& that) const {
    return std::less<T*>()(Get(), that.Get());
}

explicit operator bool() const { return !!m_pObj; }
T& operator*() const { return *m_pObj; }
T* operator->() const { return m_pObj.get(); }

private:
    std::unique_ptr<T, ReleaseDeleter<T>> m_pObj;
};

// Trivial implementation - internal ref count with virtual destructor.
class Retainable {
public:
    Retainable() = default;

    bool HasOneRef() const { return m_nRefCount == 1; }

protected:
    virtual ~Retainable() = default;

private:
    template <typename U>
    friend struct ReleaseDeleter;

    template <typename U>
    friend class RetainPtr;

    Retainable(const Retainable& that) = delete;
    Retainable& operator=(const Retainable& that) = delete;

    void Retain() const { ++m_nRefCount; }
    void Release() const {
        ASSERT(m_nRefCount > 0);
        if (--m_nRefCount == 0)
            delete this;
    }
}

```

```
mutable intptr_t m_nRefCount = 0;
};

template <typename T, typename U>
inline bool operator==(const U* lhs, const RetainPtr<T>& rhs) {
    return rhs == lhs;
}

template <typename T, typename U>
inline bool operator!=(const U* lhs, const RetainPtr<T>& rhs) {
    return rhs != lhs;
}

} // namespace fxcrt

using fxcrt::ReleaseDeleter;
using fxcrt::Retainable;
using fxcrt::RetainPtr;

namespace pdfium {

// Helper to make a RetainPtr along the lines of std::make_unique<>(),
// or pdfium::MakeUnique<>(). Arguments are forwarded to T's constructor.
// Classes managed by RetainPtr should have protected (or private)
// constructors, and should friend this function.
template <typename T, typename... Args>
RetainPtr<T> MakeRetain(Args&&... args) {
    return RetainPtr<T>(new T(std::forward<Args>(args)...));
}

// Type-deducing wrapper to make a RetainPtr from an ordinary pointer.
template <typename T>
RetainPtr<T> WrapRetain(T* that) {
    return RetainPtr<T>(that);
}

} // namespace pdfium

#endif // CORE_FXCRT_RETAIN_PTR_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_SHARED_COPY_ON_WRITE_H_
#define CORE_FXCRT_SHARED_COPY_ON_WRITE_H_

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

namespace fxcrt {

// A shared pointer to an object with Copy on Write semantics that makes it
// appear as if all instances were independent. |ObjClass| must implement the
// requirements of |Retainable| from retain_ptr.h, and must also provide a
// Clone() method. Often this will just call MakeRetain<>(*this) but will need
// to be virtual if |ObjClass| is subclassed.
template <class ObjClass>
class SharedCopyOnWrite {
public:
  SharedCopyOnWrite() {}
  SharedCopyOnWrite(const SharedCopyOnWrite& other)
    : m_pObject(other.m_pObject) {}
  ~SharedCopyOnWrite() {}

  template <typename... Args>
  ObjClass* Emplace(Args... params) {
    m_pObject = pdfium::MakeRetain<ObjClass>(params...);
    return m_pObject.Get();
  }

  SharedCopyOnWrite& operator=(const SharedCopyOnWrite& that) {
    if (*this != that)
      m_pObject = that.m_pObject;
    return *this;
  }

  void SetNull() { m_pObject.Reset(); }
  const ObjClass* GetObject() const { return m_pObject.Get(); }

  template <typename... Args>
  ObjClass* GetPrivateCopy(Args... params) {
    if (!m_pObject)
      return Emplace(params...);
    if (!m_pObject->HasOneRef())
      m_pObject = m_pObject->Clone();
    return m_pObject.Get();
  }

  bool operator==(const SharedCopyOnWrite& that) const {
    return m_pObject == that.m_pObject;
  }
  bool operator!=(const SharedCopyOnWrite& that) const {
    return !(*this == that);
  }
  explicit operator bool() const { return !!m_pObject; }

private:
  RetainPtr<ObjClass> m_pObject;
};
```

```
} // namespace fxcrt
```

```
using fxcrt::SharedCopyOnWrite;
```

```
#endif // CORE_FXCRT_SHARED_COPY_ON_WRITE_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_STRING_DATA_TEMPLATE_H_
#define CORE_FXCRT_STRING_DATA_TEMPLATE_H_

#include "core/fxcrt/fx_memory.h"
#include "core/fxcrt/fx_system.h"
#include "third_party/base/numerics/safe_math.h"

namespace fxcrt {

template <typename CharType>
class StringDataTemplate {
public:
    static StringDataTemplate* Create(size_t nLen) {
        ASSERT(nLen > 0);

        // Calculate space needed for the fixed portion of the struct plus the
        // NUL char that is not included in |m_nAllocLength|.
        int overhead = offsetof(StringDataTemplate, m_String) + sizeof(CharType);
        pdfium::base::CheckedNumeric<size_t> nSize = nLen;
        nSize *= sizeof(CharType);
        nSize += overhead;

        // Now round to an 8-byte boundary. We'd expect that this is the minimum
        // granularity of any of the underlying allocators, so there may be cases
        // where we can save a re-alloc when adding a few characters to a string
        // by using this otherwise wasted space.
        nSize += 7;
        nSize &= ~7;
        size_t totalSize = nSize.ValueOrDie();
        size_t usableLen = (totalSize - overhead) / sizeof(CharType);
        ASSERT(usableLen >= nLen);

        void* pData = GetStringPartitionAllocator().root()->Alloc(
            totalSize, "StringDataTemplate");
        return new (pData) StringDataTemplate(nLen, usableLen);
    }

    static StringDataTemplate* Create(const CharType* pStr, size_t nLen) {
        StringDataTemplate* result = Create(nLen);
        result->CopyContents(pStr, nLen);
        return result;
    }

    void Retain() { ++m_nRefs; }
    void Release() {
        if (--m_nRefs <= 0)
            GetStringPartitionAllocator().root()->Free(this);
    }

    bool CanOperateInPlace(size_t nTotalLen) const {
        return m_nRefs <= 1 && nTotalLen <= m_nAllocLength;
    }

    void CopyContents(const StringDataTemplate& other) {
        ASSERT(other.m_nDataLength <= m_nAllocLength);
        memcpy(m_String, other.m_String,
            (other.m_nDataLength + 1) * sizeof(CharType));
    }
};

```

```
    }

    void CopyContents(const CharType* pStr, size_t nLen) {
        ASSERT(nLen >= 0);
        ASSERT(nLen <= m_nAllocLength);

        memcpy(m_String, pStr, nLen * sizeof(CharType));
        m_String[nLen] = 0;
    }

    void CopyContentsAt(size_t offset, const CharType* pStr, size_t nLen) {
        ASSERT(offset >= 0);
        ASSERT(nLen >= 0);
        ASSERT(offset + nLen <= m_nAllocLength);

        memcpy(m_String + offset, pStr, nLen * sizeof(CharType));
        m_String[offset + nLen] = 0;
    }

    // To ensure ref counts do not overflow, consider the worst possible case:
    // the entire address space contains nothing but pointers to this object.
    // Since the count increments with each new pointer, the largest value is
    // the number of pointers that can fit into the address space. The size of
    // the address space itself is a good upper bound on it.
    intptr_t m_nRefs;

    // These lengths are in terms of number of characters, not bytes, and do not
    // include the terminating NUL character, but the underlying buffer is sized
    // to be capable of holding it.
    size_t m_nDataLength;
    size_t m_nAllocLength;

    // Not really 1, variable size.
    CharType m_String[1];

private:
    StringDataTemplate(size_t dataLen, size_t allocLen)
        : m_nRefs(0), m_nDataLength(dataLen), m_nAllocLength(allocLen) {
        ASSERT(dataLen >= 0);
        ASSERT(dataLen <= allocLen);
        m_String[dataLen] = 0;
    }

    ~StringDataTemplate() = delete;
};

extern template class StringDataTemplate<char>;
extern template class StringDataTemplate<wchar_t>;

} // namespace fxcrt

using fxcrt::StringDataTemplate;

#endif // CORE_FXCRT_STRING_DATA_TEMPLATE_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_STRING_POOL_TEMPLATE_H_
#define CORE_FXCRT_STRING_POOL_TEMPLATE_H_

#include <unordered_set>

#include "core/fxcrt/fx_string.h"

namespace fxcrt {

template <typename StringType>
class StringPoolTemplate {
public:
  StringType Intern(const StringType& str) { return *m_Pool.insert(str).first; }
  void Clear() { m_Pool.clear(); }

private:
  std::unordered_set<StringType> m_Pool;
};

extern template class StringPoolTemplate<ByteString>;
extern template class StringPoolTemplate<WideString>;

} // namespace fxcrt

using fxcrt::StringPoolTemplate;

using ByteStringPool = StringPoolTemplate<ByteString>;
using WideStringPool = StringPoolTemplate<WideString>;

#endif // CORE_FXCRT_STRING_POOL_TEMPLATE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FXCRT_STRING_VIEW_TEMPLATE_H
```

```
#define CORE_FXCRT_STRING_VIEW_TEMPLATE_H
```

```
#include <algorithm>
```

```
#include <iterator>
```

```
#include <type_traits>
```

```
#include <vector>
```

```
#include "core/fxcrt/fx_system.h"
```

```
#include "third_party/base/optional.h"
```

```
#include "third_party/base/span.h"
```

```
#include "third_party/base/stl_util.h"
```

```
namespace fxcrt {
```

```
// An immutable string with caller-provided storage which must outlive the
```

```
// string itself. These are not necessarily nul-terminated, so that substring
```

```
// extraction (via the Mid(), Left(), and Right() methods) is copy-free.
```

```
//
```

```
// String view arguments should be passed by value, since they are small,
```

```
// rather than const-ref, even if they are not modified.
```

```
template <typename T>
```

```
class StringViewTemplate {
```

```
  public:
```

```
    using CharType = T;
```

```
    using UnsignedType = typename std::make_unsigned<CharType>::type;
```

```
    using const_iterator = const CharType*;
```

```
    using const_reverse_iterator = std::reverse_iterator<const_iterator>;
```

```
    constexpr StringViewTemplate() noexcept = default;
```

```
    constexpr StringViewTemplate(const StringViewTemplate& src) noexcept =  
      default;
```

```
// Deliberately implicit to avoid calling on every string literal.
```

```
// NOLINTNEXTLINE(runtime/explicit)
```

```
StringViewTemplate(const CharType* ptr) noexcept
```

```
  : m_Span(reinterpret_cast<const UnsignedType*>(ptr),  
            ptr ? FXSYS_len(ptr) : 0) {}
```

```
constexpr StringViewTemplate(const CharType* ptr, size_t len) noexcept
```

```
  : m_Span(reinterpret_cast<const UnsignedType*>(ptr), len) {}
```

```
explicit constexpr StringViewTemplate(  
  const pdfium::span<const CharType>& other) noexcept
```

```
  : m_Span(reinterpret_cast<const UnsignedType*>(other.data()),  
            other.size()) {}
```

```
template <typename U = UnsignedType>
```

```
constexpr StringViewTemplate(  
  const UnsignedType* ptr,
```

```
  size_t size,
```

```
  typename std::enable_if<!std::is_same<U, CharType>::value>::type* =  
  0) noexcept
```

```
  : m_Span(ptr, size) {}
```

```
template <typename U = UnsignedType>
```

```
StringViewTemplate(  
  const U* ptr,
```

```

    const pdfium::span<U> other,
    typename std::enable_if<!std::is_same<U, CharType>::value>::type* =
        0) noexcept
    : m_Span(other) {}

// Deliberately implicit to avoid calling on every string literal.
// |ch| must be an lvalue that outlives the StringViewTemplate.
// NOLINTNEXTLINE(runtime/explicit)
constexpr StringViewTemplate(CharType& ch) noexcept
    : m_Span(reinterpret_cast<const UnsignedType*>(&ch), 1) {}

// Any changes to |vec| invalidate the string.
template <typename AllocType>
explicit StringViewTemplate(
    const std::vector<UnsignedType, AllocType>& vec) noexcept
    : m_Span(!vec.empty() ? vec.data() : nullptr, vec.size()) {}

StringViewTemplate& operator=(const CharType* src) {
    m_Span = pdfium::span<const UnsignedType>(
        reinterpret_cast<const UnsignedType*>(src), src ? FXSYS_len(src) : 0);
    return *this;
}

StringViewTemplate& operator=(const StringViewTemplate& src) {
    m_Span = src.m_Span;
    return *this;
}

const_iterator begin() const {
    return reinterpret_cast<const_iterator>(m_Span.begin());
}

const_iterator end() const {
    return reinterpret_cast<const_iterator>(m_Span.end());
}

const_reverse_iterator rbegin() const {
    return const_reverse_iterator(end());
}

const_reverse_iterator rend() const {
    return const_reverse_iterator(begin());
}

bool operator==(const StringViewTemplate& other) const {
    return m_Span == other.m_Span;
}

bool operator==(const CharType* ptr) const {
    StringViewTemplate other(ptr);
    return *this == other;
}

bool operator!=(const CharType* ptr) const { return !(*this == ptr); }
bool operator!=(const StringViewTemplate& other) const {
    return !(*this == other);
}

bool IsASCII() const {
    for (auto c : *this) {
        if (c <= 0 || c > 127) // Questionable signedness of |c|.
            return false;
    }
    return true;
}

bool EqualsASCII(const StringViewTemplate<char>& that) const {
    size_t length = GetLength();

```

```
    if (length != that.GetLength())
        return false;

    for (size_t i = 0; i < length; ++i) {
        auto c = (*this)[i];
        if (c <= 0 || c > 127 || c != that[i]) // Questionable signedness of |c|.
            return false;
    }
    return true;
}

bool EqualsASCIINoCase(const StringViewTemplate<char>& that) const {
    size_t length = GetLength();
    if (length != that.GetLength())
        return false;

    for (size_t i = 0; i < length; ++i) {
        auto c = (*this)[i];
        if (c <= 0 || c > 127 || tolower(c) != tolower(that[i]))
            return false;
    }
    return true;
}

uint32_t GetID() const {
    if (m_Span.empty())
        return 0;

    uint32_t strid = 0;
    size_t size = std::min(static_cast<size_t>(4), m_Span.size());
    for (size_t i = 0; i < size; i++)
        strid = strid * 256 + m_Span[i];

    return strid << ((4 - size) * 8);
}

pdfium::span<const UnsignedType> raw_span() const { return m_Span; }
pdfium::span<const CharType> span() const {
    return pdfium::make_span(reinterpret_cast<const CharType*>(m_Span.data()),
                              m_Span.size());
}

const UnsignedType* raw_str() const { return m_Span.data(); }
const CharType* unterminated_c_str() const {
    return reinterpret_cast<const CharType*>(m_Span.data());
}

size_t GetLength() const { return m_Span.size(); }
bool IsEmpty() const { return m_Span.empty(); }
bool IsValidIndex(size_t index) const { return index < m_Span.size(); }
bool IsValidLength(size_t length) const { return length <= m_Span.size(); }

const UnsignedType& operator[](const size_t index) const {
    return m_Span[index];
}

UnsignedType First() const { return !m_Span.empty() ? m_Span[0] : 0; }
UnsignedType Last() const {
    return !m_Span.empty() ? m_Span[m_Span.size() - 1] : 0;
}

const CharType CharAt(const size_t index) const {
    return static_cast<CharType>(m_Span[index]);
}
```

```
Optional<size_t> Find(CharType ch) const {
    const auto* found = reinterpret_cast<const UnsignedType*>(FXSYS_chr(
        reinterpret_cast<const CharType*>(m_Span.data()), ch, m_Span.size()));

    return found ? Optional<size_t>(found - m_Span.data()) : Optional<size_t>();
}

bool Contains(CharType ch) const { return Find(ch).has_value(); }

StringViewTemplate Mid(size_t first, size_t count) const {
    if (!m_Span.data())
        return StringViewTemplate();

    if (!IsValidIndex(first))
        return StringViewTemplate();

    if (count == 0 || !IsValidLength(count))
        return StringViewTemplate();

    if (!IsValidIndex(first + count - 1))
        return StringViewTemplate();

    return StringViewTemplate(m_Span.data() + first, count);
}

StringViewTemplate Left(size_t count) const {
    if (count == 0 || !IsValidLength(count))
        return StringViewTemplate();
    return Mid(0, count);
}

StringViewTemplate Right(size_t count) const {
    if (count == 0 || !IsValidLength(count))
        return StringViewTemplate();
    return Mid(GetLength() - count, count);
}

StringViewTemplate TrimmedRight(CharType ch) const {
    if (IsEmpty())
        return StringViewTemplate();

    size_t pos = GetLength();
    while (pos && CharAt(pos - 1) == ch)
        pos--;

    if (pos == 0)
        return StringViewTemplate();

    return StringViewTemplate(m_Span.data(), pos);
}

bool operator<(const StringViewTemplate& that) const {
    int result =
        FXSYS_cmp(reinterpret_cast<const CharType*>(m_Span.data()),
            reinterpret_cast<const CharType*>(that.m_Span.data()),
            std::min(m_Span.size(), that.m_Span.size()));
    return result < 0 || (result == 0 && m_Span.size() < that.m_Span.size());
}

bool operator>(const StringViewTemplate& that) const {
    int result =
        FXSYS_cmp(reinterpret_cast<const CharType*>(m_Span.data()),
```

```
        reinterpret_cast<const CharType*>(that.m_Span.data()),
        std::min(m_Span.size(), that.m_Span.size()));
    return result > 0 || (result == 0 && m_Span.size() > that.m_Span.size());
}

protected:
    pdfium::span<const UnsignedType> m_Span;

private:
    void* operator new(size_t) throw() { return nullptr; }
};

template <typename T>
inline bool operator==(const T* lhs, const StringViewTemplate<T>& rhs) {
    return rhs == lhs;
}

template <typename T>
inline bool operator!=(const T* lhs, const StringViewTemplate<T>& rhs) {
    return rhs != lhs;
}

template <typename T>
inline bool operator<(const T* lhs, const StringViewTemplate<T>& rhs) {
    return rhs > lhs;
}

// Workaround for one of the cases external template classes are
// failing in GCC before version 7 with -O0
#if !defined(__GNUC__) || __GNUC__ >= 7
extern template class StringViewTemplate<char>;
extern template class StringViewTemplate<wchar_t>;
#endif

using ByteStringView = StringViewTemplate<char>;
using WideStringView = StringViewTemplate<wchar_t>;

} // namespace fxcrt

using ByteStringView = fxcrt::ByteStringView;
using WideStringView = fxcrt::WideStringView;

#endif // CORE_FXCRT_STRING_VIEW_TEMPLATE_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_TIMERHANDLER_IFACE_H_
#define CORE_FXCRT_TIMERHANDLER_IFACE_H_

#include "core/fxcrt/fx_system.h"

namespace fxcrt {

class TimerHandlerIface {
public:
    static constexpr int32_t kInvalidTimerID = 0;
    using TimerCallback = void (*)(int32_t idEvent);

    virtual ~TimerHandlerIface() = default;

    virtual int32_t SetTimer(int32_t uElapse, TimerCallback lpTimerFunc) = 0;
    virtual void KillTimer(int32_t nTimerID) = 0;
};

} // namespace fxcrt

using fxcrt::TimerHandlerIface;

#endif // CORE_FXCRT_TIMERHANDLER_IFACE_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FXCRT_TREE_NODE_H_  
#define CORE_FXCRT_TREE_NODE_H_
```

```
#include "core/fxcrt/fx_system.h"  
#include "third_party/base/logging.h"
```

```
namespace fxcrt {
```

```
// Implements the usual DOM/XML-ish trees.
```

```
template <typename T>
```

```
class TreeNode {
```

```
public:
```

```
TreeNode() = default;
```

```
virtual ~TreeNode() = default;
```

```
T* GetParent() const { return m_pParent; }
```

```
T* GetFirstChild() const { return m_pFirstChild; }
```

```
T* GetLastChild() const { return m_pLastChild; }
```

```
T* GetNextSibling() const { return m_pNextSibling; }
```

```
T* GetPrevSibling() const { return m_pPrevSibling; }
```

```
bool HasChild(const T* child) const {  
    return child != this && child->m_pParent == this;  
}
```

```
T* GetNthChild(int32_t n) {  
    if (n < 0)  
        return nullptr;  
    T* result = GetFirstChild();  
    while (n-- && result) {  
        result = result->GetNextSibling();  
    }  
    return result;  
}
```

```
void AppendFirstChild(T* child) {  
    BecomeParent(child);  
    if (m_pFirstChild) {  
        CHECK(m_pLastChild);  
        m_pFirstChild->m_pPrevSibling = child;  
        child->m_pNextSibling = m_pFirstChild;  
        m_pFirstChild = child;  
    } else {  
        CHECK(!m_pLastChild);  
        m_pFirstChild = child;  
        m_pLastChild = child;  
    }  
}
```

```
void AppendLastChild(T* child) {  
    BecomeParent(child);  
    if (m_pLastChild) {  
        CHECK(m_pFirstChild);  
        m_pLastChild->m_pNextSibling = child;  
        child->m_pPrevSibling = m_pLastChild;  
        m_pLastChild = child;  
    } else {  
        CHECK(!m_pFirstChild);  
        m_pFirstChild = child;
```

```
    m_pLastChild = child;
}
}

void InsertBefore(T* child, T* other) {
    if (!other) {
        AppendLastChild(child);
        return;
    }
    BecomeParent(child);
    CHECK(HasChild(other));
    child->m_pNextSibling = other;
    child->m_pPrevSibling = other->m_pPrevSibling;
    if (m_pFirstChild == other) {
        CHECK(!other->m_pPrevSibling);
        m_pFirstChild = child;
    } else {
        other->m_pPrevSibling->m_pNextSibling = child;
    }
    other->m_pPrevSibling = child;
}

void InsertAfter(T* child, T* other) {
    if (!other) {
        AppendFirstChild(child);
        return;
    }
    BecomeParent(child);
    CHECK(HasChild(other));
    child->m_pNextSibling = other->m_pNextSibling;
    child->m_pPrevSibling = other;
    if (m_pLastChild == other) {
        CHECK(!other->m_pNextSibling);
        m_pLastChild = child;
    } else {
        other->m_pNextSibling->m_pPrevSibling = child;
    }
    other->m_pNextSibling = child;
}

void RemoveChild(T* child) {
    CHECK(HasChild(child));
    if (m_pLastChild == child) {
        CHECK(!child->m_pNextSibling);
        m_pLastChild = child->m_pPrevSibling;
    } else {
        child->m_pNextSibling->m_pPrevSibling = child->m_pPrevSibling;
    }
    if (m_pFirstChild == child) {
        CHECK(!child->m_pPrevSibling);
        m_pFirstChild = child->m_pNextSibling;
    } else {
        child->m_pPrevSibling->m_pNextSibling = child->m_pNextSibling;
    }
    child->m_pParent = nullptr;
    child->m_pPrevSibling = nullptr;
    child->m_pNextSibling = nullptr;
}

void RemoveAllChildren() {
    while (T* child = GetFirstChild())
        RemoveChild(child);
}
```

```
void RemoveSelfIfParented() {
    if (T* parent = GetParent())
        parent->RemoveChild(static_cast<T*>(this));
}

private:
// Child left in state where sibling members need subsequent adjustment.
void BecomeParent(T* child) {
    CHECK(child != this); // Detect attempts at self-insertion.
    if (child->m_pParent)
        child->m_pParent->TreeNode<T>::RemoveChild(child);
    child->m_pParent = static_cast<T*>(this);
    CHECK(!child->m_pNextSibling);
    CHECK(!child->m_pPrevSibling);
}

T* m_pParent = nullptr; // Raw, intra-tree pointer.
T* m_pFirstChild = nullptr; // Raw, intra-tree pointer.
T* m_pLastChild = nullptr; // Raw, intra-tree pointer.
T* m_pNextSibling = nullptr; // Raw, intra-tree pointer
T* m_pPrevSibling = nullptr; // Raw, intra-tree pointer
};

} // namespace fxcrt

using fxcrt::TreeNode;

#endif // CORE_FXCRT_TREE_NODE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

#ifndef CORE_FXCRT_UNOWNED_PTR_H_
#define CORE_FXCRT_UNOWNED_PTR_H_

#include <functional>
#include <memory>
#include <type_traits>
#include <utility>

// UnownedPtr is a smart pointer class that behaves very much like a
// standard C-style pointer. The advantages of using it over raw
// pointers are:
//
// 1. It documents the nature of the pointer with no need to add a comment
//    explaining that is it // Not owned. Additionally, an attempt to delete
//    an unowned ptr will fail to compile rather than silently succeeding,
//    since it is a class and not a raw pointer.
//
// 2. When built using the memory tool ASAN, the class provides a destructor
//    which checks that the object being pointed to is still alive.
//
// Hence, when using UnownedPtr, no dangling pointers are ever permitted,
// even if they are not de-referenced after becoming dangling. The style of
// programming required is that the lifetime an object containing an
// UnownedPtr must be strictly less than the object to which it points.
//
// The same checks are also performed at assignment time to prove that the
// old value was not a dangling pointer, either.
//
// The array indexing operation [] is not supported on an unowned ptr,
// because an unowned ptr expresses a one to one relationship with some
// other heap object. Use pdfium::span<> for the cases where indexing
// into an unowned array is desired, which performs the same checks.

namespace pdfium {

template <typename T>
class span;

} // namespace pdfium

namespace fxcrt {

template <class T>
class UnownedPtr {
public:
    constexpr UnownedPtr() noexcept = default;
    constexpr UnownedPtr(const UnownedPtr& that) noexcept = default;
    constexpr UnownedPtr(UnownedPtr&& that) noexcept : m_pObj(that.Release()) {}

    template <typename U>
    explicit constexpr UnownedPtr(U* pObj) noexcept : m_pObj(pObj) {}

    // Deliberately implicit to allow returning nullptrs.
    // NOLINTNEXTLINE(runtime/explicit)
    constexpr UnownedPtr(std::nullptr_t ptr) noexcept {}

    ~UnownedPtr() { ProbeForLowSeverityLifetimeIssue(); }

    void Reset(T* obj = nullptr) {
```

```
    ProbeForLowSeverityLifetimeIssue();
    m_pObj = obj;
}

UnownedPtr& operator=(T* that) noexcept {
    Reset(that);
    return *this;
}

UnownedPtr& operator=(const UnownedPtr& that) noexcept {
    if (*this != that)
        Reset(that.Get());
    return *this;
}

UnownedPtr& operator=(UnownedPtr&& that) noexcept {
    if (*this != that)
        Reset(that.Release());
    return *this;
}

bool operator==(const UnownedPtr& that) const { return Get() == that.Get(); }
bool operator!=(const UnownedPtr& that) const { return !(*this == that); }
bool operator<(const UnownedPtr& that) const {
    return std::less<T*>()(Get(), that.Get());
}

template <typename U>
bool operator==(const U* that) const {
    return Get() == that;
}

template <typename U>
bool operator!=(const U* that) const {
    return !(*this == that);
}

T* Get() const noexcept { return m_pObj; }

T* Release() {
    ProbeForLowSeverityLifetimeIssue();
    T* pTemp = nullptr;
    std::swap(pTemp, m_pObj);
    return pTemp;
}

explicit operator bool() const { return !!m_pObj; }
T& operator*() const { return *m_pObj; }
T* operator->() const { return m_pObj; }

private:
    friend class pdfium::span<T>;

    inline void ProbeForLowSeverityLifetimeIssue() {
#ifdef ADDRESS_SANITIZER
        if (m_pObj)
            reinterpret_cast<const volatile uint8_t*>(m_pObj)[0];
#endif
    }

    inline void ReleaseBadPointer() {
#ifdef ADDRESS_SANITIZER
        m_pObj = nullptr;
#endif
    }
}
```

```
#endif
}

T* m_pObj = nullptr;
};

template <typename T, typename U>
inline bool operator==(const U* lhs, const UnownedPtr<T>& rhs) {
    return rhs == lhs;
}

template <typename T, typename U>
inline bool operator!=(const U* lhs, const UnownedPtr<T>& rhs) {
    return rhs != lhs;
}

} // namespace fxcrt

using fxcrt::UnownedPtr;

#endif // CORE_FXCRT_UNOWNED_PTR_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_WEAK_PTR_H_
#define CORE_FXCRT_WEAK_PTR_H_

#include <stddef>
#include <memory>
#include <utility>

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

namespace fxcrt {

template <class T, class D = std::default_delete<T>>
class WeakPtr {
public:
    WeakPtr() = default;
    WeakPtr(const WeakPtr& that) : m_pHandle(that.m_pHandle) {}
    WeakPtr(WeakPtr&& that) noexcept { Swap(that); }
    explicit WeakPtr(std::unique_ptr<T, D> pObj)
        : m_pHandle(new Handle(std::move(pObj))) {}

    // Deliberately implicit to allow passing nullptr.
    // NOLINTNEXTLINE(runtime/explicit)
    WeakPtr(std::nullptr_t arg) {}

    explicit operator bool() const { return m_pHandle && !m_pHandle->Get(); }
    bool HasOneRef() const { return m_pHandle && m_pHandle->HasOneRef(); }
    T* operator->() { return m_pHandle->Get(); }
    const T* operator->() const { return m_pHandle->Get(); }
    WeakPtr& operator=(const WeakPtr& that) {
        m_pHandle = that.m_pHandle;
        return *this;
    }
    bool operator==(const WeakPtr& that) const {
        return m_pHandle == that.m_pHandle;
    }
    bool operator!=(const WeakPtr& that) const { return !(*this == that); }

    T* Get() const { return m_pHandle ? m_pHandle->Get() : nullptr; }
    void DeleteObject() {
        if (m_pHandle) {
            m_pHandle->Clear();
            m_pHandle.Reset();
        }
    }
    void Reset() { m_pHandle.Reset(); }
    void Reset(std::unique_ptr<T, D> pObj) {
        m_pHandle.Reset(new Handle(std::move(pObj)));
    }
    void Swap(WeakPtr& that) { m_pHandle.Swap(that.m_pHandle); }

private:
    class Handle {
public:
        explicit Handle(std::unique_ptr<T, D> ptr) : m_pObj(std::move(ptr)) {}

        void Reset(std::unique_ptr<T, D> ptr) { m_pObj = std::move(ptr); }
    };
};

}  // namespace fxcrt
```

```
void Clear() { // Now you're all weak ptrs ...
    m_pObj.reset(); // unique_ptr nulls first before invoking delete.
}
T* Get() const { return m_pObj.get(); }
T* Retain() {
    ++m_nCount;
    return m_pObj.get();
}
void Release() {
    if (--m_nCount == 0)
        delete this;
}
bool HasOneRef() const { return m_nCount == 1; }

private:
    ~Handle() = default;

    intptr_t m_nCount = 0;
    std::unique_ptr<T, D> m_pObj;
};

RetainPtr<Handle> m_pHandle;
};

} // namespace fxcrt

using fxcrt::WeakPtr;

#endif // CORE_FXCRT_WEAK_PTR_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_WIDESTRING_H_
#define CORE_FXCRT_WIDESTRING_H_

#include <functional>
#include <iterator>
#include <ostream>
#include <utility>

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/string_data_template.h"
#include "core/fxcrt/string_view_template.h"
#include "third_party/base/logging.h"
#include "third_party/base/optional.h"
#include "third_party/base/span.h"

namespace fxcrt {

class ByteString;
class StringPool_WideString_Test;
class WideString_Assign_Test;
class WideString_ConcatInPlace_Test;

// A mutable string with shared buffers using copy-on-write semantics that
// avoids the cost of std::string's iterator stability guarantees.
class WideString {
public:
    using CharType = wchar_t;
    using const_iterator = const CharType*;
    using const_reverse_iterator = std::reverse_iterator<const_iterator>;

    static WideString Format(const wchar_t* pFormat, ...) WARN_UNUSED_RESULT;
    static WideString FormatV(const wchar_t* lpszFormat,
                             va_list argList) WARN_UNUSED_RESULT;

    WideString();
    WideString(const WideString& other);
    WideString(WideString&& other) noexcept;

    // Deliberately implicit to avoid calling on every string literal.
    // NOLINTNEXTLINE(runtime/explicit)
    WideString(wchar_t ch);
    // NOLINTNEXTLINE(runtime/explicit)
    WideString(const wchar_t* ptr);

    // No implicit conversions from byte strings.
    // NOLINTNEXTLINE(runtime/explicit)
    WideString(char) = delete;

    WideString(const wchar_t* pStr, size_t len);

    explicit WideString(WideStringView str);
    WideString(WideStringView str1, WideStringView str2);
    WideString(const std::initializer_list<WideStringView>& list);

    ~WideString();
};
```

```

static WideString FromASCII(ByteStringView str) WARN_UNUSED_RESULT;
static WideString FromLatin1(ByteStringView str) WARN_UNUSED_RESULT;
static WideString FromDefANSI(ByteStringView str) WARN_UNUSED_RESULT;
static WideString FromUTF8(ByteStringView str) WARN_UNUSED_RESULT;
static WideString FromUTF16LE(const unsigned short* str,
                               size_t len) WARN_UNUSED_RESULT;
static WideString FromUTF16BE(const unsigned short* wstr,
                               size_t wlen) WARN_UNUSED_RESULT;

static size_t WStringLength(const unsigned short* str) WARN_UNUSED_RESULT;

// Explicit conversion to C-style wide string.
// Note: Any subsequent modification of |this| will invalidate the result.
const wchar_t* c_str() const { return m_pData ? m_pData->m_String : L""; }

// Explicit conversion to WideStringView.
// Note: Any subsequent modification of |this| will invalidate the result.
WideStringView AsStringView() const {
    return WideStringView(c_str(), GetLength());
}

// Explicit conversion to span.
// Note: Any subsequent modification of |this| will invalidate the result.
pdfium::span<const wchar_t> span() const {
    return pdfium::make_span(m_pData ? m_pData->m_String : nullptr,
                           GetLength());
}

// Note: Any subsequent modification of |this| will invalidate iterators.
const_iterator begin() const { return m_pData ? m_pData->m_String : nullptr; }
const_iterator end() const {
    return m_pData ? m_pData->m_String + m_pData->m_nDataLength : nullptr;
}

// Note: Any subsequent modification of |this| will invalidate iterators.
const_reverse_iterator rbegin() const {
    return const_reverse_iterator(end());
}
const_reverse_iterator rend() const {
    return const_reverse_iterator(begin());
}

void clear() { m_pData.Reset(); }

size_t GetLength() const { return m_pData ? m_pData->m_nDataLength : 0; }
size_t GetStringLength() const {
    return m_pData ? wcslen(m_pData->m_String) : 0;
}
bool IsEmpty() const { return !GetLength(); }
bool IsValidIndex(size_t index) const { return index < GetLength(); }
bool IsValidLength(size_t length) const { return length <= GetLength(); }

WideString& operator=(const wchar_t* str);
WideString& operator=(WideStringView str);
WideString& operator=(const WideString& that);
WideString& operator=(WideString&& that);

WideString& operator+=(const wchar_t* str);
WideString& operator+=(wchar_t ch);
WideString& operator+=(const WideString& str);
WideString& operator+=(WideStringView str);

bool operator==(const wchar_t* ptr) const;

```

```
bool operator==(WideStringView str) const;
bool operator==(const WideString& other) const;

bool operator!=(const wchar_t* ptr) const { return !(*this == ptr); }
bool operator!=(WideStringView str) const { return !(*this == str); }
bool operator!=(const WideString& other) const { return !(*this == other); }

bool operator<(const wchar_t* ptr) const;
bool operator<(WideStringView str) const;
bool operator<(const WideString& other) const;

CharType operator[](const size_t index) const {
    CHECK(IsValidIndex(index));
    return m_pData->m_String[index];
}

CharType First() const { return GetLength() ? (*this)[0] : 0; }
CharType Last() const { return GetLength() ? (*this)[GetLength() - 1] : 0; }

void SetAt(size_t index, wchar_t c);

int Compare(const wchar_t* str) const;
int Compare(const WideString& str) const;
int CompareNoCase(const wchar_t* str) const;

WideString Mid(size_t first, size_t count) const;
WideString Left(size_t count) const;
WideString Right(size_t count) const;

size_t Insert(size_t index, wchar_t ch);
size_t InsertAtFront(wchar_t ch) { return Insert(0, ch); }
size_t InsertAtBack(wchar_t ch) { return Insert(GetLength(), ch); }
size_t Delete(size_t index, size_t count = 1);

void MakeLower();
void MakeUpper();

void Trim();
void Trim(wchar_t target);
void Trim(WideStringView targets);

void TrimLeft();
void TrimLeft(wchar_t target);
void TrimLeft(WideStringView targets);

void TrimRight();
void TrimRight(wchar_t target);
void TrimRight(WideStringView targets);

void Reserve(size_t len);

// Note: any modification of the string (including ReleaseBuffer()) may
// invalidate the span, which must not outlive its buffer.
pdfium::span<wchar_t> GetBuffer(size_t nMinBufLength);
void ReleaseBuffer(size_t nNewLength);

int GetInteger() const;

Optional<size_t> Find(WideStringView subStr, size_t start = 0) const;
Optional<size_t> Find(wchar_t ch, size_t start = 0) const;
Optional<size_t> ReverseFind(wchar_t ch) const;

bool Contains(WideStringView lpszSub, size_t start = 0) const {
```

```

    return Find(lpszSub, start).has_value();
}

bool Contains(char ch, size_t start = 0) const {
    return Find(ch, start).has_value();
}

size_t Replace(WideStringView pOld, WideStringView pNew);
size_t Remove(wchar_t ch);

bool IsASCII() const { return AsStringView().IsASCII(); }
bool EqualsASCII(ByteStringView that) const {
    return AsStringView().EqualsASCII(that);
}
bool EqualsASCIIIgnoreCase(ByteStringView that) const {
    return AsStringView().EqualsASCIIIgnoreCase(that);
}

ByteString ToASCII() const;
ByteString ToLatin1() const;
ByteString ToDefANSI() const;
ByteString ToUTF8() const;

// This method will add \0\0 to the end of the string to represent the
// wide string terminator. These values are in the string, not just the data,
// so GetLength() will include them.
ByteString ToUTF16LE() const;

protected:
using StringData = StringDataTemplate<wchar_t>;

void ReallocBeforeWrite(size_t nNewLength);
void AllocBeforeWrite(size_t nNewLength);
void AllocCopy(WideString& dest, size_t nCopyLen, size_t nCopyIndex) const;
void AssignCopy(const wchar_t* pSrcData, size_t nSrcLen);
void Concat(const wchar_t* pSrcData, size_t nSrcLen);
intptr_t ReferenceCountForTesting() const;

RetainPtr<StringData> m_pData;

friend WideString_ConcatInPlace_Test;
friend WideString_Assign_Test;
friend StringPool_WideString_Test;
};

inline WideString operator+(WideStringView str1, WideStringView str2) {
    return WideString(str1, str2);
}
inline WideString operator+(WideStringView str1, const wchar_t* str2) {
    return WideString(str1, str2);
}
inline WideString operator+(const wchar_t* str1, WideStringView str2) {
    return WideString(str1, str2);
}
inline WideString operator+(WideStringView str1, wchar_t ch) {
    return WideString(str1, WideStringView(ch));
}
inline WideString operator+(wchar_t ch, WideStringView str2) {
    return WideString(ch, str2);
}
inline WideString operator+(const WideString& str1, const WideString& str2) {
    return WideString(str1.AsStringView(), str2.AsStringView());
}

```

```
inline WideString operator+(const WideString& str1, wchar_t ch) {
    return WideString(str1.AsStringView(), WideStringView(ch));
}
inline WideString operator+(wchar_t ch, const WideString& str2) {
    return WideString(ch, str2.AsStringView());
}
inline WideString operator+(const WideString& str1, const wchar_t* str2) {
    return WideString(str1.AsStringView(), str2);
}
inline WideString operator+(const wchar_t* str1, const WideString& str2) {
    return WideString(str1, str2.AsStringView());
}
inline WideString operator+(const WideString& str1, WideStringView str2) {
    return WideString(str1.AsStringView(), str2);
}
inline WideString operator+(WideStringView str1, const WideString& str2) {
    return WideString(str1, str2.AsStringView());
}
inline bool operator==(const wchar_t* lhs, const WideString& rhs) {
    return rhs == lhs;
}
inline bool operator==(WideStringView lhs, const WideString& rhs) {
    return rhs == lhs;
}
inline bool operator!=(const wchar_t* lhs, const WideString& rhs) {
    return rhs != lhs;
}
inline bool operator!=(WideStringView lhs, const WideString& rhs) {
    return rhs != lhs;
}
inline bool operator<(const wchar_t* lhs, const WideString& rhs) {
    return rhs.Compare(lhs) > 0;
}

std::wostream& operator<<(std::wostream& os, const WideString& str);
std::ostream& operator<<(std::ostream& os, const WideString& str);
std::wostream& operator<<(std::wostream& os, WideStringView str);
std::ostream& operator<<(std::ostream& os, WideStringView str);

} // namespace fxcrt

using WideString = fxcrt::WideString;

uint32_t FX_HashCode_GetW(WideStringView str, bool bIgnoreCase);

namespace std {

template <>
struct hash<WideString> {
    std::size_t operator()(const WideString& str) const {
        return FX_HashCode_GetW(str.AsStringView(), false);
    }
};

} // namespace std

extern template struct std::hash<WideString>;

#endif // CORE_FXCRT_WIDESTRING_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_XML_CFX_XMLCHARDATA_H
#define CORE_FXCRT_XML_CFX_XMLCHARDATA_H

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/xml/cfx_xmltext.h"

class CFX_XMLDocument;

class CFX_XMLCharData final : public CFX_XMLText {
public:
    explicit CFX_XMLCharData(const WideString& wsCData);
    ~CFX_XMLCharData() override;

    // CFX_XMLNode
    Type GetType() const override;
    CFX_XMLNode* Clone(CFX_XMLDocument* doc) override;
    void Save(const RetainPtr<IFX_SeekableWriteStream>& pXMLStream) override;
};

inline CFX_XMLCharData* ToXMLCharData(CFX_XMLNode* pNode) {
    return pNode && pNode->GetType() == CFX_XMLNode::Type::kCharData
        ? static_cast<CFX_XMLCharData*>(pNode)
        : nullptr;
}

#endif // CORE_FXCRT_XML_CFX_XMLCHARDATA_H
```

```
// Copyright 2018 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

#ifndef CORE_FXCRT_XML_CFX_XMLDOCUMENT_H_
#define CORE_FXCRT_XML_CFX_XMLDOCUMENT_H_

#include <memory>
#include <utility>
#include <vector>

#include "core/fxcrt/unowned_ptr.h"
#include "core/fxcrt/xml/cfx_xmlelement.h"
#include "third_party/base/ptr_util.h"

class CFX_XMLNode;

class CFX_XMLDocument {
public:
  CFX_XMLDocument();
  ~CFX_XMLDocument();

  CFX_XMLElement* GetRoot() const { return root_.Get(); }

  template <typename T, typename... Args>
  T* CreateNode(Args&&... args) {
    nodes_.push_back(pdfium::MakeUnique<T>(std::forward<Args>(args)...));
    return static_cast<T*>(nodes_.back().get());
  }

  // Transfers ownership of entries in |nodes_| from |other| to |this|.
  // This is used in CFX_Node::loadXML to transfer ownership of the newly
  // created nodes to the top-level XML doc for the PDF, after parsing an XML
  // blob.
  void AppendNodesFrom(CFX_XMLDocument* other);

private:
  std::vector<std::unique_ptr<CFX_XMLNode>> nodes_;
  UnownedPtr<CFX_XMLElement> root_;
};

#endif // CORE_FXCRT_XML_CFX_XMLDOCUMENT_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_XML_CFX_XMLELEMENT_H_
#define CORE_FXCRT_XML_CFX_XMLELEMENT_H_

#include <map>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/xml/cfx_xmlnode.h"

class CFX_XMLDocument;

class CFX_XMLElement final : public CFX_XMLNode {
public:
    explicit CFX_XMLElement(const WideString& wsTag);
    ~CFX_XMLElement() override;

    // CFX_XMLNode
    Type GetType() const override;
    CFX_XMLNode* Clone(CFX_XMLDocument* doc) override;
    void Save(const RetainPtr<IFX_SeekableWriteStream>& pXMLStream) override;

    const WideString& GetName() const { return name_; }

    const std::map<WideString, WideString>& GetAttributes() const {
        return attrs_;
    }
    bool HasAttribute(const WideString& name) const;
    void SetAttribute(const WideString& name, const WideString& value);
    WideString GetAttribute(const WideString& name) const;

    void RemoveAttribute(const WideString& name);

    CFX_XMLElement* GetFirstChildNamed(WideStringView name) const;
    CFX_XMLElement* GetNthChildNamed(WideStringView name, size_t idx) const;

    WideString GetLocalTagName() const;
    WideString GetNamespacePrefix() const;
    WideString GetNamespaceURI() const;

    WideString GetTextData() const;

private:
    WideString AttributeToString(const WideString& name, const WideString& value);

    const WideString name_;
    std::map<WideString, WideString> attrs_;
};

inline CFX_XMLElement* ToXMLElement(CFX_XMLNode* pNode) {
    return pNode && pNode->GetType() == CFX_XMLNode::Type::kElement
        ? static_cast<CFX_XMLElement*>(pNode)
        : nullptr;
}

inline const CFX_XMLElement* ToXMLElement(const CFX_XMLNode* pNode) {
    return pNode && pNode->GetType() == CFX_XMLNode::Type::kElement
        ? static_cast<const CFX_XMLElement*>(pNode)
        : nullptr;
}
```

```
}  
  
#endif // CORE_FXCRT_XML_CFX_XMLELEMENT_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_XML_CFX_XMLINSTRUCTION_H_
#define CORE_FXCRT_XML_CFX_XMLINSTRUCTION_H_

#include <vector>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/xml/cfx_xmlnode.h"

class CFX_XMLDocument;

class CFX_XMLInstruction final : public CFX_XMLNode {
public:
    explicit CFX_XMLInstruction(const WideString& wsTarget);
    ~CFX_XMLInstruction() override;

    // CFX_XMLNode
    Type GetType() const override;
    CFX_XMLNode* Clone(CFX_XMLDocument* doc) override;
    void Save(const RetainPtr<IFX_SeekableWriteStream>& pXMLStream) override;

    bool IsOriginalXFAVersion() const;
    bool IsAcrobat() const;

    const std::vector<WideString>& GetTargetData() const { return target_data_; }
    void AppendData(const WideString& wsData);

private:
    const WideString name_;
    std::vector<WideString> target_data_;
};

inline CFX_XMLInstruction* ToXMLInstruction(CFX_XMLNode* pNode) {
    return pNode && pNode->GetType() == CFX_XMLNode::Type::kInstruction
        ? static_cast<CFX_XMLInstruction*>(pNode)
        : nullptr;
}

#endif // CORE_FXCRT_XML_CFX_XMLINSTRUCTION_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_XML_CFX_XMLNODE_H_
#define CORE_FXCRT_XML_CFX_XMLNODE_H_

#include "core/fxcrt/fx_stream.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/tree_node.h"

class CFX_XMLDocument;

class CFX_XMLNode : public TreeNode<CFX_XMLNode> {
public:
    enum class Type {
        kInstruction = 0,
        kElement,
        kText,
        kCharData,
    };

    CFX_XMLNode();
    ~CFX_XMLNode() override;

    virtual Type GetType() const = 0;
    virtual CFX_XMLNode* Clone(CFX_XMLDocument* doc) = 0;
    virtual void Save(const RetainPtr<IFX_SeekableWriteStream>& pXMLStream) = 0;

    CFX_XMLNode* GetRoot();
    void InsertChildNode(CFX_XMLNode* pNode, int32_t index);

protected:
    WideString EncodeEntities(const WideString& value);
};

#endif // CORE_FXCRT_XML_CFX_XMLNODE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_XML_CFX_XMLPARSER_H
#define CORE_FXCRT_XML_CFX_XMLPARSER_H

#include <memory>
#include <vector>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"

class CFX_SeekableStreamProxy;
class CFX_XMLDocument;
class CFX_XMLElement;
class CFX_XMLNode;
class IFX_SeekableReadStream;

class CFX_XMLParser final {
public:
    static bool IsXMLNameChar(wchar_t ch, bool bFirstChar);

    explicit CFX_XMLParser(const RetainPtr<IFX_SeekableReadStream>& pStream);
    ~CFX_XMLParser();

    std::unique_ptr<CFX_XMLDocument> Parse();

private:
    enum class FDE_XmlSyntaxState {
        Text,
        Node,
        Target,
        Tag,
        AttriName,
        AttriEqualSign,
        AttriQuotation,
        AttriValue,
        CloseInstruction,
        BreakElement,
        CloseElement,
        SkipDeclNode,
        SkipComment,
        SkipCommentOrDecl,
        SkipCDATA,
        TargetData
    };

    bool DoSyntaxParse(CFX_XMLDocument* doc);
    WideString GetTextData();
    void ProcessTextChar(wchar_t ch);
    void ProcessTargetData();

    CFX_XMLNode* current_node_ = nullptr;
    RetainPtr<CFX_SeekableStreamProxy> stream_;
    std::vector<wchar_t, FxAllocAllocator<wchar_t>> current_text_;
    size_t xml_plane_size_ = 1024;
    int32_t entity_start_ = -1;
};
```

#endif // CORE\_FXCRT\_XML\_CFX\_XMLPARSER\_H\_

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXCRT_XML_CFX_XMLTEXT_H_
#define CORE_FXCRT_XML_CFX_XMLTEXT_H_

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/xml/cfx_xmlnode.h"

class CFX_XMLDocument;

class CFX_XMLText : public CFX_XMLNode {
public:
  explicit CFX_XMLText(const WideString& wsText);
  ~CFX_XMLText() override;

  // CFX_XMLNode
  Type GetType() const override;
  CFX_XMLNode* Clone(CFX_XMLDocument* doc) override;
  void Save(const RetainPtr<IFX_SeekableWriteStream>& pXMLStream) override;

  const WideString& GetText() const { return text_; }
  void SetText(const WideString& wsText) { text_ = wsText; }

private:
  WideString text_;
};

inline bool IsXMLText(const CFX_XMLNode* pNode) {
  CFX_XMLNode::Type type = pNode->GetType();
  return type == CFX_XMLNode::Type::kText ||
         type == CFX_XMLNode::Type::kCharData;
}

inline CFX_XMLText* ToXMLText(CFX_XMLNode* pNode) {
  return pNode && IsXMLText(pNode) ? static_cast<CFX_XMLText*>(pNode) : nullptr;
}

inline const CFX_XMLText* ToXMLText(const CFX_XMLNode* pNode) {
  return pNode && IsXMLText(pNode) ? static_cast<const CFX_XMLText*>(pNode)
                                     : nullptr;
}

#endif // CORE_FXCRT_XML_CFX_XMLTEXT_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_AGG_FX_AGG_DRIVER_H_
#define CORE_FXGE_AGG_FX_AGG_DRIVER_H_

#include <memory>
#include <vector>

#include "build/build_config.h"
#include "core/fxge/renderdevicedriver_iface.h"
#include "third_party/agg23/agg_clip_liang_barsky.h"
#include "third_party/agg23/agg_path_storage.h"
#include "third_party/agg23/agg_rasterizer_scanline_aa.h"

class CFX_ClipRgn;
class CFX_GraphStateData;
class CFX_Matrix;
class CFX_PathData;

class CAgg_PathData {
public:
    CAgg_PathData() {}
    ~CAgg_PathData() {}
    void BuildPath(const CFX_PathData* pPathData,
                  const CFX_Matrix* pObject2Device);

    agg::path_storage m_PathData;
};

class CFX_AggDeviceDriver final : public RenderDeviceDriverIface {
public:
    CFX_AggDeviceDriver(const RetainPtr<CFX_DIBitmap>& pBitmap,
                       bool bRgbByteOrder,
                       const RetainPtr<CFX_DIBitmap>& pBackdropBitmap,
                       bool bGroupKnockout);
    ~CFX_AggDeviceDriver() override;

    void InitPlatform();
    void DestroyPlatform();

    // RenderDeviceDriverIface:
    DeviceType GetDeviceType() const override;
    int GetDeviceCaps(int caps_id) const override;
    void SaveState() override;
    void RestoreState(bool bKeepSaved) override;
    bool SetClip_PathFill(const CFX_PathData* pPathData,
                         const CFX_Matrix* pObject2Device,
                         int fill_mode) override;
    bool SetClip_PathStroke(const CFX_PathData* pPathData,
                            const CFX_Matrix* pObject2Device,
                            const CFX_GraphStateData* pGraphState) override;
    bool DrawPath(const CFX_PathData* pPathData,
                 const CFX_Matrix* pObject2Device,
                 const CFX_GraphStateData* pGraphState,
                 uint32_t fill_color,
                 uint32_t stroke_color,
                 int fill_mode,
                 BlendMode blend_type) override;
    bool SetPixel(int x, int y, uint32_t color) override;
};
```

```
bool FillRectWithBlend(const FX_RECT& rect,
                      uint32_t fill_color,
                      BlendMode blend_type) override;
bool GetClipBox(FX_RECT* pRect) override;
bool GetDIBits(const RetainPtr<CFX_DIBitmap>& pBitmap,
              int left,
              int top) override;
RetainPtr<CFX_DIBitmap> GetBackDrop() override;
bool SetDIBits(const RetainPtr<CFX_DIBase>& pBitmap,
              uint32_t argb,
              const FX_RECT& src_rect,
              int left,
              int top,
              BlendMode blend_type) override;
bool StretchDIBits(const RetainPtr<CFX_DIBase>& pSource,
                  uint32_t argb,
                  int dest_left,
                  int dest_top,
                  int dest_width,
                  int dest_height,
                  const FX_RECT* pClipRect,
                  const FXDIB_ResampleOptions& options,
                  BlendMode blend_type) override;
bool StartDIBits(const RetainPtr<CFX_DIBase>& pSource,
                int bitmap_alpha,
                uint32_t argb,
                const CFX_Matrix& matrix,
                const FXDIB_ResampleOptions& options,
                std::unique_ptr<CFX_ImageRenderer>* handle,
                BlendMode blend_type) override;
bool ContinueDIBits(CFX_ImageRenderer* handle,
                  PauseIndicatorIface* pPause) override;
bool DrawDeviceText(int nChars,
                   const TextCharPos* pCharPos,
                   CFX_Font* pFont,
                   const CFX_Matrix& mtObject2Device,
                   float font_size,
                   uint32_t color) override;
int GetDriverType() const override;

bool RenderRasterizer(agg::rasterizer_scanline_aa& rasterizer,
                     uint32_t color,
                     bool bFullCover,
                     bool bGroupKnockout);

void SetClipMask(agg::rasterizer_scanline_aa& rasterizer);

virtual uint8_t* GetBuffer() const;

private:
RetainPtr<CFX_DIBitmap> m_pBitmap;
std::unique_ptr<CFX_ClipRgn> m_pClipRgn;
std::vector<std::unique_ptr<CFX_ClipRgn>> m_StateStack;
#ifdef OS_MACOSX
void* m_pPlatformGraphics = nullptr;
#endif
int m_FillFlags = 0;
const bool m_bRgbByteOrder;
const bool m_bGroupKnockout;
RetainPtr<CFX_DIBitmap> m_pBackdropBitmap;
};

#endif // CORE_FXGE_AGG_FX_AGG_DRIVER_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_ANDROID_CFPF_SKIADEVICEMODULE_H_
#define CORE_FXGE_ANDROID_CFPF_SKIADEVICEMODULE_H_

#include <memory>

class CFPF_SkiaFontMgr;

class CFPF_SkiaDeviceModule {
public:
  CFPF_SkiaDeviceModule();
  ~CFPF_SkiaDeviceModule();

  void Destroy();
  CFPF_SkiaFontMgr* GetFontMgr();

protected:
  std::unique_ptr<CFPF_SkiaFontMgr> m_pFontMgr;
};

CFPF_SkiaDeviceModule* CFPF_GetSkiaDeviceModule();

#endif // CORE_FXGE_ANDROID_CFPF_SKIADEVICEMODULE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_ANDROID_CFPF_SKIAFONT_H_
#define CORE_FXGE_ANDROID_CFPF_SKIAFONT_H_

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/cfx_face.h"
#include "core/fxge/fx_freetype.h"
#include "third_party/base/span.h"

class CFPF_SkiaFontMgr;
class CFPF_SkiaPathFont;
struct FX_RECT;

class CFPF_SkiaFont {
public:
  CFPF_SkiaFont(CFPF_SkiaFontMgr* pFontMgr,
               const CFPF_SkiaPathFont* pFont,
               uint32_t dwStyle,
               uint8_t uCharset);
  ~CFPF_SkiaFont();

  bool IsValid() const { return !!m_Face; }

  ByteString GetFamilyName();
  ByteString GetPsName();
  uint32_t GetFontStyle() const { return m_dwStyle; }
  uint8_t GetCharset() const { return m_uCharset; }
  int32_t GetGlyphIndex(wchar_t wUnicode);
  int32_t GetGlyphWidth(int32_t iGlyphIndex);
  int32_t GetAscent() const;
  int32_t GetDescent() const;
  bool GetGlyphBBox(int32_t iGlyphIndex, FX_RECT& rtBBox);
  bool GetBBox(FX_RECT& rtBBox);
  int32_t GetHeight() const;
  int32_t GetItalicAngle() const;
  uint32_t GetFontData(uint32_t dwTable, pdfium::span<uint8_t> pBuffer);
  FXFT_FaceRec* GetFaceRec() const { return m_Face->GetRec(); }

private:
  UnownedPtr<CFPF_SkiaFontMgr> const m_pFontMgr;
  UnownedPtr<const CFPF_SkiaPathFont> const m_pFont;
  RetainPtr<CFX_Face> const m_Face;
  const uint32_t m_dwStyle;
  const uint8_t m_uCharset;
};

#endif // CORE_FXGE_ANDROID_CFPF_SKIAFONT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_ANDROID_CFPF_SKIAFONTMGR_H_
#define CORE_FXGE_ANDROID_CFPF_SKIAFONTMGR_H_

#include <map>
#include <memory>
#include <vector>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/cfx_face.h"
#include "core/fxge/fx_freetype.h"

class CFPF_SkiaFont;
class CFPF_SkiaPathFont;

class CFPF_SkiaFontMgr {
public:
  CFPF_SkiaFontMgr();
  ~CFPF_SkiaFontMgr();

  void LoadSystemFonts();
  CFPF_SkiaFont* CreateFont(ByteStringView bsFamilyname,
                           uint8_t uCharset,
                           uint32_t dwStyle);

  bool InitFTLibrary();
  RetainPtr<CFX_Face> GetFontFace(ByteStringView bsFile, int32_t iFaceIndex);

private:
  void ScanPath(const ByteString& path);
  void ScanFile(const ByteString& file);
  std::unique_ptr<CFPF_SkiaPathFont> ReportFace(RetainPtr<CFX_Face> face,
                                               const ByteString& file);

  bool m_bLoaded = false;
  ScopedFXFTLibraryRec m_FTLibrary;
  std::vector<std::unique_ptr<CFPF_SkiaPathFont>> m_FontFaces;
  std::map<uint32_t, std::unique_ptr<CFPF_SkiaFont>> m_FamilyFonts;
};

#endif // CORE_FXGE_ANDROID_CFPF_SKIAFONTMGR_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_ANDROID_CFPF_SKIAPATHFONT_H_
#define CORE_FXGE_ANDROID_CFPF_SKIAPATHFONT_H_

#include "core/fxcrt/bytestring.h"
#include "core/fxcrt/fx_system.h"

class CFPF_SkiaPathFont {
public:
  CFPF_SkiaPathFont(const ByteString& path,
                    const char* pFamily,
                    uint32_t dwStyle,
                    int32_t iFaceIndex,
                    uint32_t dwCharsets,
                    int32_t iGlyphNum);
  ~CFPF_SkiaPathFont();

  const char* path() const { return m_bsPath.c_str(); }
  const char* family() const { return m_bsFamily.c_str(); }
  uint32_t style() const { return m_dwStyle; }
  int32_t face_index() const { return m_iFaceIndex; }
  uint32_t charsets() const { return m_dwCharsets; }
  int32_t glyph_num() const { return m_iGlyphNum; }

private:
  const ByteString m_bsPath;
  const ByteString m_bsFamily;
  const uint32_t m_dwStyle;
  const int32_t m_iFaceIndex;
  const uint32_t m_dwCharsets;
  const int32_t m_iGlyphNum;
};

#endif // CORE_FXGE_ANDROID_CFPF_SKIAPATHFONT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_ANDROID_CFX_ANDROIDFONTINFO_H_
#define CORE_FXGE_ANDROID_CFX_ANDROIDFONTINFO_H_

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/cfx_fontmapper.h"
#include "core/fxge/systemfontinfo_iface.h"
#include "third_party/base/span.h"

class CFPF_SkiaFontMgr;

class CFX_AndroidFontInfo final : public SystemFontInfoIface {
public:
  CFX_AndroidFontInfo();
  ~CFX_AndroidFontInfo() override;

  bool Init(CFPF_SkiaFontMgr* pFontMgr);

  // SystemFontInfoIface:
  bool EnumFontList(CFX_FontMapper* pMapper) override;
  void* MapFont(int weight,
               bool bItalic,
               int charset,
               int pitch_family,
               const char* face) override;
  void* GetFont(const char* face) override;
  uint32_t GetFontData(void* hFont,
                      uint32_t table,
                      pdfium::span<uint8_t> buffer) override;
  bool GetFaceName(void* hFont, ByteString* name) override;
  bool GetFontCharset(void* hFont, int* charset) override;
  void DeleteFont(void* hFont) override;

private:
  UnownedPtr<CFPF_SkiaFontMgr> m_pFontMgr;
};

#endif // CORE_FXGE_ANDROID_CFX_ANDROIDFONTINFO_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_APPLE_APPLE_INT_H_
#define CORE_FXGE_APPLE_APPLE_INT_H_

#include "core/fxcrt/fx_system.h"

#include <Carbon/Carbon.h>

#include "core/fxge/cfx_gemodule.h"
#include "core/fxge/cfx_graphstatedata.h"
#include "core/fxge/cfx_pathdata.h"
#include "core/fxge/fx_dib.h"
#include "core/fxge/renderdevicedriver_iface.h"

class CQuartz2D {
public:
    void* CreateGraphics(const RetainPtr<CFX_DIBitmap>& bitmap);
    void DestroyGraphics(void* graphics);

    void* CreateFont(const uint8_t* pFontData, uint32_t dwFontSize);
    void DestroyFont(void* pFont);
    void SetGraphicsTextMatrix(void* graphics, const CFX_Matrix& matrix);
    bool DrawGraphicsString(void* graphics,
                            void* font,
                            float fontSize,
                            uint16_t* glyphIndices,
                            CGPoint* glyphPositions,
                            int32_t chars,
                            FX_ARGB argb);
};

class CApplePlatform : public CFX_GEModule::PlatformIface {
public:
    CApplePlatform();
    ~CApplePlatform() override;

    // CFX_GEModule::PlatformIface:
    void Init() override;

    CQuartz2D m_quartz2d;
};

#endif // CORE_FXGE_APPLE_APPLE_INT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_CLIPRGN_H_
#define CORE_FXGE_CFX_CLIPRGN_H_

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"

class CFX_DIBitmap;

class CFX_ClipRgn {
public:
    enum ClipType { RectI, MaskF };

    CFX_ClipRgn(int device_width, int device_height);
    CFX_ClipRgn(const CFX_ClipRgn& src);
    ~CFX_ClipRgn();

    ClipType GetType() const { return m_Type; }
    const FX_RECT& GetBox() const { return m_Box; }
    RetainPtr<CFX_DIBitmap> GetMask() const { return m_Mask; }

    void IntersectRect(const FX_RECT& rect);
    void IntersectMaskF(int left, int top, const RetainPtr<CFX_DIBitmap>& Mask);

private:
    void IntersectMaskRect(FX_RECT rect,
                          FX_RECT mask_rect,
                          const RetainPtr<CFX_DIBitmap>& Mask);

    ClipType m_Type;
    FX_RECT m_Box;
    RetainPtr<CFX_DIBitmap> m_Mask;
};

#endif // CORE_FXGE_CFX_CLIPRGN_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_COLOR_H_
#define CORE_FXGE_CFX_COLOR_H_

#include "core/fxge/fx_dib.h"

struct CFX_Color {
  // Ordered by increasing number of components.
  enum Type { kTransparent = 0, kGray, kRGB, kCMYK };

  explicit CFX_Color(FX_COLORREF ref)
    : CFX_Color(FXARGB_R(ref), FXARGB_G(ref), FXARGB_B(ref)) {}

  CFX_Color(int32_t type = CFX_Color::kTransparent,
            float color1 = 0.0f,
            float color2 = 0.0f,
            float color3 = 0.0f,
            float color4 = 0.0f)
    : nColorType(type),
      fColor1(color1),
      fColor2(color2),
      fColor3(color3),
      fColor4(color4) {}

  CFX_Color(int32_t r, int32_t g, int32_t b)
    : nColorType(CFX_Color::kRGB),
      fColor1(r / 255.0f),
      fColor2(g / 255.0f),
      fColor3(b / 255.0f),
      fColor4(0) {}

  CFX_Color(const CFX_Color&) = default;

  CFX_Color operator/(float fColorDivide) const;
  CFX_Color operator-(float fColorSub) const;

  CFX_Color ConvertColorType(int32_t nConvertColorType) const;

  FX_COLORREF ToFXColor(int32_t nTransparency) const;

  void Reset() {
    nColorType = CFX_Color::kTransparent;
    fColor1 = 0.0f;
    fColor2 = 0.0f;
    fColor3 = 0.0f;
    fColor4 = 0.0f;
  }

  int32_t nColorType;
  float fColor1;
  float fColor2;
  float fColor3;
  float fColor4;
};

inline bool operator==(const CFX_Color& c1, const CFX_Color& c2) {
  return c1.nColorType == c2.nColorType && c1.fColor1 - c2.fColor1 < 0.0001 &&
    c1.fColor1 - c2.fColor1 > -0.0001 &&

```

```
    c1.fColor2 - c2.fColor2 < 0.0001 &&  
    c1.fColor2 - c2.fColor2 > -0.0001 &&  
    c1.fColor3 - c2.fColor3 < 0.0001 &&  
    c1.fColor3 - c2.fColor3 > -0.0001 &&  
    c1.fColor4 - c2.fColor4 < 0.0001 && c1.fColor4 - c2.fColor4 > -0.0001;  
}  
  
inline bool operator!=(const CFX_Color& c1, const CFX_Color& c2) {  
    return !(c1 == c2);  
}  
  
#endif // CORE_FXGE_CFX_COLOR_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_DEFAULTRENDERDEVICE_H_
#define CORE_FXGE_CFX_DEFAULTRENDERDEVICE_H_

#include "core/fxge/cfx_renderdevice.h"
#include "core/fxge/fx_dib.h"

class SkPictureRecorder;

class CFX_DefaultRenderDevice final : public CFX_RenderDevice {
public:
  CFX_DefaultRenderDevice();
  ~CFX_DefaultRenderDevice() override;

  bool Attach(const RetainPtr<CFX_DIBitmap>& pBitmap,
              bool bRgbByteOrder,
              const RetainPtr<CFX_DIBitmap>& pBackdropBitmap,
              bool bGroupKnockout);
  bool Create(int width,
              int height,
              FXDIB_Format format,
              const RetainPtr<CFX_DIBitmap>& pBackdropBitmap);

#ifdef _SKIA_SUPPORT_
  bool AttachRecorder(SkPictureRecorder* recorder);
  void Clear(uint32_t color);
  SkPictureRecorder* CreateRecorder(int size_x, int size_y);
  void DebugVerifyBitmapIsPreMultiplied() const override;
  bool SetBitsWithMask(const RetainPtr<CFX_DIBBase>& pBitmap,
                       const RetainPtr<CFX_DIBBase>& pMask,
                       int left,
                       int top,
                       int bitmap_alpha,
                       BlendMode blend_type) override;
#endif
};

#endif // CORE_FXGE_CFX_DEFAULTRENDERDEVICE_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.  
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
#ifndef CORE_FXGE_CFX_FACE_H_  
#define CORE_FXGE_CFX_FACE_H_
```

```
#include "core/fxcrt/observed_ptr.h"  
#include "core/fxcrt/retain_ptr.h"  
#include "core/fxge/fx_freetype.h"  
#include "third_party/base/span.h"
```

```
class CFX_Face : public Retainable, public Observable {  
public:  
    static RetainPtr<CFX_Face> New(FT_Library library,  
                                   const RetainPtr<Retainable>& pDesc,  
                                   pdfium::span<const FT_Byte> data,  
                                   FT_Long face_index);  
  
    static RetainPtr<CFX_Face> Open(FT_Library library,  
                                    const FT_Open_Args* args,  
                                    FT_Long face_index);  
  
    ~CFX_Face() override;  
  
    FXFT_FaceRec* GetRec() { return m_pRec.get(); }  
  
private:  
    CFX_Face(FXFT_FaceRec* pRec, const RetainPtr<Retainable>& pDesc);  
  
    ScopedFXFTFaceRec const m_pRec;  
    RetainPtr<Retainable> const m_pDesc;  
};  
  
#endif // CORE_FXGE_CFX_FACE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_FOLDERFONTINFO_H
#define CORE_FXGE_CFX_FOLDERFONTINFO_H

#include <map>
#include <memory>
#include <vector>

#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/cfx_fontmapper.h"
#include "core/fxge/systemfontinfo_iface.h"

class CFX_FolderFontInfo : public SystemFontInfoIface {
public:
    CFX_FolderFontInfo();
    ~CFX_FolderFontInfo() override;

    void AddPath(const ByteString& path);

    // IFX_SystemFontInfo:
    bool EnumFontList(CFX_FontMapper* pMapper) override;
    void* MapFont(int weight,
                 bool bItalic,
                 int charset,
                 int pitch_family,
                 const char* family) override;
    void* GetFont(const char* face) override;
    uint32_t GetFontData(void* hFont,
                        uint32_t table,
                        pdfium::span<uint8_t> buffer) override;
    void DeleteFont(void* hFont) override;
    bool GetFaceName(void* hFont, ByteString* name) override;
    bool GetFontCharset(void* hFont, int* charset) override;

protected:
    class FontFaceInfo {
    public:
        FontFaceInfo(ByteString filePath,
                    ByteString faceName,
                    ByteString fontTables,
                    uint32_t fontOffset,
                    uint32_t fileSize);

        const ByteString m_FilePath;
        const ByteString m_FaceName;
        const ByteString m_FontTables;
        const uint32_t m_FontOffset;
        const uint32_t m_FileSize;
        uint32_t m_Styles;
        uint32_t m_Charsets;
    };

    void ScanPath(const ByteString& path);
    void ScanFile(const ByteString& path);
    void ReportFace(const ByteString& path,
                  FILE* pFile,
                  uint32_t filesize,
                  uint32_t offset);
};
```

```
void* GetSubstFont(const ByteString& face);
void* FindFont(int weight,
               bool bItalic,
               int charset,
               int pitch_family,
               const char* family,
               bool bMatchName);

std::map<ByteString, std::unique_ptr<FontFaceInfo>> m_FontList;
std::vector<ByteString> m_PathList;
UnownedPtr<CFX_FontMapper> m_pMapper;
};

#endif // CORE_FXGE_CFX_FOLDERFONTINFO_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_FONTCACHE_H_
#define CORE_FXGE_CFX_FONTCACHE_H_

#include <map>
#include <memory>

#include "core/fxcrt/fx_system.h"
#include "core/fxge/cfx_glyphcache.h"
#include "core/fxge/fx_freetype.h"

class CFX_Font;

class CFX_FontCache {
public:
  CFX_FontCache();
  ~CFX_FontCache();

  RetainPtr<CFX_GlyphCache> GetGlyphCache(const CFX_Font* pFont);
#ifdef _SKIA_SUPPORT_
  CFX_TypeFace* GetDeviceCache(const CFX_Font* pFont);
#endif

private:
  std::map<CFX_Face*, ObservedPtr<CFX_GlyphCache>> m_GlyphCacheMap;
  std::map<CFX_Face*, ObservedPtr<CFX_GlyphCache>> m_ExtGlyphCacheMap;
};

#endif // CORE_FXGE_CFX_FONTCACHE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_FONT_H_
#define CORE_FXGE_CFX_FONT_H_

#include <memory>
#include <vector>

#include "build/build_config.h"
#include "core/foxcrt/bytestring.h"
#include "core/foxcrt/fox_coordinates.h"
#include "core/foxcrt/fox_memory_wrappers.h"
#include "core/foxge/cfx_face.h"
#include "core/foxge/fox_freetype.h"
#include "third_party/base/span.h"

#if defined _SKIA_SUPPORT_ || defined _SKIA_SUPPORT_PATHS_
#include "core/foxge/fox_font.h"
#endif

class CFX_GlyphCache;
class CFX_GlyphBitmap;
class CFX_PathData;
class CFX_SubstFont;
class IFX_SeekableReadStream;

class CFX_Font {
public:
  CFX_Font();
  ~CFX_Font();

  // Used when the font name is empty.
  static const char kUntitledFontName[];

  static const char kDefaultAnsiFontName[];
  static const char kUniversalDefaultFontName[];
  static ByteString GetDefaultFontNameByCharset(uint8_t nCharset);
  static uint8_t GetCharSetFromUnicode(uint16_t word);

  void LoadSubst(const ByteString& face_name,
                bool bTrueType,
                uint32_t flags,
                int weight,
                int italic_angle,
                int CharsetCP,
                bool bVertical);

  bool LoadEmbedded(pdfium::span<const uint8_t> src_span,
                   bool bForceAsVertical);
  RetainPtr<CFX_Face> GetFace() const { return m_Face; }
  FXFT_FaceRec* GetFaceRec() const {
    return m_Face ? m_Face->GetRec() : nullptr;
  }
  CFX_SubstFont* GetSubstFont() const { return m_pSubstFont.get(); }
  int GetSubstFontItalicAngle() const;
};

#if defined(PDF_ENABLE_XFA)
bool LoadFile(const RetainPtr<IFX_SeekableReadStream>& pFile, int nFaceIndex);
#endif

```

```

#if !defined(OS_WIN)
    void SetFace(RetainPtr<CFX_Face> face);
    void SetFontSpan(pdfium::span<uint8_t> pSpan) { m_FontData = pSpan; }
    void SetSubstFont(std::unique_ptr<CFX_SubstFont> subst);
#endif // !defined(OS_WIN)
#endif // defined(PDF_ENABLE_XFA)

    const CFX_GlyphBitmap* LoadGlyphBitmap(uint32_t glyph_index,
                                           bool bFontStyle,
                                           const CFX_Matrix& matrix,
                                           uint32_t dest_width,
                                           int anti_alias,
                                           int* pTextFlags) const;
    const CFX_PathData* LoadGlyphPath(uint32_t glyph_index,
                                       uint32_t dest_width) const;

#if defined _SKIA_SUPPORT_ || defined _SKIA_SUPPORT_PATHS_
    CFX_TypeFace* GetDeviceCache() const;
#endif

    uint32_t GetGlyphWidth(uint32_t glyph_index);
    int GetAscent() const;
    int GetDescent() const;
    bool GetGlyphBBox(uint32_t glyph_index, FX_RECT* pBBox);
    bool IsItalic() const;
    bool IsBold() const;
    bool IsFixedWidth() const;
    bool IsVertical() const { return m_bVertical; }
    ByteString GetPsName() const;
    ByteString GetFamilyName() const;
    ByteString GetFaceName() const;
    ByteString GetBaseFontName(bool restrict_to_psname) const;
    bool IsTTFont() const;
    bool GetBBox(FX_RECT* pBBox);
    bool IsEmbedded() const { return m_bEmbedded; }
    uint8_t* GetSubData() const { return m_pGsubData.get(); }
    void SetSubData(uint8_t* data) { m_pGsubData.reset(data); }
    pdfium::span<uint8_t> GetFontSpan() const { return m_FontData; }
    void AdjustMMPParams(int glyph_index, int dest_width, int weight) const;
    CFX_PathData* LoadGlyphPathImpl(uint32_t glyph_index,
                                      uint32_t dest_width) const;
#if defined(OS_MACOSX)
    void* GetPlatformFont() const { return m_pPlatformFont; }
    void SetPlatformFont(void* font) { m_pPlatformFont = font; }
#endif

    static const size_t kAngleSkewArraySize = 30;
    static const char s_AngleSkew[kAngleSkewArraySize];
    static const size_t kWeightPowArraySize = 100;
    static const uint8_t s_WeightPow[kWeightPowArraySize];
    static const uint8_t s_WeightPow_11[kWeightPowArraySize];
    static const uint8_t s_WeightPow_SHIFTJIS[kWeightPowArraySize];

    // This struct should be the same as FPDF_CharsetFontMap.
    struct CharsetFontMap {
        int charset; // Character Set Enum value, see FX_CHARSET_XXX.
        const char* fontname; // Name of default font to use with that charset.
    };

    /**
     * Pointer to the default character set to TT Font name map. The
     * map is an array of CharsetFontMap structs, with its end indicated
     * by a { -1, NULL } entry.

```

```
    **/  
    static const CharSetFontMap defaultTTFMap[];  
  
private:  
    RetainPtr<CFX_GlyphCache> GetOrCreateGlyphCache() const;  
    void ClearGlyphCache();  
#if defined(OS_MACOSX)  
    void ReleasePlatformResource();  
#endif  
    ByteString GetFamilyNameOrUntitled() const;  
  
#if defined(PDF_ENABLE_XFA)  
    std::unique_ptr<FXFT_StreamRec> m_pOwnedStream; // Must outlive |m_Face|.   
#endif  
    mutable RetainPtr<CFX_Face> m_Face;  
    mutable RetainPtr<CFX_GlyphCache> m_GlyphCache;  
    std::unique_ptr<CFX_SubstFont> m_pSubstFont;  
    std::unique_ptr<uint8_t, FxFreeDeleter> m_pGsubData;  
    std::vector<uint8_t, FxAllocAllocator<uint8_t>> m_FontDataAllocation;  
    pdfium::span<uint8_t> m_FontData;  
    bool m_bEmbedded = false;  
    bool m_bVertical = false;  
#if defined(OS_MACOSX)  
    void* m_pPlatformFont = nullptr;  
#endif  
};  
  
#endif // CORE_FXGE_CFX_FONT_H_
```



```
bool IsBuiltinFace(const RetainPtr<CFX_Face>& face) const;
int GetFaceSize() const;
ByteString GetFaceName(int index) const { return m_FaceArray[index].name; }

std::vector<ByteString> m_InstalledTTFonts;
std::vector<std::pair<ByteString, ByteString>> m_LocalizedTTFonts;

private:
static const size_t MM_FACE_COUNT = 2;
static const size_t FOXIT_FACE_COUNT = 14;

uint32_t GetChecksumFromTT(void* hFont);
ByteString GetPSNameFromTT(void* hFont);
ByteString MatchInstalledFonts(const ByteString& norm_name);
RetainPtr<CFX_Face> UseInternalSubst(CFX_SubstFont* pSubstFont,
                                   int iBaseFont,
                                   int italic_angle,
                                   int weight,
                                   int pitch_family);
RetainPtr<CFX_Face> GetCachedTTCFace(void* hFont,
                                   uint32_t ttc_size,
                                   uint32_t font_size);
RetainPtr<CFX_Face> GetCachedFace(void* hFont,
                                 ByteString SubstName,
                                 int weight,
                                 bool bItalic,
                                 uint32_t font_size);

struct FaceData {
    ByteString name;
    uint32_t charset;
};

bool m_bListLoaded = false;
ByteString m_LastFamily;
std::vector<FaceData> m_FaceArray;
std::unique_ptr<SystemFontInfoIface> m_pFontInfo;
UnownedPtr<CFX_FontMgr> const m_pFontMgr;
RetainPtr<CFX_Face> m_MMFaces[MM_FACE_COUNT];
RetainPtr<CFX_Face> m_FoxitFaces[FOXIT_FACE_COUNT];
};

#endif // CORE_FXGE_CFX_FONTMAPPER_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_FONTMGR_H_
#define CORE_FXGE_CFX_FONTMGR_H_

#include <map>
#include <memory>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/fx_freetype.h"
#include "third_party/base/optional.h"
#include "third_party/base/span.h"

class CFX_Face;
class CFX_FontMapper;
class CFX_SubstFont;
class SystemFontInfoIface;

class CFX_FontMgr {
public:
  class FontDesc final : public Retainable, public Observable {
public:
  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  ~FontDesc() override;

  pdfium::span<uint8_t> FontData() const {
    return {m_pFontData.get(), m_Size};
  }
  void SetFace(size_t index, CFX_Face* face);
  CFX_Face* GetFace(size_t index) const;

private:
  FontDesc(std::unique_ptr<uint8_t, FxFreeDeleter> pData, size_t size);

  const size_t m_Size;
  std::unique_ptr<uint8_t, FxFreeDeleter> const m_pFontData;
  ObservedPtr<CFX_Face> m_TTCFaces[16];
};

static Optional<pdfium::span<const uint8_t>> GetBuiltinFont(size_t index);

CFX_FontMgr();
~CFX_FontMgr();

RetainPtr<FontDesc> GetCachedFontDesc(const ByteString& face_name,
                                     int weight,
                                     bool bItalic);

RetainPtr<FontDesc> AddCachedFontDesc(
  const ByteString& face_name,
  int weight,
  bool bItalic,
  std::unique_ptr<uint8_t, FxFreeDeleter> pData,
  uint32_t size);
```

```
RetainPtr<FontDesc> GetCachedTTCTFontDesc(int ttc_size, uint32_t checksum);
RetainPtr<FontDesc> AddCachedTTCTFontDesc(
    int ttc_size,
    uint32_t checksum,
    std::unique_ptr<uint8_t, FxFreeDeleter> pData,
    uint32_t size);

RetainPtr<CFX_Face> NewFixedFace(const RetainPtr<FontDesc>& pDesc,
                               pdfium::span<const uint8_t> span,
                               int face_index);
RetainPtr<CFX_Face> FindSubstFont(const ByteString& face_name,
                                 bool bTrueType,
                                 uint32_t flags,
                                 int weight,
                                 int italic_angle,
                                 int CharsetCP,
                                 CFX_SubstFont* pSubstFont);

void SetSystemFontInfo(std::unique_ptr<SystemFontInfoIface> pFontInfo);

// Always present.
CFX_FontMapper* GetBuiltinMapper() const { return m_pBuiltinMapper.get(); }

FXFT_LibraryRec* GetFTLibrary() const { return m_FTLibrary.get(); }
bool FTLibrarySupportsHinting() const { return m_FTLibrarySupportsHinting; }

private:
bool FreeTypeVersionSupportsHinting() const;
bool SetLcdFilterMode() const;

// Must come before |m_pBuiltinMapper| and |m_FaceMap|.
ScopedFXFTLibraryRec const m_FTLibrary;
std::unique_ptr<CFX_FontMapper> m_pBuiltinMapper;
std::map<ByteString, ObservedPtr<FontDesc>> m_FaceMap;
const bool m_FTLibrarySupportsHinting;
};

#endif // CORE_FXGE_CFX_FONTMGR_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_GEMODULE_H_
#define CORE_FXGE_CFX_GEMODULE_H_

#include <memory>

class CFX_FontCache;
class CFX_FontMgr;

class CFX_GEModule {
public:
  class PlatformIface {
public:
    static std::unique_ptr<PlatformIface> Create();
    virtual ~PlatformIface() {}

    virtual void Init() = 0;
  };

  static void Create(const char** pUserFontPaths);
  static void Destroy();
  static CFX_GEModule* Get();

  CFX_FontCache* GetFontCache() const { return m_pFontCache.get(); }
  CFX_FontMgr* GetFontMgr() const { return m_pFontMgr.get(); }
  PlatformIface* GetPlatform() const { return m_pPlatform.get(); }
  const char** GetUserFontPaths() const { return m_pUserFontPaths; }

private:
  explicit CFX_GEModule(const char** pUserFontPaths);
  ~CFX_GEModule();

  std::unique_ptr<PlatformIface> const m_pPlatform;
  std::unique_ptr<CFX_FontMgr> const m_pFontMgr;
  std::unique_ptr<CFX_FontCache> const m_pFontCache;
  const char** const m_pUserFontPaths;
};

#endif // CORE_FXGE_CFX_GEMODULE_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_GLYPHBITMAP_H_
#define CORE_FXGE_CFX_GLYPHBITMAP_H_

#include <vector>

#include "core/fxcrt/retain_ptr.h"

class CFX_DIBitmap;

class CFX_GlyphBitmap {
public:
  CFX_GlyphBitmap(int left, int top);
  ~CFX_GlyphBitmap();

  CFX_GlyphBitmap(const CFX_GlyphBitmap&) = delete;
  CFX_GlyphBitmap& operator=(const CFX_GlyphBitmap&) = delete;

  const RetainPtr<CFX_DIBitmap>& GetBitmap() const { return m_pBitmap; }
  int left() const { return m_Left; }
  int top() const { return m_Top; }

private:
  const int m_Left;
  const int m_Top;
  RetainPtr<CFX_DIBitmap> m_pBitmap;
};

#endif // CORE_FXGE_CFX_GLYPHBITMAP_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_GLYPHCACHE_H
#define CORE_FXGE_CFX_GLYPHCACHE_H

#include <map>
#include <memory>
#include <tuple>

#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/observed_ptr.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/cfx_face.h"

#if defined _SKIA_SUPPORT_ || _SKIA_SUPPORT_PATHS_
#include "core/fxge/fx_font.h"
#include "third_party/skia/include/core/SkTypeface.h"
#endif

class CFX_Font;
class CFX_GlyphBitmap;
class CFX_Matrix;
class CFX_PathData;

class CFX_GlyphCache : public Retainable, public Observable {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    ~CFX_GlyphCache() override;

    const CFX_GlyphBitmap* LoadGlyphBitmap(const CFX_Font* pFont,
        uint32_t glyph_index,
        bool bFontStyle,
        const CFX_Matrix& matrix,
        uint32_t dest_width,
        int anti_alias,
        int* pTextFlags);

    const CFX_PathData* LoadGlyphPath(const CFX_Font* pFont,
        uint32_t glyph_index,
        uint32_t dest_width);

    RetainPtr<CFX_Face> GetFace() { return m_Face; }
    FXFT_FaceRec* GetFaceRec() { return m_Face ? m_Face->GetRec() : nullptr; }

#if defined _SKIA_SUPPORT_ || _SKIA_SUPPORT_PATHS_
    CFX_TypeFace* GetDeviceCache(const CFX_Font* pFont);
#endif

private:
    explicit CFX_GlyphCache(RetainPtr<CFX_Face> face);

    using SizeGlyphCache = std::map<uint32_t, std::unique_ptr<CFX_GlyphBitmap>>;
    // <glyph_index, width, weight, angle, vertical>
    using PathMapKey = std::tuple<uint32_t, uint32_t, int, int, bool>;

    std::unique_ptr<CFX_GlyphBitmap> RenderGlyph(const CFX_Font* pFont,
        uint32_t glyph_index,
        bool bFontStyle,
```

```
        const CFX_Matrix& matrix,
        uint32_t dest_width,
        int anti_alias);

std::unique_ptr<CFX_GlyphBitmap> RenderGlyph_Nativetext(
    const CFX_Font* pFont,
    uint32_t glyph_index,
    const CFX_Matrix& matrix,
    uint32_t dest_width,
    int anti_alias);

CFX_GlyphBitmap* LookUpGlyphBitmap(const CFX_Font* pFont,
    const CFX_Matrix& matrix,
    const ByteString& FaceGlyphsKey,
    uint32_t glyph_index,
    bool bFontStyle,
    uint32_t dest_width,
    int anti_alias);

void InitPlatform();
void DestroyPlatform();

RetainPtr<CFX_Face> const m_Face;
std::map<ByteString, SizeGlyphCache> m_SizeMap;
std::map<PathMapKey, std::unique_ptr<CFX_PathData>> m_PathMap;
#if defined _SKIA_SUPPORT_ || _SKIA_SUPPORT_PATHS_
    sk_sp<SkTypeface> m_pTypeface;
#endif
};

#endif // CORE_FXGE_CFX_GLYPHCACHE_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_GRAPHSTATEDATA_H
#define CORE_FXGE_CFX_GRAPHSTATEDATA_H

#include <vector>

#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

class CFX_GraphStateData {
public:
    enum LineCap : uint8_t {
        LineCapButt = 0,
        LineCapRound = 1,
        LineCapSquare = 2
    };

    enum LineJoin : uint8_t {
        LineJoinMiter = 0,
        LineJoinRound = 1,
        LineJoinBevel = 2
    };

    CFX_GraphStateData();
    CFX_GraphStateData(const CFX_GraphStateData& src);
    CFX_GraphStateData(CFX_GraphStateData&& src);
    ~CFX_GraphStateData();

    CFX_GraphStateData& operator=(const CFX_GraphStateData& that);
    CFX_GraphStateData& operator=(CFX_GraphStateData&& that);

    LineCap m_LineCap = LineCapButt;
    LineJoin m_LineJoin = LineJoinMiter;
    float m_DashPhase = 0.0f;
    float m_MiterLimit = 10.0f;
    float m_LineWidth = 1.0f;
    std::vector<float> m_DashArray;
};

class CFX_RetainableGraphStateData : public Retainable,
                                     public CFX_GraphStateData {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    RetainPtr<CFX_RetainableGraphStateData> Clone() const;

private:
    CFX_RetainableGraphStateData();
    CFX_RetainableGraphStateData(const CFX_RetainableGraphStateData& src);
    ~CFX_RetainableGraphStateData() override;
};

#endif // CORE_FXGE_CFX_GRAPHSTATEDATA_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_GRAPHSTATE_H_
#define CORE_FXGE_CFX_GRAPHSTATE_H_

#include <vector>

#include "core/fxcrt/shared_copy_on_write.h"
#include "core/fxge/cfx_graphstatedata.h"

class CFX_GraphState {
public:
  CFX_GraphState();
  CFX_GraphState(const CFX_GraphState& that);
  ~CFX_GraphState();

  void Emplace();

  void SetLineDash(std::vector<float> dashes, float phase, float scale);

  float GetLineWidth() const;
  void SetLineWidth(float width);

  CFX_GraphStateData::LineCap GetLineCap() const;
  void SetLineCap(CFX_GraphStateData::LineCap cap);

  CFX_GraphStateData::LineJoin GetLineJoin() const;
  void SetLineJoin(CFX_GraphStateData::LineJoin join);

  float GetMiterLimit() const;
  void SetMiterLimit(float limit);

  // FIXME(tsepez): remove when all GraphStateData usage gone.
  const CFX_GraphStateData* GetObject() const { return m_Ref.GetObject(); }

private:
  SharedCopyOnWrite<CFX_RetainableGraphStateData> m_Ref;
};

#endif // CORE_FXGE_CFX_GRAPHSTATE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_PATHDATA_H_
#define CORE_FXGE_CFX_PATHDATA_H_

#include <vector>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxcrt/retain_ptr.h"

enum class FXPT_TYPE : uint8_t { LineTo, BezierTo, MoveTo };

class FX_PATHPOINT {
public:
    FX_PATHPOINT();
    FX_PATHPOINT(const CFX_PointF& point, FXPT_TYPE type, bool close);
    FX_PATHPOINT(const FX_PATHPOINT& other);
    ~FX_PATHPOINT();

    bool IsTypeAndOpen(FXPT_TYPE type) const {
        return m_Type == type && !m_CloseFigure;
    }

    CFX_PointF m_Point;
    FXPT_TYPE m_Type;
    bool m_CloseFigure;
};

class CFX_PathData {
public:
    CFX_PathData();
    CFX_PathData(const CFX_PathData& src);
    CFX_PathData(CFX_PathData&& src);
    ~CFX_PathData();

    void Clear();

    FXPT_TYPE GetType(int index) const { return m_Points[index].m_Type; }
    bool IsClosingFigure(int index) const {
        return m_Points[index].m_CloseFigure;
    }

    CFX_PointF GetPoint(int index) const { return m_Points[index].m_Point; }
    const std::vector<FX_PATHPOINT>& GetPoints() const { return m_Points; }
    std::vector<FX_PATHPOINT>& GetPoints() { return m_Points; }

    CFX_FloatRect GetBoundingBox() const;
    CFX_FloatRect GetBoundingBox(float line_width, float miter_limit) const;

    void Transform(const CFX_Matrix& matrix);
    bool IsRect() const;
    bool GetZeroAreaPath(const CFX_Matrix* pMatrix,
                        bool bAdjust,
                        CFX_PathData* NewPath,
                        bool* bThin,
                        bool* setIdentity) const;
    bool IsRect(const CFX_Matrix* pMatrix, CFX_FloatRect* rect) const;
};
```

```
void Append(const CFX_PathData* pSrc, const CFX_Matrix* pMatrix);
void AppendFloatRect(const CFX_FloatRect& rect);
void AppendRect(float left, float bottom, float right, float top);
void AppendLine(const CFX_PointF& pt1, const CFX_PointF& pt2);
void AppendPoint(const CFX_PointF& point, FXPT_TYPE type, bool closeFigure);
void ClosePath();

private:
    std::vector<FX_PATHPOINT> m_Points;
};

class CFX_RetainablePathData final : public Retainable, public CFX_PathData {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    RetainPtr<CFX_RetainablePathData> Clone() const;

private:
    CFX_RetainablePathData();
    CFX_RetainablePathData(const CFX_RetainablePathData& src);
    ~CFX_RetainablePathData() override;
};

#endif // CORE_FXGE_CFX_PATHDATA_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_RENDERDEVICE_H_
#define CORE_FXGE_CFX_RENDERDEVICE_H_

#include <memory>
#include <vector>

#include "build/build_config.h"
#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/fx_dib.h"
#include "core/fxge/render_defines.h"
#include "core/fxge/renderdevicedriver_iface.h"

class CFX_DIBBase;
class CFX_DIBitmap;
class CFX_Font;
class CFX_GraphStateData;
class CFX_ImageRenderer;
class CFX_PathData;
class PauseIndicatorIface;
class TextCharPos;
struct CFX_Color;

enum class BorderStyle { SOLID, DASH, BEVELED, INSET, UNDERLINE };

class CFX_RenderDevice {
public:
    class StateRestorer {
    public:
        explicit StateRestorer(CFX_RenderDevice* pDevice);
        ~StateRestorer();

    private:
        UnownedPtr<CFX_RenderDevice> m_pDevice;
    };

    CFX_RenderDevice();
    virtual ~CFX_RenderDevice();

    static CFX_Matrix GetFlipMatrix(float width,
                                    float height,
                                    float left,
                                    float top);

    void SetDeviceDriver(std::unique_ptr<RenderDeviceDriverIface> pDriver);
    RenderDeviceDriverIface* GetDeviceDriver() const {
        return m_pDeviceDriver.get();
    }

    void SaveState();
    void RestoreState(bool bKeepSaved);

    int GetWidth() const { return m_Width; }
    int GetHeight() const { return m_Height; }
    DeviceType GetDeviceType() const { return m_DeviceType; }
    int GetRenderCaps() const { return m_RenderCaps; }
    int GetDeviceCaps(int id) const;
```

```
RetainPtr<CFX_DIBitmap> GetBitmap() const;
void SetBitmap(const RetainPtr<CFX_DIBitmap>& pBitmap);
bool CreateCompatibleBitmap(const RetainPtr<CFX_DIBitmap>& pDIB,
                           int width,
                           int height) const;
const FX_RECT& GetClipBox() const { return m_ClipBox; }
void SetBaseClip(const FX_RECT& rect);
bool SetClip_PathFill(const CFX_PathData* pPathData,
                     const CFX_Matrix* pObject2Device,
                     int fill_mode);
bool SetClip_PathStroke(const CFX_PathData* pPathData,
                       const CFX_Matrix* pObject2Device,
                       const CFX_GraphStateData* pGraphState);
bool SetClip_Rect(const FX_RECT& pRect);
bool DrawPath(const CFX_PathData* pPathData,
             const CFX_Matrix* pObject2Device,
             const CFX_GraphStateData* pGraphState,
             uint32_t fill_color,
             uint32_t stroke_color,
             int fill_mode) {
    return DrawPathWithBlend(pPathData, pObject2Device, pGraphState, fill_color,
                             stroke_color, fill_mode, BlendMode::kNormal);
}
bool DrawPathWithBlend(const CFX_PathData* pPathData,
                      const CFX_Matrix* pObject2Device,
                      const CFX_GraphStateData* pGraphState,
                      uint32_t fill_color,
                      uint32_t stroke_color,
                      int fill_mode,
                      BlendMode blend_type);
bool FillRect(const FX_RECT& rect, uint32_t color) {
    return FillRectWithBlend(rect, color, BlendMode::kNormal);
}

RetainPtr<CFX_DIBitmap> GetBackDrop();
bool GetDIBits(const RetainPtr<CFX_DIBitmap>& pBitmap, int left, int top);
bool SetDIBits(const RetainPtr<CFX_DIBase>& pBitmap, int left, int top) {
    return SetDIBitsWithBlend(pBitmap, left, top, BlendMode::kNormal);
}
bool SetDIBitsWithBlend(const RetainPtr<CFX_DIBase>& pBitmap,
                       int left,
                       int top,
                       BlendMode blend_mode);
bool StretchDIBits(const RetainPtr<CFX_DIBase>& pBitmap,
                  int left,
                  int top,
                  int dest_width,
                  int dest_height) {
    return StretchDIBitsWithFlagsAndBlend(pBitmap, left, top, dest_width,
                                           dest_height, FXDIB_ResampleOptions(),
                                           BlendMode::kNormal);
}
bool StretchDIBitsWithFlagsAndBlend(const RetainPtr<CFX_DIBase>& pBitmap,
                                   int left,
                                   int top,
                                   int dest_width,
                                   int dest_height,
                                   const FXDIB_ResampleOptions& options,
                                   BlendMode blend_mode);
bool SetBitMask(const RetainPtr<CFX_DIBase>& pBitmap,
               int left,
               int top,
               uint32_t argb);
```

```
bool StretchBitMask(const RetainPtr<CFX_DIBBase>& pBitmap,
                   int left,
                   int top,
                   int dest_width,
                   int dest_height,
                   uint32_t color);

bool StretchBitMaskWithFlags(const RetainPtr<CFX_DIBBase>& pBitmap,
                              int left,
                              int top,
                              int dest_width,
                              int dest_height,
                              uint32_t argb,
                              const FXDIB_ResampleOptions& options);

bool StartDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
                 int bitmap_alpha,
                 uint32_t color,
                 const CFX_Matrix& matrix,
                 const FXDIB_ResampleOptions& options,
                 std::unique_ptr<CFX_ImageRenderer>* handle) {
    return StartDIBitsWithBlend(pBitmap, bitmap_alpha, color, matrix, options,
                                handle, BlendMode::kNormal);
}

bool StartDIBitsWithBlend(const RetainPtr<CFX_DIBBase>& pBitmap,
                          int bitmap_alpha,
                          uint32_t argb,
                          const CFX_Matrix& matrix,
                          const FXDIB_ResampleOptions& options,
                          std::unique_ptr<CFX_ImageRenderer>* handle,
                          BlendMode blend_mode);

bool ContinueDIBits(CFX_ImageRenderer* handle, PauseIndicatorIface* pPause);

bool DrawNormalText(int nChars,
                   const TextCharPos* pCharPos,
                   CFX_Font* pFont,
                   float font_size,
                   const CFX_Matrix& mtText2Device,
                   uint32_t fill_color,
                   uint32_t text_flags);

bool DrawTextPath(int nChars,
                  const TextCharPos* pCharPos,
                  CFX_Font* pFont,
                  float font_size,
                  const CFX_Matrix& mtText2User,
                  const CFX_Matrix* pUser2Device,
                  const CFX_GraphStateData* pGraphState,
                  uint32_t fill_color,
                  uint32_t stroke_color,
                  CFX_PathData* pClippingPath,
                  int nFlag);

void DrawFillRect(const CFX_Matrix* pUser2Device,
                 const CFX_FloatRect& rect,
                 const CFX_Color& color,
                 int32_t nTransparency);

void DrawFillRect(const CFX_Matrix* pUser2Device,
                 const CFX_FloatRect& rect,
                 const FX_COLORREF& color);

void DrawStrokeRect(const CFX_Matrix& mtUser2Device,
                   const CFX_FloatRect& rect,
                   const FX_COLORREF& color,
                   float fWidth);

void DrawStrokeLine(const CFX_Matrix* pUser2Device,
                   const CFX_PointF& ptMoveTo,
```

```
        const CFX_PointF& ptLineTo,
        const FX_COLORREF& color,
        float fWidth);
void DrawBorder(const CFX_Matrix* pUser2Device,
               const CFX_FloatRect& rect,
               float fWidth,
               const CFX_Color& color,
               const CFX_Color& crLeftTop,
               const CFX_Color& crRightBottom,
               BorderStyle nStyle,
               int32_t nTransparency);
void DrawFillArea(const CFX_Matrix& mtUser2Device,
                 const std::vector<CFX_PointF>& points,
                 const FX_COLORREF& color);
void DrawShadow(const CFX_Matrix& mtUser2Device,
               bool bVertical,
               bool bHorizontal,
               const CFX_FloatRect& rect,
               int32_t nTransparency,
               int32_t nStartGray,
               int32_t nEndGray);

#ifdef _SKIA_SUPPORT_
    virtual void DebugVerifyBitmapIsPreMultiplied() const;
    virtual bool SetBitsWithMask(const RetainPtr<CFX_DIBBase>& pBitmap,
                                const RetainPtr<CFX_DIBBase>& pMask,
                                int left,
                                int top,
                                int bitmap_alpha,
                                BlendMode blend_type);
#endif
#ifdef defined _SKIA_SUPPORT_ || defined _SKIA_SUPPORT_PATHS_
    void Flush(bool release);
#endif

private:
    void InitDeviceInfo();
    void UpdateClipBox();
    bool DrawFillStrokePath(const CFX_PathData* pPathData,
                          const CFX_Matrix* pObject2Device,
                          const CFX_GraphStateData* pGraphState,
                          uint32_t fill_color,
                          uint32_t stroke_color,
                          int fill_mode,
                          BlendMode blend_type);
    bool DrawCosmeticLine(const CFX_PointF& ptMoveTo,
                        const CFX_PointF& ptLineTo,
                        uint32_t color,
                        int fill_mode,
                        BlendMode blend_type);
    bool FillRectWithBlend(const FX_RECT& rect,
                        uint32_t color,
                        BlendMode blend_type);

    RetainPtr<CFX_DIBitmap> m_pBitmap;
    int m_Width = 0;
    int m_Height = 0;
    int m_bpp = 0;
    int m_RenderCaps = 0;
    DeviceType m_DeviceType = DeviceType::kUnknown;
    FX_RECT m_ClipBox;
    std::unique_ptr<RenderDeviceDriverIface> m_pDeviceDriver;
};
```

```
#endif // CORE_FXGE_CFX_RENDERDEVICE_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_SUBSTFONT_H_
#define CORE_FXGE_CFX_SUBSTFONT_H_

#include "core/fxcrt/fx_codepage.h"
#include "core/fxcrt/fx_string.h"

class CFX_SubstFont {
public:
  CFX_SubstFont();
  ~CFX_SubstFont();

  ByteString m_Family;
  int m_Charset = FX_CHARSET_ANSI;
  int m_Weight = 0;
  int m_ItalicAngle = 0;
  int m_WeightCJK = 0;
  bool m_bSubstCJK = false;
  bool m_bItalicCJK = false;
  bool m_bFlagMM = false;
};

#endif // CORE_FXGE_CFX_SUBSTFONT_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_UNICODEENCODINGEX_H_
#define CORE_FXGE_CFX_UNICODEENCODINGEX_H_

#include <memory>

#include "core/fxcrt/fx_system.h"
#include "core/fxge/cfx_unicodeencoding.h"

class CFX_UnicodeEncodingEx final : public CFX_UnicodeEncoding {
public:
    static constexpr uint32_t kInvalidCharCode = static_cast<uint32_t>(-1);

    CFX_UnicodeEncodingEx(CFX_Font* pFont, uint32_t EncodingID);
    ~CFX_UnicodeEncodingEx() override;

    // CFX_UnicodeEncoding:
    uint32_t GlyphFromCharCode(uint32_t charcode) override;

    // Returns |kInvalidCharCode| on error.
    uint32_t CharCodeFromUnicode(wchar_t Unicode) const;

private:
    uint32_t m_nEncodingID;
};

std::unique_ptr<CFX_UnicodeEncodingEx> FX_CreateFontEncodingEx(CFX_Font* pFont);

#endif // CORE_FXGE_CFX_UNICODEENCODINGEX_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_UNICODEENCODING_H_
#define CORE_FXGE_CFX_UNICODEENCODING_H_

#include <stdint.h>

#include "core/fxcrt/unowned_ptr.h"

class CFX_Font;

class CFX_UnicodeEncoding {
 public:
  explicit CFX_UnicodeEncoding(CFX_Font* pFont);
  virtual ~CFX_UnicodeEncoding();

  virtual uint32_t GlyphFromCharCode(uint32_t charcode);

 protected:
  UnownedPtr<CFX_Font> const m_pFont;
};

#endif // CORE_FXGE_CFX_UNICODEENCODING_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_CFX_WINDOWSRENDERDEVICE_H_
#define CORE_FXGE_CFX_WINDOWSRENDERDEVICE_H_

#include <windows.h>

#include "core/fxge/cfx_renderdevice.h"

enum WindowsPrintMode {
  kModeEmf = 0,
  kModeTextOnly = 1,
  kModePostScript2 = 2,
  kModePostScript3 = 3,
  kModePostScript2PassThrough = 4,
  kModePostScript3PassThrough = 5,
};

class RenderDeviceDriverIface;
struct EncoderIface;

#ifdef PDFIUM_PRINT_TEXT_WITH_GDI
typedef void (*PDFiumEnsureTypefaceCharactersAccessible)(const LOGFONT* font,
                                                         const wchar_t* text,
                                                         size_t text_length);

extern bool g_pdfium_print_text_with_gdi;
extern PDFiumEnsureTypefaceCharactersAccessible
  g_pdfium_typeface_accessible_func;
#endif
extern WindowsPrintMode g_pdfium_print_mode;

class CFX_WindowsRenderDevice : public CFX_RenderDevice {
 public:
  CFX_WindowsRenderDevice(HDC hDC, const EncoderIface* pEncoderIface);
  ~CFX_WindowsRenderDevice() override;

 private:
  static RenderDeviceDriverIface* CreateDriver(
    HDC hDC,
    const EncoderIface* pEncoderIface);
};

#endif // CORE_FXGE_CFX_WINDOWSRENDERDEVICE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_CFX_BITMAPCOMPOSER_H_
#define CORE_FXGE_DIB_CFX_BITMAPCOMPOSER_H_

#include <vector>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/dib/cfx_scanlinecomposer.h"
#include "core/fxge/dib/scanlinecomposer_iface.h"
#include "core/fxge/fx_dib.h"

class CFX_ClipRgn;
class CFX_DIBitmap;

class CFX_BitmapComposer final : public ScanlineComposerIface {
public:
    CFX_BitmapComposer();
    ~CFX_BitmapComposer() override;

    void Compose(const RetainPtr<CFX_DIBitmap>& pDest,
                 const CFX_ClipRgn* pClipRgn,
                 int bitmap_alpha,
                 uint32_t mask_color,
                 const FX_RECT& dest_rect,
                 bool bVertical,
                 bool bFlipX,
                 bool bFlipY,
                 bool bRgbByteOrder,
                 BlendMode blend_type);

    // ScanlineComposerIface
    bool SetInfo(int width,
                 int height,
                 FXDIB_Format src_format,
                 uint32_t* pSrcPalette) override;

    void ComposeScanline(int line,
                          const uint8_t* scanline,
                          const uint8_t* scan_extra_alpha) override;

private:
    void DoCompose(uint8_t* dest_scan,
                  const uint8_t* src_scan,
                  int dest_width,
                  const uint8_t* clip_scan,
                  const uint8_t* src_extra_alpha,
                  uint8_t* dst_extra_alpha);
    void ComposeScanlineV(int line,
                           const uint8_t* scanline,
                           const uint8_t* scan_extra_alpha);

    RetainPtr<CFX_DIBitmap> m_pBitmap;
    UnownedPtr<const CFX_ClipRgn> m_pClipRgn;
    FXDIB_Format m_SrcFormat;
    int m_DestLeft;
    int m_DestTop;
};
```

```
int m_DestWidth;
int m_DestHeight;
int m_BitmapAlpha;
uint32_t m_MaskColor;
RetainPtr<CFX_DIBitmap> m_pClipMask;
CFX_ScanlineCompositor m_Compositor;
bool m_bVertical;
bool m_bFlipX;
bool m_bFlipY;
bool m_bRgbByteOrder = false;
BlendMode m_BlendType = BlendMode::kNormal;
std::vector<uint8_t> m_pScanlineV;
std::vector<uint8_t> m_pClipScanV;
std::vector<uint8_t> m_pAddClipScan;
std::vector<uint8_t> m_pScanlineAlphaV;
};

#endif // CORE_FXGE_DIB_CFX_BITMAPCOMPOSER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifdef CORE_FXGE_DIB_CFX_BITMAPSTORER_H
#define CORE_FXGE_DIB_CFX_BITMAPSTORER_H

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/dib/scanlinecomposer_iface.h"

class CFX_DIBitmap;

class CFX_BitmapStorer final : public ScanlineComposerIface {
public:
    CFX_BitmapStorer();
    ~CFX_BitmapStorer() override;

    // ScanlineComposerIface
    void ComposeScanline(int line,
                        const uint8_t* scanline,
                        const uint8_t* scan_extra_alpha) override;
    bool SetInfo(int width,
                int height,
                FXDIB_Format src_format,
                uint32_t* pSrcPalette) override;

    RetainPtr<CFX_DIBitmap> GetBitmap() { return m_pBitmap; }
    RetainPtr<CFX_DIBitmap> Detach();
    void Replace(RetainPtr<CFX_DIBitmap>&& pBitmap);

private:
    RetainPtr<CFX_DIBitmap> m_pBitmap;
};

#endif // CORE_FXGE_DIB_CFX_BITMAPSTORER_H
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_CFX_CMYK_TO_SRGB_H_
#define CORE_FXGE_DIB_CFX_CMYK_TO_SRGB_H_

#include <stdint.h>

#include <tuple>

namespace fxge {

std::tuple<float, float, float> AdobeCMYK_to_sRGB(float c,
                                                float m,
                                                float y,
                                                float k);

std::tuple<uint8_t, uint8_t, uint8_t> AdobeCMYK_to_sRGB1(uint8_t c,
                                                         uint8_t m,
                                                         uint8_t y,
                                                         uint8_t k);

} // namespace fxge

using fxge::AdobeCMYK_to_sRGB;
using fxge::AdobeCMYK_to_sRGB1;

#endif // CORE_FXGE_DIB_CFX_CMYK_TO_SRGB_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_CFX_DIBBASE_H_
#define CORE_FXGE_DIB_CFX_DIBBASE_H_

#include <memory>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/fx_dib.h"

enum FXDIB_Channel {
  FXDIB_Red = 1,
  FXDIB_Green,
  FXDIB_Blue,
  FXDIB_Cyan,
  FXDIB_Magenta,
  FXDIB_Yellow,
  FXDIB_Black,
  FXDIB_Alpha
};

class CFX_ClipRgn;
class CFX_DIBitmap;
class PauseIndicatorIface;

// Base class for all Device-Independent Bitmaps.
class CFX_DIBBase : public Retainable {
public:
  ~CFX_DIBBase() override;

  virtual uint8_t* GetBuffer() const;
  virtual const uint8_t* GetScanline(int line) const = 0;
  virtual bool SkipToScanline(int line, PauseIndicatorIface* pPause) const;
  virtual void DownSampleScanline(int line,
                                  uint8_t* dest_scan,
                                  int dest_bpp,
                                  int dest_width,
                                  bool bFlipX,
                                  int clip_left,
                                  int clip_width) const = 0;

  uint8_t* GetWritableScanline(int line) {
    return const_cast<uint8_t*>(GetScanline(line));
  }
  int GetWidth() const { return m_Width; }
  int GetHeight() const { return m_Height; }

  FXDIB_Format GetFormat() const {
    return static_cast<FXDIB_Format>(m_AlphaFlag * 0x100 + m_bpp);
  }
  uint32_t GetPitch() const { return m_Pitch; }
  uint32_t* GetPalette() const { return m_pPalette.get(); }
  int GetBPP() const { return m_bpp; }

  // TODO(thestig): Investigate this. Given the possible values of FXDIB_Format,
  // it feels as though this should be implemented as !(m_AlphaFlag & 1) and
  // IsOpaqueImage() below should never be able to return true.

```

```
bool IsAlphaMask() const { return m_AlphaFlag == 1; }
bool HasAlpha() const { return !(m_AlphaFlag & 2); }
bool IsOpaqueImage() const { return !(m_AlphaFlag & 3); }
bool IsCmykImage() const { return !(m_AlphaFlag & 4); }

int GetPaletteSize() const {
    return IsAlphaMask() ? 0 : (m_bpp == 1 ? 2 : (m_bpp == 8 ? 256 : 0));
}

uint32_t GetPaletteArgb(int index) const;
void SetPaletteArgb(int index, uint32_t color);

// Copies into internally-owned palette.
void SetPalette(const uint32_t* pSrcPal);

RetainPtr<CFX_DIBitmap> Clone(const FX_RECT* pClip) const;
RetainPtr<CFX_DIBitmap> CloneConvert(FXDIB_Format format);
RetainPtr<CFX_DIBitmap> StretchTo(int dest_width,
                                int dest_height,
                                const FXDIB_ResampleOptions& options,
                                const FX_RECT* pClip);
RetainPtr<CFX_DIBitmap> TransformTo(const CFX_Matrix& mtDest,
                                   int* left,
                                   int* top);
RetainPtr<CFX_DIBitmap> SwapXY(bool bXFlip, bool bYFlip) const;
RetainPtr<CFX_DIBitmap> FlipImage(bool bXFlip, bool bYFlip) const;

RetainPtr<CFX_DIBitmap> CloneAlphaMask() const;

// Copies into internally-owned mask.
bool SetAlphaMask(const RetainPtr<CFX_DIBBase>& pAlphaMask,
                  const FX_RECT* pClip);

bool GetOverlapRect(int& dest_left,
                    int& dest_top,
                    int& width,
                    int& height,
                    int src_width,
                    int src_height,
                    int& src_left,
                    int& src_top,
                    const CFX_ClipRgn* pClipRgn);

#ifdef _SKIA_SUPPORT_ || defined _SKIA_SUPPORT_PATHS_
void DebugVerifyBitmapIsPreMultiplied(void* buffer) const;
#endif

RetainPtr<CFX_DIBitmap> m_pAlphaMask;

protected:
CFX_DIBBase();

static bool ConvertBuffer(FXDIB_Format dest_format,
                          uint8_t* dest_buf,
                          int dest_pitch,
                          int width,
                          int height,
                          const RetainPtr<CFX_DIBBase>& pSrcBitmap,
                          int src_left,
                          int src_top,
                          std::unique_ptr<uint32_t, FxFreeDeleter>* pal);

void BuildPalette();
```

```
bool BuildAlphaMask();
int FindPalette(uint32_t color) const;
void GetPalette(uint32_t* pal, int alpha) const;

int m_Width;
int m_Height;
int m_bpp;
uint32_t m_AlphaFlag;
uint32_t m_Pitch;
// TODO(weili): Use std::vector for this.
std::unique_ptr<uint32_t, FxFreeDeleter> m_pPalette;
};

#endif // CORE_FXGE_DIB_CFX_DIBBASE_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_CFX_DIBEXTRACTOR_H_
#define CORE_FXGE_DIB_CFX_DIBEXTRACTOR_H_

#include "core/fxcrt/retain_ptr.h"

class CFX_DIBBase;
class CFX_DIBitmap;

class CFX_DIBExtractor {
public:
  explicit CFX_DIBExtractor(const RetainPtr<CFX_DIBBase>& pSrc);
  ~CFX_DIBExtractor();

  RetainPtr<CFX_DIBitmap> GetBitmap() { return m_pBitmap; }

private:
  RetainPtr<CFX_DIBitmap> m_pBitmap;
};

#endif // CORE_FXGE_DIB_CFX_DIBEXTRACTOR_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_CFX_DIBITMAP_H
#define CORE_FXGE_DIB_CFX_DIBITMAP_H

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/maybe_owned.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/dib/cfx_dibbase.h"
#include "core/fxge/fx_dib.h"

class CFX_DIBitmap : public CFX_DIBBase {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    bool Create(int width, int height, FXDIB_Format format);

    bool Create(int width,
                int height,
                FXDIB_Format format,
                uint8_t* pBuffer,
                uint32_t pitch);

    bool Copy(const RetainPtr<CFX_DIBBase>& pSrc);

    // CFX_DIBBase
    uint8_t* GetBuffer() const override;
    const uint8_t* GetScanline(int line) const override;
    void DownSampleScanline(int line,
                             uint8_t* dest_scan,
                             int dest_bpp,
                             int dest_width,
                             bool bFlipX,
                             int clip_left,
                             int clip_width) const override;

    void TakeOver(RetainPtr<CFX_DIBitmap>&& pSrcBitmap);
    bool ConvertFormat(FXDIB_Format format);
    void Clear(uint32_t color);

    uint32_t GetPixel(int x, int y) const;
    void SetPixel(int x, int y, uint32_t color);

    bool LoadChannelFromAlpha(FXDIB_Channel destChannel,
                              const RetainPtr<CFX_DIBBase>& pSrcBitmap);
    bool LoadChannel(FXDIB_Channel destChannel, int value);

    bool MultiplyAlpha(int alpha);
    bool MultiplyAlpha(const RetainPtr<CFX_DIBBase>& pSrcBitmap);

    bool TransferBitmap(int dest_left,
                        int dest_top,
                        int width,
                        int height,
                        const RetainPtr<CFX_DIBBase>& pSrcBitmap,
                        int src_left,
                        int src_top);
};
```

```
bool CompositeBitmap(int dest_left,
                    int dest_top,
                    int width,
                    int height,
                    const RetainPtr<CFX_DIBBase>& pSrcBitmap,
                    int src_left,
                    int src_top,
                    BlendMode blend_type,
                    const CFX_ClipRgn* pClipRgn,
                    bool bRgbByteOrder);

bool CompositeMask(int dest_left,
                  int dest_top,
                  int width,
                  int height,
                  const RetainPtr<CFX_DIBBase>& pMask,
                  uint32_t color,
                  int src_left,
                  int src_top,
                  BlendMode blend_type,
                  const CFX_ClipRgn* pClipRgn,
                  bool bRgbByteOrder);

bool CompositeRect(int dest_left,
                  int dest_top,
                  int width,
                  int height,
                  uint32_t color,
                  int alpha_flag);

bool ConvertColorScale(uint32_t forecolor, uint32_t backcolor);

static bool CalculatePitchAndSize(int height,
                                 int width,
                                 FXDIB_Format format,
                                 uint32_t* pitch,
                                 uint32_t* size);

#if defined _SKIA_SUPPORT_ || _SKIA_SUPPORT_PATHS_
    void PreMultiply();
#endif
#if defined _SKIA_SUPPORT_PATHS_
    void UnPreMultiply();
#endif

protected:
    CFX_DIBitmap();
    CFX_DIBitmap(const CFX_DIBitmap& src);
    ~CFX_DIBitmap() override;

#if defined _SKIA_SUPPORT_PATHS_
    enum class Format { kCleared, kPreMultiplied, kUnPreMultiplied };
#endif

    MaybeOwned<uint8_t, FxFreeDeleter> m_pBuffer;
#if defined _SKIA_SUPPORT_PATHS_
    Format m_nFormat;
#endif

private:
    void ConvertBGRColorScale(uint32_t forecolor, uint32_t backcolor);
    void ConvertCMYKColorScale(uint32_t forecolor, uint32_t backcolor);
```

```
bool TransferWithUnequalFormats(FXDIB_Format dest_format,
                                int dest_left,
                                int dest_top,
                                int width,
                                int height,
                                const RetainPtr<CFX_DIBBase>& pSrcBitmap,
                                int src_left,
                                int src_top);

void TransferWithMultipleBPP(int dest_left,
                              int dest_top,
                              int width,
                              int height,
                              const RetainPtr<CFX_DIBBase>& pSrcBitmap,
                              int src_left,
                              int src_top);

void TransferEqualFormatsOneBPP(int dest_left,
                                 int dest_top,
                                 int width,
                                 int height,
                                 const RetainPtr<CFX_DIBBase>& pSrcBitmap,
                                 int src_left,
                                 int src_top);

};

#endif // CORE_FXGE_DIB_CFX_DIBITMAP_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_CFX_FILTEREDDIB_H_
#define CORE_FXGE_DIB_CFX_FILTEREDDIB_H_

#include <vector>

#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/dib/cfx_dibase.h"

class CFX_FilteredDIB : public CFX_DIBBase {
public:
  template <typename T, typename... Args>
  friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

  virtual FXDIB_Format GetDestFormat() = 0;
  virtual uint32_t* GetDestPalette() = 0;
  virtual void TranslateScanline(const uint8_t* src_buf,
                                std::vector<uint8_t>* dest_buf) const = 0;
  virtual void TranslateDownSamples(uint8_t* dest_buf,
                                    const uint8_t* src_buf,
                                    int pixels,
                                    int Bpp) const = 0;

  void LoadSrc(const RetainPtr<CFX_DIBBase>& pSrc);

protected:
  CFX_FilteredDIB();
  ~CFX_FilteredDIB() override;

  // CFX_DIBBase
  const uint8_t* GetScanline(int line) const override;
  void DownSampleScanline(int line,
                          uint8_t* dest_scan,
                          int dest_bpp,
                          int dest_width,
                          bool bFlipX,
                          int clip_left,
                          int clip_width) const override;

  RetainPtr<CFX_DIBBase> m_pSrc;
  mutable std::vector<uint8_t> m_Scanline;
};

#endif // CORE_FXGE_DIB_CFX_FILTEREDDIB_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_CFX_IMAGERENDERER_H_
#define CORE_FXGE_DIB_CFX_IMAGERENDERER_H_

#include <memory>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/dib/cfx_bitmapcomposer.h"

class CFX_DIBBase;
class CFX_DIBitmap;
class CFX_ImageTransformer;
class CFX_ImageStretcher;
class PauseIndicatorIface;

class CFX_ImageRenderer {
public:
    CFX_ImageRenderer(const RetainPtr<CFX_DIBitmap>& pDevice,
                     const CFX_ClipRgn* pClipRgn,
                     const RetainPtr<CFX_DIBBase>& pSource,
                     int bitmap_alpha,
                     uint32_t mask_color,
                     const CFX_Matrix& matrix,
                     const FXDIB_ResampleOptions& options,
                     bool bRgbByteOrder);
    ~CFX_ImageRenderer();

    bool Continue(PauseIndicatorIface* pPause);

private:
    RetainPtr<CFX_DIBitmap> const m_pDevice;
    UnownedPtr<const CFX_ClipRgn> const m_pClipRgn;
    const CFX_Matrix m_Matrix;
    std::unique_ptr<CFX_ImageTransformer> m_pTransformer;
    std::unique_ptr<CFX_ImageStretcher> m_Stretcher;
    CFX_BitmapComposer m_Composer;
    FX_RECT m_ClipBox;
    const int m_BitmapAlpha;
    int m_Status = 0;
    uint32_t m_MaskColor;
    const bool m_bRgbByteOrder;
};

#endif // CORE_FXGE_DIB_CFX_IMAGERENDERER_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifdef CORE_FXGE_DIB_CFX_IMAGESTRETCHER_H
#define CORE_FXGE_DIB_CFX_IMAGESTRETCHER_H

#include <memory>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/dib/scanlinecomposer_iface.h"
#include "core/fxge/fx_dib.h"

class CFX_DIBBase;
class CStretchEngine;
class PauseIndicatorIface;

class CFX_ImageStretcher {
public:
    CFX_ImageStretcher(ScanlineComposerIface* pDest,
                      const RetainPtr<CFX_DIBBase>& pSource,
                      int dest_width,
                      int dest_height,
                      const FX_RECT& bitmap_rect,
                      const FXDIB_ResampleOptions& options);
    ~CFX_ImageStretcher();

    bool Start();
    bool Continue(PauseIndicatorIface* pPause);

    RetainPtr<CFX_DIBBase> source();

private:
    bool StartQuickStretch();
    bool StartStretch();
    bool ContinueQuickStretch(PauseIndicatorIface* pPause);
    bool ContinueStretch(PauseIndicatorIface* pPause);

    UnownedPtr<ScanlineComposerIface> const m_pDest;
    RetainPtr<CFX_DIBBase> m_pSource;
    std::unique_ptr<CStretchEngine> m_pStretchEngine;
    std::unique_ptr<uint8_t, FxFreeDeleter> m_pScanline;
    std::unique_ptr<uint8_t, FxFreeDeleter> m_pMaskScanline;
    const FXDIB_ResampleOptions m_ResampleOptions;
    bool m_bFlipX;
    bool m_bFlipY;
    int m_DestWidth;
    int m_DestHeight;
    const FX_RECT m_ClipRect;
    const FXDIB_Format m_DestFormat;
    const int m_DestBPP;
    int m_LineIndex;
};

#endif // CORE_FXGE_DIB_CFX_IMAGESTRETCHER_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_CFX_IMAGETRANSFORMER_H_
#define CORE_FXGE_DIB_CFX_IMAGETRANSFORMER_H_

#include <memory>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/dib/cfx_bitmapstorer.h"

class CFX_DIBBase;
class CFX_DIBitmap;
class CFX_ImageStretcher;
class PauseIndicatorIface;

class CFX_ImageTransformer {
public:
    struct BilinearData {
        int res_x;
        int res_y;
        int src_col_l;
        int src_row_l;
        int src_col_r;
        int src_row_r;
        int row_offset_l;
        int row_offset_r;
    };

    struct BicubicData {
        int res_x;
        int res_y;
        int src_col_l;
        int src_row_l;
        int src_col_r;
        int src_row_r;
        int pos_pixel[8];
        int u_w[4];
        int v_w[4];
    };

    struct DownSampleData {
        int src_col;
        int src_row;
    };

    struct CalcData {
        CFX_DIBitmap* bitmap;
        const CFX_Matrix& matrix;
        const uint8_t* buf;
        uint32_t pitch;
    };

    CFX_ImageTransformer(const RetainPtr<CFX_DIBBase>& pSrc,
                        const CFX_Matrix& matrix,
                        const FXDIB_ResampleOptions& options,
                        const FX_RECT* pClip);
    ~CFX_ImageTransformer();
};
```

```
    bool Continue(PauseIndicatorIface* pPause);

    const FX_RECT& result() const { return m_result; }
    RetainPtr<CFX_DIBitmap> DetachBitmap();

private:
    void CalcMask(const CalcData& cdata);
    void CalcAlpha(const CalcData& cdata);
    void CalcMono(const CalcData& cdata, FXDIB_Format format);
    void CalcColor(const CalcData& cdata, FXDIB_Format format, int Bpp);

    bool IsBilinear() const;
    bool IsBiCubic() const;

    RetainPtr<CFX_DIBBase> const m_pSrc;
    const CFX_Matrix m_matrix;
    FX_RECT m_StretchClip;
    FX_RECT m_result;
    CFX_Matrix m_dest2stretch;
    std::unique_ptr<CFX_ImageStretcher> m_Stretcher;
    CFX_BitmapStorer m_Storer;
    const FXDIB_ResampleOptions m_ResampleOptions;
    int m_Status = 0;
};

#endif // CORE_FXGE_DIB_CFX_IMAGETRANSFORMER_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_CFX_SCANLINECOMPOSITOR_H
#define CORE_FXGE_DIB_CFX_SCANLINECOMPOSITOR_H

#include <memory>

#include "core/fxcrt/fx_memory_wrappers.h"
#include "core/fxge/fx_dib.h"

class CFX_ScanlineCompositor {
public:
    CFX_ScanlineCompositor();
    ~CFX_ScanlineCompositor();

    bool Init(FXDIB_Format dest_format,
             FXDIB_Format src_format,
             int32_t width,
             uint32_t* pSrcPalette,
             uint32_t mask_color,
             BlendMode blend_type,
             bool bClip,
             bool bRgbByteOrder);

    void CompositeRgbBitmapLine(uint8_t* dest_scan,
                               const uint8_t* src_scan,
                               int width,
                               const uint8_t* clip_scan,
                               const uint8_t* src_extra_alpha,
                               uint8_t* dst_extra_alpha);

    void CompositePalBitmapLine(uint8_t* dest_scan,
                                const uint8_t* src_scan,
                                int src_left,
                                int width,
                                const uint8_t* clip_scan,
                                const uint8_t* src_extra_alpha,
                                uint8_t* dst_extra_alpha);

    void CompositeByteMaskLine(uint8_t* dest_scan,
                               const uint8_t* src_scan,
                               int width,
                               const uint8_t* clip_scan,
                               uint8_t* dst_extra_alpha);

    void CompositeBitMaskLine(uint8_t* dest_scan,
                              const uint8_t* src_scan,
                              int src_left,
                              int width,
                              const uint8_t* clip_scan,
                              uint8_t* dst_extra_alpha);

private:
    void InitSourcePalette(FXDIB_Format src_format,
                          FXDIB_Format dest_format,
                          const uint32_t* pSrcPalette);

    void InitSourceMask(uint32_t mask_color);
};
```

```
int m_iTransparency;
FXDIB_Format m_SrcFormat;
FXDIB_Format m_DestFormat;
std::unique_ptr<uint32_t, FxFreeDeleter> m_pSrcPalette;
int m_MaskAlpha;
int m_MaskRed;
int m_MaskGreen;
int m_MaskBlue;
BlendMode m_BlendType = BlendMode::kNormal;
bool m_bRgbByteOrder = false;
};

#endif // CORE_FXGE_DIB_CFX_SCANLINECOMPOSITOR_H
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_CSTRETCHENGINE_H
#define CORE_FXGE_DIB_CSTRETCHENGINE_H

#include <vector>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/retain_ptr.h"
#include "core/fxcrt/unowned_ptr.h"
#include "core/fxge/fx_dib.h"

class CFX_DIBBase;
class PauseIndicatorIface;
class ScanlineComposerIface;

class CStretchEngine {
public:
    CStretchEngine(ScanlineComposerIface* pDestBitmap,
                  FXDIB_Format dest_format,
                  int dest_width,
                  int dest_height,
                  const FX_RECT& clip_rect,
                  const RetainPtr<CFX_DIBBase>& pSrcBitmap,
                  const FXDIB_ResampleOptions& options);
    ~CStretchEngine();

    bool Continue(PauseIndicatorIface* pPause);

    bool StartStretchHorz();
    bool ContinueStretchHorz(PauseIndicatorIface* pPause);
    void StretchVert();

class CWeightTable {
public:
    CWeightTable();
    ~CWeightTable();

    bool Calc(int dest_len,
              int dest_min,
              int dest_max,
              int src_len,
              int src_min,
              int src_max,
              const FXDIB_ResampleOptions& options);

    const PixelWeight* GetPixelWeight(int pixel) const;
    PixelWeight* GetPixelWeight(int pixel) {
        return const_cast<PixelWeight*>(
            static_cast<const CWeightTable*>(this)->GetPixelWeight(pixel));
    }

    int* GetValueFromPixelWeight(PixelWeight* pWeight, int index) const;
    size_t GetPixelWeightSize() const;

private:
    int m_DestMin = 0;
    int m_ItemSize = 0;
    size_t m_dwWeightTablesSize = 0;
};
};
```

```
    std::vector<uint8_t> m_WeightTables;
};

enum class State : uint8_t { kInitial, kHorizontal, kVertical };

enum class TransformMethod : uint8_t {
    k1BppTo8Bpp,
    k1BppToManyBpp,
    k8BppTo8Bpp,
    k8BppTo8BppWithAlpha,
    k8BppToManyBpp,
    k8BppToManyBppWithAlpha,
    kManyBpptoManyBpp,
    kManyBpptoManyBppWithAlpha
};

const FXDIB_Format m_DestFormat;
const int m_DestBpp;
const int m_SrcBpp;
const int m_bHasAlpha;
RetainPtr<CFX_DIBBase> const m_pSource;
const uint32_t* m_pSrcPalette;
const int m_SrcWidth;
const int m_SrcHeight;
UnownedPtr<ScanlineComposerIface> const m_pDestBitmap;
const int m_DestWidth;
const int m_DestHeight;
const FX_RECT m_DestClip;
std::vector<uint8_t> m_DestScanline;
std::vector<uint8_t> m_DestMaskScanline;
std::vector<uint8_t> m_InterBuf;
std::vector<uint8_t> m_ExtraAlphaBuf;
FX_RECT m_SrcClip;
int m_InterPitch;
int m_ExtraMaskPitch;
FXDIB_ResampleOptions m_ResampleOptions;
TransformMethod m_TransMethod;
State m_State = State::kInitial;
int m_CurRow;
CWeightTable m_WeightTable;
};

#endif // CORE_FXGE_DIB_CSTRETCHENGINE_H_
```

```
// Copyright 2017 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_DIB_SCANLINECOMPOSER_IFACE_H
#define CORE_FXGE_DIB_SCANLINECOMPOSER_IFACE_H

#include "core/fxge/fx_dib.h"

class ScanlineComposerIface {
public:
    virtual ~ScanlineComposerIface() = default;

    virtual void ComposeScanline(int line,
                                const uint8_t* scanline,
                                const uint8_t* scan_extra_alpha) = 0;

    virtual bool SetInfo(int width,
                        int height,
                        FXDIB_Format src_format,
                        uint32_t* pSrcPalette) = 0;
};

#endif // CORE_FXGE_DIB_SCANLINECOMPOSER_IFACE_H
```

```
// Copyright 2015 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be  
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FXGE_FONTDATA_CHROMEFONTDATA_CHROMEFONTDATA_H
```

```
#define CORE_FXGE_FONTDATA_CHROMEFONTDATA_CHROMEFONTDATA_H
```

```
extern const unsigned char g_FoxitFixedItalicFontData[18746];  
extern const unsigned char g_FoxitFixedFontData[17597];  
extern const unsigned char g_FoxitSansItalicFontData[16339];  
extern const unsigned char g_FoxitSansFontData[15025];  
extern const unsigned char g_FoxitSerifItalicFontData[21227];  
extern const unsigned char g_FoxitSerifFontData[19469];  
extern const unsigned char g_FoxitFixedBoldItalicFontData[19151];  
extern const unsigned char g_FoxitFixedBoldFontData[18055];  
extern const unsigned char g_FoxitSansBoldItalicFontData[16418];  
extern const unsigned char g_FoxitSansBoldFontData[16344];  
extern const unsigned char g_FoxitSerifBoldItalicFontData[20733];  
extern const unsigned char g_FoxitSerifBoldFontData[19395];  
extern const unsigned char g_FoxitSymbolFontData[16729];  
extern const unsigned char g_FoxitDingbatsFontData[29513];  
extern const unsigned char g_FoxitSerifMMFontData[113417];  
extern const unsigned char g_FoxitSansMMFontData[66919];
```

```
#endif // CORE_FXGE_FONTDATA_CHROMEFONTDATA_CHROMEFONTDATA_H
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_FX_DIB_H
#define CORE_FXGE_FX_DIB_H

#include <tuple>
#include <utility>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/widestring.h"

enum FXDIB_Format {
    FXDIB_Invalid = 0,
    FXDIB_1bppRgb = 0x001,
    FXDIB_8bppRgb = 0x008,
    FXDIB_Rgb = 0x018,
    FXDIB_Rgb32 = 0x020,
    FXDIB_1bppMask = 0x101,
    FXDIB_8bppMask = 0x108,
    FXDIB_8bppRgba = 0x208,
    FXDIB_Rgba = 0x218,
    FXDIB_Argb = 0x220,
    FXDIB_1bppCmyk = 0x401,
    FXDIB_8bppCmyk = 0x408,
    FXDIB_Cmyk = 0x420,
    FXDIB_8bppCmyka = 0x608,
    FXDIB_Cmyka = 0x620,
};

struct PixelWeight {
    int m_SrcStart;
    int m_SrcEnd;
    int m_Weights[1];
};

using FX_ARGB = uint32_t;

// FX_COLORREF, like win32 COLORREF, is BGR.
using FX_COLORREF = uint32_t;

using FX_CMYK = uint32_t;

extern const int16_t SDP_Table[513];

struct FXDIB_ResampleOptions {
    FXDIB_ResampleOptions();

    bool HasAnyOptions() const;

    bool bInterpolateDownsample = false;
    bool bInterpolateBilinear = false;
    bool bInterpolateBicubic = false;
    bool bHalftone = false;
    bool bNoSmoothing = false;
    bool bLossy = false;
};

// See PDF 1.7 spec, table 7.2 and 7.3. The enum values need to be in the same
// order as listed in the spec.
```

```
enum class BlendMode {
    kNormal = 0,
    kMultiply,
    kScreen,
    kOverlay,
    kDarken,
    kLighten,
    kColorDodge,
    kColorBurn,
    kHardLight,
    kSoftLight,
    kDifference,
    kExclusion,
    kHue,
    kSaturation,
    kColor,
    kLuminosity,
    kLast = kLuminosity,
};

constexpr uint32_t FXSYS_BGR(uint8_t b, uint8_t g, uint8_t r) {
    return (b << 16) | (g << 8) | r;
}

constexpr uint8_t FXSYS_GetRValue(uint32_t bgr) {
    return bgr & 0xff;
}

constexpr uint8_t FXSYS_GetGValue(uint32_t bgr) {
    return (bgr >> 8) & 0xff;
}

constexpr uint8_t FXSYS_GetBValue(uint32_t bgr) {
    return (bgr >> 16) & 0xff;
}

constexpr unsigned int FXSYS_GetUnsignedAlpha(float alpha) {
    return static_cast<unsigned int>(alpha * 255.f + 0.5f);
}

#define FXSYS_GetCValue(cmyk) ((uint8_t)((cmyk) >> 24) & 0xff)
#define FXSYS_GetMValue(cmyk) ((uint8_t)((cmyk) >> 16) & 0xff)
#define FXSYS_GetYValue(cmyk) ((uint8_t)((cmyk) >> 8) & 0xff)
#define FXSYS_GetKValue(cmyk) ((uint8_t)(cmyk)&0xff)

// Bits per pixel, not bytes.
inline int GetBppFromFormat(FXDIB_Format format) {
    return format & 0xff;
}

// AKA bytes per pixel, assuming 8-bits per component.
inline int GetCompsFromFormat(FXDIB_Format format) {
    return (format & 0xff) / 8;
}

inline bool GetIsAlphaFromFormat(FXDIB_Format format) {
    return format & 0x200;
}

inline bool GetIsCmykFromFormat(FXDIB_Format format) {
    return format & 0x400;
}
```

```

inline FX_CMYK CmykEncode(int c, int m, int y, int k) {
    return (c << 24) | (m << 16) | (y << 8) | k;
}

// Returns (a, r, g, b)
std::tuple<int, int, int, int> ArgbDecode(FX_ARGB argb);

// Returns (a, FX_COLORREF)
std::pair<int, FX_COLORREF> ArgbToAlphaAndColorRef(FX_ARGB argb);

// Returns FX_COLORREF.
FX_COLORREF ArgbToColorRef(FX_ARGB argb);

constexpr FX_ARGB ArgbEncode(int a, int r, int g, int b) {
    return (a << 24) | (r << 16) | (g << 8) | b;
}

FX_ARGB AlphaAndColorRefToArgb(int a, FX_COLORREF colorref);

FX_ARGB StringToFXARGB(WideStringView view);

#define FXARGB_A(argb) ((uint8_t)((argb) >> 24))
#define FXARGB_R(argb) ((uint8_t)((argb) >> 16))
#define FXARGB_G(argb) ((uint8_t)((argb) >> 8))
#define FXARGB_B(argb) ((uint8_t)(argb))
#define FXARGB_MUL_ALPHA(argb, alpha) \
    (((argb) >> 24) * (alpha) / 255) << 24 | ((argb)&0xffffffff)

#define FXRGB2GRAY(r, g, b) ((b)*11 + (g)*59 + (r)*30) / 100
#define FXDIB_ALPHA_MERGE(backdrop, source, source_alpha) \
    ((backdrop) * (255 - (source_alpha)) + (source) * (source_alpha)) / 255)
#define FXARGB_GETDIB(p) \
    (((uint8_t*)(p))[0] | (((uint8_t*)(p))[1] << 8) | \
    ((uint8_t*)(p))[2] << 16) | (((uint8_t*)(p))[3] << 24))
#define FXARGB_SETDIB(p, argb) \
    ((uint8_t*)(p))[0] = (uint8_t)(argb), \
    ((uint8_t*)(p))[1] = (uint8_t)((argb) >> 8), \
    ((uint8_t*)(p))[2] = (uint8_t)((argb) >> 16), \
    ((uint8_t*)(p))[3] = (uint8_t)((argb) >> 24)
#define FXARGB_SETRGBORDERDIB(p, argb) \
    ((uint8_t*)(p))[3] = (uint8_t)(argb >> 24), \
    ((uint8_t*)(p))[0] = (uint8_t)((argb) >> 16), \
    ((uint8_t*)(p))[1] = (uint8_t)((argb) >> 8), \
    ((uint8_t*)(p))[2] = (uint8_t)(argb)
#define FXARGB_TODIB(argb) (argb)
#define FXCMYK_TODIB(cmyk) \
    ((uint8_t)((cmyk) >> 24) | ((uint8_t)((cmyk) >> 16)) << 8 | \
    ((uint8_t)((cmyk) >> 8)) << 16 | ((uint8_t)(cmyk) << 24))
#define FXARGB_TOBGRORDERDIB(argb) \
    ((uint8_t)(argb >> 16) | ((uint8_t)(argb >> 8)) << 8 | \
    ((uint8_t)(argb)) << 16 | ((uint8_t)(argb >> 24) << 24))

FX_RECT FXDIB_SwapClipBox(const FX_RECT& clip,
                          int width,
                          int height,
                          bool bFlipX,
                          bool bFlipY);

#endif // CORE_FXGE_FX_DIB_H

```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_FX_FONT_H_
#define CORE_FXGE_FX_FONT_H_

#include <vector>

#include "core/fxcrt/fx_coordinates.h"
#include "core/fxcrt/fx_string.h"
#include "core/fxcrt/fx_system.h"
#include "core/fxge/fx_freetype.h"
#include "third_party/base/span.h"

/* Font pitch and family flags */
#define FXFONT_FF_FIXEDPITCH (1 << 0)
#define FXFONT_FF_ROMAN (1 << 4)
#define FXFONT_FF_SCRIPT (4 << 4)

/* Typical weight values */
#define FXFONT_FW_NORMAL 400
#define FXFONT_FW_BOLD 700
#define FXFONT_FW_BOLD_BOLD 900

/* Font styles as defined in PDF 1.7 Table 5.20 */
#define FXFONT_NORMAL (0)
#define FXFONT_FIXED_PITCH (1 << 0)
#define FXFONT_SERIF (1 << 1)
#define FXFONT_SYMBOLIC (1 << 2)
#define FXFONT_SCRIPT (1 << 3)
#define FXFONT_NONSYMBOLIC (1 << 5)
#define FXFONT_ITALIC (1 << 6)
#define FXFONT_ALLCAP (1 << 16)
#define FXFONT_SMALLCAP (1 << 17)
#define FXFONT_FORCE_BOLD (1 << 18)

/* Other font flags */
#define FXFONT_USEEXTERNATTR 0x80000
#define FXFONT_CIDFONT 0x100000

#define GET_TT_SHORT(w) ((w)[0] << 8) | (w)[1]
#define GET_TT_LONG(w) \
    ((w)[0] << 24) | ((w)[1] << 16) | ((w)[2] << 8) | (w)[3]

#if defined _SKIA_SUPPORT_ || defined _SKIA_SUPPORT_PATHS_
class SkTypeface;

using CFX_TypeFace = SkTypeface;
#endif

class TextGlyphPos;

FX_RECT GetGlyphsBBox(const std::vector<TextGlyphPos>& glyphs, int anti_alias);

ByteString GetNameFromTT(pdfium::span<const uint8_t> name_table, uint32_t name);
int GetTTIndex(pdfium::span<const uint8_t> pFontData, uint32_t font_offset);

inline bool FontStyleIsForceBold(uint32_t style) {
    return !(style & FXFONT_FORCE_BOLD);
}

```

```
inline bool FontStyleIsItalic(uint32_t style) {
    return !(style & FXFONT_ITALIC);
}
inline bool FontStyleIsFixedPitch(uint32_t style) {
    return !(style & FXFONT_FIXED_PITCH);
}
inline bool FontStyleIsSymbolic(uint32_t style) {
    return !(style & FXFONT_SYMBOLIC);
}
inline bool FontStyleIsNonSymbolic(uint32_t style) {
    return !(style & FXFONT_NONSYMBOLIC);
}
inline bool FontStyleIsAllCaps(uint32_t style) {
    return !(style & FXFONT_ALLCAP);
}
inline bool FontStyleIsSerif(uint32_t style) {
    return !(style & FXFONT_SERIF);
}
inline bool FontStyleIsScript(uint32_t style) {
    return !(style & FXFONT_SCRIPT);
}

inline bool FontFamilyIsFixedPitch(uint32_t family) {
    return !(family & FXFONT_FF_FIXEDPITCH);
}
inline bool FontFamilyIsRoman(uint32_t family) {
    return !(family & FXFONT_FF_ROMAN);
}
inline bool FontFamilyIsScript(uint32_t family) {
    return !(family & FXFONT_FF_SCRIPT);
}

wchar_t PDF_UnicodeFromAdobeName(const char* name);
ByteString PDF_AdobeNameFromUnicode(wchar_t unicode);

#endif // CORE_FXGE_FX_FONT_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_FX_FREETYPE_H_
#define CORE_FXGE_FX_FREETYPE_H_

#include <ft2build.h>

#include <memory>

#include FT_FREETYPE_H
#include FT_GLYPH_H
#include FT_LCD_FILTER_H
#include FT_MULTIPLE_MASTERS_H
#include FT_OUTLINE_H
#include FT_TRUETYPE_TABLES_H

using FXFT_LibraryRec = struct FT_LibraryRec_;
using FXFT_FaceRec = struct FT_FaceRec_;
using FXFT_StreamRec = struct FT_StreamRec_;
using FXFT_MM_VarPtr = FT_MM_Var*;

struct FXFTFaceRecDeleter {
    inline void operator()(FXFT_FaceRec* pRec) {
        if (pRec)
            FT_Done_Face(pRec);
    }
};

struct FXFTLibraryRecDeleter {
    inline void operator()(FXFT_LibraryRec* pRec) {
        if (pRec)
            FT_Done_FreeType(pRec);
    }
};

using ScopedFXFTFaceRec = std::unique_ptr<FXFT_FaceRec, FXFTFaceRecDeleter>;
using ScopedFXFTLibraryRec =
    std::unique_ptr<FXFT_LibraryRec, FXFTLibraryRecDeleter>;

#define FXFT_Select_Charmap(face, encoding) \
    FT_Select_Charmap(face, static_cast<FT_Encoding>(encoding))
#define FXFT_Get_Name_Index(face, name) \
    FT_Get_Name_Index(face, const_cast<char*>(name))
#define FXFT_Get_Glyph_Outline(face) &((face)->glyph->outline)
#define FXFT_Render_Glyph(face, mode) \
    FT_Render_Glyph((face)->glyph, static_cast<enum FT_Render_Mode_>(mode))

#define FXFT_Has_Glyph_Names(face) \
    (((face)->face_flags) & FT_FACE_FLAG_GLYPH_NAMES)
#define FXFT_Clear_Face_External_Stream(face) \
    ((face)->face_flags &= ~FT_FACE_FLAG_EXTERNAL_STREAM)
#define FXFT_Get_Face_External_Stream(face) \
    (((face)->face_flags) & FT_FACE_FLAG_EXTERNAL_STREAM)
#define FXFT_Is_Face_TT_OT(face) (((face)->face_flags) & FT_FACE_FLAG_SFNT)
#define FXFT_Is_Face_Tricky(face) (((face)->face_flags) & FT_FACE_FLAG_TRICKY)
#define FXFT_Is_Face_fixedwidth(face) \
    (((face)->face_flags) & FT_FACE_FLAG_FIXED_WIDTH)
#define FXFT_Get_Face_Stream_Base(face) (face)->stream->base
#define FXFT_Get_Face_Stream_Size(face) (face)->stream->size
```

```
#define FXFT_Get_Face_Family_Name(face) (face)->family_name
#define FXFT_Get_Face_Style_Name(face) (face)->style_name
#define FXFT_Is_Face_Italic(face) (((face)->style_flags) & FT_STYLE_FLAG_ITALIC)
#define FXFT_Is_Face_Bold(face) (((face)->style_flags) & FT_STYLE_FLAG_BOLD)
#define FXFT_Get_Face_Charmaps(face) (face)->charmaps
#define FXFT_Get_Glyph_HoriBearingX(face) (face)->glyph->metrics.horiBearingX
#define FXFT_Get_Glyph_HoriBearingY(face) (face)->glyph->metrics.horiBearingY
#define FXFT_Get_Glyph_Width(face) (face)->glyph->metrics.width
#define FXFT_Get_Glyph_Height(face) (face)->glyph->metrics.height
#define FXFT_Get_Face_CharmapCount(face) (face)->num_charmaps
#define FXFT_Get_Charmap_Encoding(charmap) (charmap)->encoding
#define FXFT_Get_Face_Charmap(face) (face)->charmap
#define FXFT_Get_Charmap_PlatformID(charmap) (charmap)->platform_id
#define FXFT_Get_Charmap_EncodingID(charmap) (charmap)->encoding_id
#define FXFT_Get_Face_UnitsPerEM(face) (face)->units_per_EM
#define FXFT_Get_Face_xMin(face) (face)->bbox.xMin
#define FXFT_Get_Face_xMax(face) (face)->bbox.xMax
#define FXFT_Get_Face_yMin(face) (face)->bbox.yMin
#define FXFT_Get_Face_yMax(face) (face)->bbox.yMax
#define FXFT_Get_Face_Height(face) (face)->height
#define FXFT_Get_Face_Ascender(face) (face)->ascender
#define FXFT_Get_Face_Descender(face) (face)->descender
#define FXFT_Get_Glyph_HoriAdvance(face) (face)->glyph->metrics.horiAdvance
#define FXFT_Get_MM_Axis(var, index) (var)->axis[index]
#define FXFT_Get_MM_Axis_Min(axis) (axis).minimum
#define FXFT_Get_MM_Axis_Max(axis) (axis).maximum
#define FXFT_Get_MM_Axis_Def(axis) (axis).def
#define FXFT_Free(face, p) (face)->memory->free((face)->memory, p)
#define FXFT_Get_Glyph_Outline(face) &((face)->glyph->outline)
#define FXFT_Get_Glyph_Bitmap(face) (face)->glyph->bitmap
#define FXFT_Get_Bitmap_Width(bitmap) (bitmap).width
#define FXFT_Get_Bitmap_Rows(bitmap) (bitmap).rows
#define FXFT_Get_Bitmap_PixelMode(bitmap) (bitmap).pixel_mode
#define FXFT_Get_Bitmap_Pitch(bitmap) (bitmap).pitch
#define FXFT_Get_Bitmap_Buffer(bitmap) (bitmap).buffer
#define FXFT_Get_Glyph_BitmapLeft(face) (face)->glyph->bitmap_left
#define FXFT_Get_Glyph_BitmapTop(face) (face)->glyph->bitmap_top

int FXFT_unicode_from_adobe_name(const char* glyph_name);
void FXFT_adobe_name_from_unicode(char* name, wchar_t unicode);

#endif // CORE_FXGE_FX_FREETYPE_H
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FXGE_RENDER_DEFINES_H
```

```
#define CORE_FXGE_RENDER_DEFINES_H
```

```
#define FXDC_PIXEL_WIDTH 2
```

```
#define FXDC_PIXEL_HEIGHT 3
```

```
#define FXDC_BITS_PIXEL 4
```

```
#define FXDC_HORZ_SIZE 5
```

```
#define FXDC_VERT_SIZE 6
```

```
#define FXDC_RENDER_CAPS 7
```

```
#define FXRC_GET_BITS 0x01
```

```
#define FXRC_BIT_MASK 0x02
```

```
#define FXRC_ALPHA_PATH 0x10
```

```
#define FXRC_ALPHA_IMAGE 0x20
```

```
#define FXRC_ALPHA_OUTPUT 0x40
```

```
#define FXRC_BLEND_MODE 0x80
```

```
#define FXRC_SOFT_CLIP 0x100
```

```
#define FXRC_CMYK_OUTPUT 0x200
```

```
#define FXRC_BITMASK_OUTPUT 0x400
```

```
#define FXRC_BYTEMASK_OUTPUT 0x800
```

```
#define FXRC_FILLSTROKE_LOSSY 0x1000
```

```
#define FXRC_FILLSTROKE_PATH 0x2000
```

```
#define FXRC_SHADING 0x4000
```

```
#define FXFILL_ALTERNATE 1
```

```
#define FXFILL_WINDING 2
```

```
#define FXFILL_FULLCOVER 4
```

```
#define FXFILL_RECT_AA 8
```

```
#define FX_FILL_STROKE 16
```

```
#define FX_STROKE_ADJUST 32
```

```
#define FX_STROKE_TEXT_MODE 64
```

```
#define FX_FILL_TEXT_MODE 128
```

```
#define FX_ZEROAREA_FILL 256
```

```
#define FXFILL_NOPATHSMOOTH 512
```

```
#define FXTEXT_CLEARTEXT 0x01
```

```
#define FXTEXT_BGR_STRIPE 0x02
```

```
#define FXTEXT_PRINTGRAPHICTEXT 0x04
```

```
#define FXTEXT_NO_NATIVE_TEXT 0x08
```

```
#define FXTEXT_PRINTIMAGETEXT 0x10
```

```
#define FXTEXT_NOSMOOTH 0x20
```

```
#endif // CORE_FXGE_RENDER_DEFINES_H
```



```
        BlendMode blend_type);
virtual bool DrawCosmeticLine(const CFX_PointF& ptMoveTo,
        const CFX_PointF& ptLineTo,
        uint32_t color,
        BlendMode blend_type);

virtual bool GetClipBox(FX_RECT* pRect) = 0;
virtual bool GetDIBits(const RetainPtr<CFX_DIBitmap>& pBitmap,
        int left,
        int top);
virtual RetainPtr<CFX_DIBitmap> GetBackDrop();
virtual bool SetDIBits(const RetainPtr<CFX_DIBase>& pBitmap,
        uint32_t color,
        const FX_RECT& src_rect,
        int dest_left,
        int dest_top,
        BlendMode blend_type) = 0;
virtual bool StretchDIBits(const RetainPtr<CFX_DIBase>& pBitmap,
        uint32_t color,
        int dest_left,
        int dest_top,
        int dest_width,
        int dest_height,
        const FX_RECT* pClipRect,
        const FXDIB_ResampleOptions& options,
        BlendMode blend_type) = 0;
virtual bool StartDIBits(const RetainPtr<CFX_DIBase>& pBitmap,
        int bitmap_alpha,
        uint32_t color,
        const CFX_Matrix& matrix,
        const FXDIB_ResampleOptions& options,
        std::unique_ptr<CFX_ImageRenderer>* handle,
        BlendMode blend_type) = 0;
virtual bool ContinueDIBits(CFX_ImageRenderer* handle,
        PauseIndicatorIface* pPause);
virtual bool DrawDeviceText(int nChars,
        const TextCharPos* pCharPos,
        CFX_Font* pFont,
        const CFX_Matrix& mtObject2Device,
        float font_size,
        uint32_t color);

virtual int GetDriverType() const;
virtual void ClearDriver();
virtual bool DrawShading(const CPDF_ShadingPattern* pPattern,
        const CFX_Matrix* pMatrix,
        const FX_RECT& clip_rect,
        int alpha,
        bool bAlphaMode);
virtual bool SetBitsWithMask(const RetainPtr<CFX_DIBase>& pBitmap,
        const RetainPtr<CFX_DIBase>& pMask,
        int left,
        int top,
        int bitmap_alpha,
        BlendMode blend_type);
#if defined _SKIA_SUPPORT_ || defined _SKIA_SUPPORT_PATHS_
    virtual void Flush();
#endif
};

#endif // CORE_FXGE_RENDERDEVICEDRIVER_IFACE_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_SCOPED_FONT_TRANSFORM_H_
#define CORE_FXGE_SCOPED_FONT_TRANSFORM_H_

#include "core/fxge/cfx_face.h"
#include "core/fxge/fx_freetype.h"

// Sets the given transform on the font, and resets it to the identity when it
// goes out of scope.
class ScopedFontTransform {
public:
  ScopedFontTransform(RefPtr<CFX_Face> face, FT_Matrix* matrix);
  ~ScopedFontTransform();

private:
  RefPtr<CFX_Face> m_Face;
};

#endif // CORE_FXGE_SCOPED_FONT_TRANSFORM_H_
```





```
bool DrawBitsWithMask(const RetainPtr<CFX_DIBBase>& pBitmap,
                     const RetainPtr<CFX_DIBBase>& pMask,
                     int bitmap_alpha,
                     const CFX_Matrix& matrix,
                     BlendMode blend_type);

bool DrawDeviceText(int nChars,
                   const TextCharPos* pCharPos,
                   CFX_Font* pFont,
                   const CFX_Matrix& mtObject2Device,
                   float font_size,
                   uint32_t color) override;

int GetDriverType() const override;

bool DrawShading(const CPDF_ShadingPattern* pPattern,
                 const CFX_Matrix* pMatrix,
                 const FX_RECT& clip_rect,
                 int alpha,
                 bool bAlphaMode) override;

virtual uint8_t* GetBuffer() const;

void PaintStroke(SkPaint* spaint,
                 const CFX_GraphStateData* pGraphState,
                 const SkMatrix& matrix);

void Clear(uint32_t color);
void Flush() override;
SkPictureRecorder* GetRecorder() const { return m_pRecorder; }
void PreMultiply();
static void PreMultiply(const RetainPtr<CFX_DIBitmap>& pDIBitmap);
SkCanvas* SkiaCanvas() { return m_pCanvas; }
void DebugVerifyBitmapIsPreMultiplied() const;
void Dump() const;

bool GetGroupKnockout() const { return m_bGroupKnockout; }

#ifdef _SKIA_SUPPORT_PATHS_
const CFX_ClipRgn* clip_region() const { return m_pClipRgn.get(); }
const std::vector<std::unique_ptr<CFX_ClipRgn>>& stack() const {
    return m_StateStack;
}
#endif

private:
    RetainPtr<CFX_DIBitmap> m_pBitmap;
    RetainPtr<CFX_DIBitmap> m_pBackdropBitmap;
    SkCanvas* m_pCanvas;
    SkPictureRecorder* const m_pRecorder;
    std::unique_ptr<SkiaState> m_pCache;
#ifdef _SKIA_SUPPORT_PATHS_
    std::unique_ptr<CFX_ClipRgn> m_pClipRgn;
    std::vector<std::unique_ptr<CFX_ClipRgn>> m_StateStack;
    int m_FillFlags;
    bool m_bRgbByteOrder;
#endif
    bool m_bGroupKnockout;
};
#endif // defined _SKIA_SUPPORT_ || defined _SKIA_SUPPORT_PATHS_

#endif // CORE_FXGE_SKIA_FX_SKIA_DEVICE_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_SYSTEMFONTINFO_IFACE_H
#define CORE_FXGE_SYSTEMFONTINFO_IFACE_H

#include <memory>

#include "core/fxge/cfx_fontmapper.h"
#include "third_party/base/span.h"

constexpr uint32_t kTableName = CFX_FontMapper::MakeTag('n', 'a', 'm', 'e');
constexpr uint32_t kTableTTCF = CFX_FontMapper::MakeTag('t', 't', 'c', 'f');

class SystemFontInfoIface {
public:
    static std::unique_ptr<SystemFontInfoIface> CreateDefault(
        const char** pUserPaths);

    virtual ~SystemFontInfoIface() = default;

    virtual bool EnumFontList(CFX_FontMapper* pMapper) = 0;
    virtual void* MapFont(int weight,
                          bool bItalic,
                          int charset,
                          int pitch_family,
                          const char* face) = 0;
    virtual void* GetFont(const char* face) = 0;
    virtual uint32_t GetFontData(void* hFont,
                                 uint32_t table,
                                 pdfium::span<uint8_t> buffer) = 0;
    virtual bool GetFaceName(void* hFont, ByteString* name) = 0;
    virtual bool GetFontCharset(void* hFont, int* charset) = 0;
    virtual int GetFaceIndex(void* hFont);
    virtual void DeleteFont(void* hFont) = 0;
};

#endif // CORE_FXGE_SYSTEMFONTINFO_IFACE_H
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_TEXT_CHAR_POS_H_
#define CORE_FXGE_TEXT_CHAR_POS_H_

#include "core/fxcrt/fx_coordinates.h"

class TextCharPos {
public:
  TextCharPos();
  TextCharPos(const TextCharPos&);
  ~TextCharPos();

  CFX_PointF m-Origin;
  uint32_t m_Unicode = 0;
  uint32_t m_GlyphIndex = 0;
  uint32_t m_FontCharWidth = 0;
#ifdef OS_MACOSX
  uint32_t m_ExtGID = 0;
#endif
  int32_t m_FallbackFontPosition = 0;
  bool m_bGlyphAdjust = false;
  bool m_bFontStyle = false;
  float m_AdjustMatrix[4];
};

#endif // CORE_FXGE_TEXT_CHAR_POS_H_
```

```
// Copyright 2019 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_TEXT_GLYPH_POS_H_
#define CORE_FXGE_TEXT_GLYPH_POS_H_

#include "core/fxcrt/fx_coordinates.h"

#include "core/fxcrt/unowned_ptr.h"
#include "third_party/base/optional.h"

class CFX_GlyphBitmap;

class TextGlyphPos {
public:
  TextGlyphPos();
  TextGlyphPos(const TextGlyphPos&);
  ~TextGlyphPos();

  Optional<CFX_Point> GetOrigin(const CFX_Point& offset) const;

  UnownedPtr<const CFX_GlyphBitmap> m_pGlyph;
  CFX_Point m_Origin;
  CFX_PointF m_fOrigin;
};

#endif // CORE_FXGE_TEXT_GLYPH_POS_H_
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
```

```
// Use of this source code is governed by a BSD-style license that can be
```

```
// found in the LICENSE file.
```

```
// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com
```

```
#ifndef CORE_FXGE_WIN32_CFX_PSRENDERER_H
```

```
#define CORE_FXGE_WIN32_CFX_PSRENDERER_H
```

```
#include <memory>
```

```
#include <vector>
```

```
#include "core/fxcrt/fx_coordinates.h"
```

```
#include "core/fxcrt/fx_memory_wrappers.h"
```

```
#include "core/fxcrt/fx_stream.h"
```

```
#include "core/fxcrt/fx_system.h"
```

```
#include "core/fxcrt/retain_ptr.h"
```

```
#include "core/fxge/cfx_graphstatedata.h"
```

```
class CFX_DIBBase;
```

```
class CFX_GlyphCache;
```

```
class CFX_Font;
```

```
class CFX_PathData;
```

```
class CPSFont;
```

```
class TextCharPos;
```

```
struct FXDIB_ResampleOptions;
```

```
struct EncoderIface {
```

```
    bool (*pA85EncodeFunc)(pdfium::span<const uint8_t> src_buf,  
                           std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,  
                           uint32_t* dest_size);
```

```
    void (*pFaxEncodeFunc)(const uint8_t* src_buf,  
                           int width,  
                           int height,  
                           int pitch,  
                           std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,  
                           uint32_t* dest_size);
```

```
    bool (*pFlateEncodeFunc)(const uint8_t* src_buf,  
                              uint32_t src_size,  
                              std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,  
                              uint32_t* dest_size);
```

```
    bool (*pJpegEncodeFunc)(const RetainPtr<CFX_DIBBase>& pSource,  
                             uint8_t** dest_buf,  
                             size_t* dest_size);
```

```
    bool (*pRunLengthEncodeFunc)(  
        pdfium::span<const uint8_t> src_buf,  
        std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,  
        uint32_t* dest_size);
```

```
};
```

```
class CFX_PSRenderer {
```

```
public:
```

```
    explicit CFX_PSRenderer(const EncoderIface* pEncoderIface);  
    ~CFX_PSRenderer();
```

```
    void Init(const RetainPtr<IFX_RetainableWriteStream>& stream,  
             int pslevel,  
             int width,  
             int height,  
             bool bCmykOutput);
```

```
    bool StartRendering();
```

```
    void EndRendering();
```

```
    void SaveState();
```

```
void RestoreState(bool bKeepSaved);
void SetClip_PathFill(const CFX_PathData* pPathData,
                     const CFX_Matrix* pObject2Device,
                     int fill_mode);
void SetClip_PathStroke(const CFX_PathData* pPathData,
                        const CFX_Matrix* pObject2Device,
                        const CFX_GraphStateData* pGraphState);
FX_RECT GetClipBox() { return m_ClipBox; }
bool DrawPath(const CFX_PathData* pPathData,
              const CFX_Matrix* pObject2Device,
              const CFX_GraphStateData* pGraphState,
              uint32_t fill_color,
              uint32_t stroke_color,
              int fill_mode);
bool SetDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
               uint32_t color,
               int dest_left,
               int dest_top);
bool StretchDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
                  uint32_t color,
                  int dest_left,
                  int dest_top,
                  int dest_width,
                  int dest_height,
                  const FXDIB_ResampleOptions& options);
bool DrawDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
                uint32_t color,
                const CFX_Matrix& matrix,
                const FXDIB_ResampleOptions& options);
bool DrawText(int nChars,
              const TextCharPos* pCharPos,
              CFX_Font* pFont,
              const CFX_Matrix& mtObject2Device,
              float font_size,
              uint32_t color);

private:
void OutputPath(const CFX_PathData* pPathData,
                const CFX_Matrix* pObject2Device);
void SetGraphState(const CFX_GraphStateData* pGraphState);
void SetColor(uint32_t color);
void FindPSFontGlyph(CFX_GlyphCache* pGlyphCache,
                     CFX_Font* pFont,
                     const TextCharPos& charpos,
                     int* ps_fontnum,
                     int* ps_glyphindex);
bool FaxCompressData(std::unique_ptr<uint8_t, FxFreeDeleter> src_buf,
                    int width,
                    int height,
                    std::unique_ptr<uint8_t, FxFreeDeleter>* dest_buf,
                    uint32_t* dest_size) const;
void PSCompressData(uint8_t* src_buf,
                   uint32_t src_size,
                   uint8_t** output_buf,
                   uint32_t* output_size,
                   const char** filter) const;
void WritePSBinary(const uint8_t* data, int len);
void WriteToStream(std::ostream* stringStream);

bool m_bInited = false;
bool m_bGraphStateSet = false;
bool m_bCmykOutput;
bool m_bColorSet = false;
```

```
int m_PSLevel = 0;
uint32_t m_LastColor = 0;
FX_RECT m_ClipBox;
CFX_GraphStateData m_CurGraphState;
const EncoderIface* const m_pEncoderIface;
RetainPtr<IFX_RetainableWriteStream> m_pStream;
std::vector<std::unique_ptr<CPSFont>> m_PSFontList;
std::vector<FX_RECT> m_ClipBoxStack;
};

#endif // CORE_FXGE_WIN32_CFX_PSRENDERER_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifdef CORE_FXGE_WIN32_CFX_WINDOWS_DIB_H
#define CORE_FXGE_WIN32_CFX_WINDOWS_DIB_H

#include <windows.h>

#include "core/fxcrt/bytestring.h"
#include "core/fxge/dib/cfx_dibitmap.h"

#define WINDIB_OPEN_MEMORY 0x1
#define WINDIB_OPEN_PATHNAME 0x2

struct WINDIB_Open_Args_ {
    int flags;
    const uint8_t* memory_base;
    size_t memory_size;
    const wchar_t* path_name;
};

class CFX_WindowsDIB final : public CFX_DIBitmap {
public:
    template <typename T, typename... Args>
    friend RetainPtr<T> pdfium::MakeRetain(Args&&... args);

    static ByteString GetBitmapInfo(const RetainPtr<CFX_DIBitmap>& pBitmap);
    static HBITMAP GetDDBitmap(const RetainPtr<CFX_DIBitmap>& pBitmap, HDC hDC);

    static RetainPtr<CFX_DIBitmap> LoadFromBuf(BITMAPINFO* pbmi, void* pData);
    static RetainPtr<CFX_DIBitmap> LoadFromFile(const wchar_t* filename);
    static RetainPtr<CFX_DIBitmap> LoadFromFile(const char* filename);
    static RetainPtr<CFX_DIBitmap> LoadDIBitmap(WINDIB_Open_Args_ args);

    HBITMAP GetWindowsBitmap() const { return m_hBitmap; }

    void LoadFromDevice(HDC hDC, int left, int top);
    void SetToDevice(HDC hDC, int left, int top);

private:
    CFX_WindowsDIB(HDC hDC, int width, int height);
    ~CFX_WindowsDIB() override;

    HDC m_hMemDC;
    HBITMAP m_hBitmap;
    HBITMAP m_hOldBitmap;
};

#endif // CORE_FXGE_WIN32_CFX_WINDOWS_DIB_H
```

```
// Copyright 2016 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_WIN32_CPSOUTPUT_H_
#define CORE_FXGE_WIN32_CPSOUTPUT_H_

#include <windows.h>

#include "core/fxcrt/fx_stream.h"
#include "core/fxcrt/fx_system.h"

class CPSOutput final : public IFX_RetainableWriteStream {
public:
  enum class OutputMode { kExtEscape, kGdiComment };

  CPSOutput(HDC hDC, OutputMode mode);
  ~CPSOutput() override;

  // IFX_Writestream
  bool WriteBlock(const void* str, size_t len) override;
  bool WriteString(ByteStringView str) override;

private:
  const HDC m_hDC;
  const OutputMode m_mode;
};

#endif // CORE_FXGE_WIN32_CPSOUTPUT_H_
```

```
// Copyright 2014 PDFium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Original code copyright 2014 Foxit Software Inc. http://www.foxitsoftware.com

#ifndef CORE_FXGE_WIN32_WIN32_INT_H_
#define CORE_FXGE_WIN32_WIN32_INT_H_

#include <windows.h>

#include <memory>
#include <vector>

#include "core/fxcrt/retain_ptr.h"
#include "core/fxge/cfx_gemodule.h"
#include "core/fxge/cfx_pathdata.h"
#include "core/fxge/cfx_windowsrenderdevice.h"
#include "core/fxge/renderdevicedriver_iface.h"
#include "core/fxge/win32/cfx_psrenderer.h"
#include "core/fxge/win32/cpsoutput.h"
#include "third_party/base/optional.h"

class CFX_ImageRenderer;
class TextCharPos;
struct WINDIB_Open_Args_;

RetainPtr<CFX_DIBitmap> FX_WindowsDIB_LoadFromBuf(BITMAPINFO* pbmi,
                                                LPVOID pData,
                                                bool bAlpha);

class CGdiplusExt {
public:
  CGdiplusExt();
  ~CGdiplusExt();

  void Load();
  bool IsAvailable() { return !!m_hModule; }
  bool StretchDIBits(HDC hDC,
                    const RetainPtr<CFX_DIBitmap>& pBitmap,
                    int dest_left,
                    int dest_top,
                    int dest_width,
                    int dest_height,
                    const FX_RECT* pClipRect,
                    const FXDIB_ResampleOptions& options);

  bool DrawPath(HDC hDC,
               const CFX_PathData* pPathData,
               const CFX_Matrix* pObject2Device,
               const CFX_GraphStateData* pGraphState,
               uint32_t fill_argb,
               uint32_t stroke_argb,
               int fill_mode);

  RetainPtr<CFX_DIBitmap> LoadDIBitmap(WINDIB_Open_Args_ args);

  std::vector<FARPROC> m_Functions;

protected:
  HMODULE m_hModule = nullptr;
  HMODULE m_GdiModule = nullptr;
};

class CWin32Platform : public CFX_GEModule::PlatformIface {
```

```
public:
    CWin32Platform();
    ~CWin32Platform() override;

    // CFX_GEModule::PlatformIface:
    void Init() override;

    bool m_bHalfTone = false;
    CGdiplusExt m_GdiplusExt;
};

class CGdiDeviceDriver : public RenderDeviceDriverIface {
protected:
    CGdiDeviceDriver(HDC hDC, DeviceType device_type);
    ~CGdiDeviceDriver() override;

    // RenderDeviceDriverIface:
    DeviceType GetDeviceType() const override;
    int GetDeviceCaps(int caps_id) const override;
    void SaveState() override;
    void RestoreState(bool bKeepSaved) override;
    void SetBaseClip(const FX_RECT& rect) override;
    bool SetClip_PathFill(const CFX_PathData* pPathData,
                          const CFX_Matrix* pObject2Device,
                          int fill_mode) override;
    bool SetClip_PathStroke(const CFX_PathData* pPathData,
                            const CFX_Matrix* pObject2Device,
                            const CFX_GraphStateData* pGraphState) override;
    bool DrawPath(const CFX_PathData* pPathData,
                 const CFX_Matrix* pObject2Device,
                 const CFX_GraphStateData* pGraphState,
                 uint32_t fill_color,
                 uint32_t stroke_color,
                 int fill_mode,
                 BlendMode blend_type) override;
    bool FillRectWithBlend(const FX_RECT& rect,
                          uint32_t fill_color,
                          BlendMode blend_type) override;
    bool DrawCosmeticLine(const CFX_PointF& ptMoveTo,
                         const CFX_PointF& ptLineTo,
                         uint32_t color,
                         BlendMode blend_type) override;
    bool GetClipBox(FX_RECT* pRect) override;

    void DrawLine(float x1, float y1, float x2, float y2);

    bool GDI_SetDIBits(const RetainPtr<CFX_DIBitmap>& pBitmap,
                     const FX_RECT& src_rect,
                     int left,
                     int top);
    bool GDI_StretchDIBits(const RetainPtr<CFX_DIBitmap>& pBitmap,
                          int dest_left,
                          int dest_top,
                          int dest_width,
                          int dest_height,
                          const FXDIB_ResampleOptions& options);
    bool GDI_StretchBitMask(const RetainPtr<CFX_DIBitmap>& pBitmap,
                          int dest_left,
                          int dest_top,
                          int dest_width,
                          int dest_height,
                          uint32_t bitmap_color);
};
```

```
    const HDC m_hDC;
    bool m_bMetafileDCType;
    int m_Width;
    int m_Height;
    int m_nBitsPerPixel;
    const DeviceType m_DeviceType;
    int m_RenderCaps;
    pdfium::Optional<FX_RECT> m_BaseClipBox;
};

class CGdiDisplayDriver final : public CGdiDeviceDriver {
public:
    explicit CGdiDisplayDriver(HDC hDC);
    ~CGdiDisplayDriver() override;

private:
    // CGdiDisplayDriver:
    int GetDeviceCaps(int caps_id) const override;
    bool GetDIBits(const RetainPtr<CFX_DIBitmap>& pBitmap,
                  int left,
                  int top) override;
    bool SetDIBits(const RetainPtr<CFX_DIBase>& pBitmap,
                  uint32_t color,
                  const FX_RECT& src_rect,
                  int left,
                  int top,
                  BlendMode blend_type) override;
    bool StretchDIBits(const RetainPtr<CFX_DIBase>& pBitmap,
                      uint32_t color,
                      int dest_left,
                      int dest_top,
                      int dest_width,
                      int dest_height,
                      const FX_RECT* pClipRect,
                      const FXDIB_ResampleOptions& options,
                      BlendMode blend_type) override;
    bool StartDIBits(const RetainPtr<CFX_DIBase>& pBitmap,
                    int bitmap_alpha,
                    uint32_t color,
                    const CFX_Matrix& matrix,
                    const FXDIB_ResampleOptions& options,
                    std::unique_ptr<CFX_ImageRenderer>* handle,
                    BlendMode blend_type) override;
    bool UseFoxitStretchEngine(const RetainPtr<CFX_DIBase>& pSource,
                              uint32_t color,
                              int dest_left,
                              int dest_top,
                              int dest_width,
                              int dest_height,
                              const FX_RECT* pClipRect,
                              const FXDIB_ResampleOptions& options);
};

class CGdiPrinterDriver final : public CGdiDeviceDriver {
public:
    explicit CGdiPrinterDriver(HDC hDC);
    ~CGdiPrinterDriver() override;

private:
    // CGdiPrinterDriver:
    int GetDeviceCaps(int caps_id) const override;
    bool SetDIBits(const RetainPtr<CFX_DIBase>& pBitmap,
                  uint32_t color,
```

```

        const FX_RECT& src_rect,
        int left,
        int top,
        BlendMode blend_type) override;
bool StretchDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
                  uint32_t color,
                  int dest_left,
                  int dest_top,
                  int dest_width,
                  int dest_height,
                  const FX_RECT* pClipRect,
                  const FXDIB_ResampleOptions& options,
                  BlendMode blend_type) override;
bool StartDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
                 int bitmap_alpha,
                 uint32_t color,
                 const CFX_Matrix& matrix,
                 const FXDIB_ResampleOptions& options,
                 std::unique_ptr<CFX_ImageRenderer>* handle,
                 BlendMode blend_type) override;
bool DrawDeviceText(int nChars,
                   const TextCharPos* pCharPos,
                   CFX_Font* pFont,
                   const CFX_Matrix& mtObject2Device,
                   float font_size,
                   uint32_t color) override;

const int m_HorzSize;
const int m_VertSize;
};

class CPSPrinterDriver final : public RenderDeviceDriverIface {
public:
    CPSPrinterDriver(HDC hDC,
                    WindowsPrintMode mode,
                    bool bCmykOutput,
                    const EncoderIface* pEncoderIface);
    ~CPSPrinterDriver() override;

private:
    // RenderDeviceDriverIface:
    DeviceType GetDeviceType() const override;
    int GetDeviceCaps(int caps_id) const override;
    bool StartRendering() override;
    void EndRendering() override;
    void SaveState() override;
    void RestoreState(bool bKeepSaved) override;
    bool SetClip_PathFill(const CFX_PathData* pPathData,
                         const CFX_Matrix* pObject2Device,
                         int fill_mode) override;
    bool SetClip_PathStroke(const CFX_PathData* pPathData,
                           const CFX_Matrix* pObject2Device,
                           const CFX_GraphStateData* pGraphState) override;
    bool DrawPath(const CFX_PathData* pPathData,
                 const CFX_Matrix* pObject2Device,
                 const CFX_GraphStateData* pGraphState,
                 uint32_t fill_color,
                 uint32_t stroke_color,
                 int fill_mode,
                 BlendMode blend_type) override;
    bool GetClipBox(FX_RECT* pRect) override;
    bool SetDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
                  uint32_t color,

```

```

        const FX_RECT& src_rect,
        int left,
        int top,
        BlendMode blend_type) override;
bool StretchDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
                  uint32_t color,
                  int dest_left,
                  int dest_top,
                  int dest_width,
                  int dest_height,
                  const FX_RECT* pClipRect,
                  const FXDIB_ResampleOptions& options,
                  BlendMode blend_type) override;
bool StartDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
                 int bitmap_alpha,
                 uint32_t color,
                 const CFX_Matrix& matrix,
                 const FXDIB_ResampleOptions& options,
                 std::unique_ptr<CFX_ImageRenderer>* handle,
                 BlendMode blend_type) override;
bool DrawDeviceText(int nChars,
                   const TextCharPos* pCharPos,
                   CFX_Font* pFont,
                   const CFX_Matrix& mtObject2Device,
                   float font_size,
                   uint32_t color) override;

HDC m_hDC;
const bool m_bCmykOutput;
int m_Width;
int m_Height;
int m_nBitsPerPixel;
int m_HorzSize;
int m_VertSize;
CFX_PSRenderer m_PSRenderer;
};

class CTextOnlyPrinterDriver final : public RenderDeviceDriverIface {
public:
    explicit CTextOnlyPrinterDriver(HDC hDC);
    ~CTextOnlyPrinterDriver() override;

private:
    // RenderDeviceDriverIface:
    DeviceType GetDeviceType() const override;
    int GetDeviceCaps(int caps_id) const override;
    void SaveState() override {}
    void RestoreState(bool bKeepSaved) override {}
    bool SetClip_PathFill(const CFX_PathData* pPathData,
                         const CFX_Matrix* pObject2Device,
                         int fill_mode) override;
    bool SetClip_PathStroke(const CFX_PathData* pPathData,
                           const CFX_Matrix* pObject2Device,
                           const CFX_GraphStateData* pGraphState) override;
    bool DrawPath(const CFX_PathData* pPathData,
                 const CFX_Matrix* pObject2Device,
                 const CFX_GraphStateData* pGraphState,
                 uint32_t fill_color,
                 uint32_t stroke_color,
                 int fill_mode,
                 BlendMode blend_type) override;
    bool GetClipBox(FX_RECT* pRect) override;
    bool SetDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,

```

```
    uint32_t color,
    const FX_RECT& src_rect,
    int left,
    int top,
    BlendMode blend_type) override;
bool StretchDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
    uint32_t color,
    int dest_left,
    int dest_top,
    int dest_width,
    int dest_height,
    const FX_RECT* pClipRect,
    const FXDIB_ResampleOptions& options,
    BlendMode blend_type) override;
bool StartDIBits(const RetainPtr<CFX_DIBBase>& pBitmap,
    int bitmap_alpha,
    uint32_t color,
    const CFX_Matrix& matrix,
    const FXDIB_ResampleOptions& options,
    std::unique_ptr<CFX_ImageRenderer>* handle,
    BlendMode blend_type) override;
bool DrawDeviceText(int nChars,
    const TextCharPos* pCharPos,
    CFX_Font* pFont,
    const CFX_Matrix& mtObject2Device,
    float font_size,
    uint32_t color) override;

HDC m_hDC;
const int m_Width;
const int m_Height;
int m_nBitsPerPixel;
const int m_HorzSize;
const int m_VertSize;
float m_OriginY;
bool m_SetOrigin;
};
#endif // CORE_FXGE_WIN32_WIN32_INT_H
```