NextLabs CADRMX Client SDK

v1.0

Chapter 1

CONFIDENTIALITY NOTICE

To provide feedback on this document, email the documentation team at techpubs@nextlabs.com.

TRADEMARKS

LICENSE AGREEMENT

Published in San Mateo, CA, by NextLabs, Inc.
https://www.nextlabs.com
info@nextlabs.com
support@nextlabs.com
650.577.9101

Chapter 2

Introduction

2.1 Overview

The CADRMX Client SDK provides object-oriented C++ APIs to develop Rights Management eXtension (RMX) for CAD applications.

The following table describes the terminology and components.

Terminology/Component	Description
Rights Management eXtension (RMX)	A plugin of CAD appliations.
Rights Protection Manager (RPM)	A Windows driver running in the kernel.
RPM Folder	A secure folder to let native application handle NXL file transparent.
NXL file	The NextLabs encrypt file format with .nxl extension.
Tag	The meta data persistent in NXL file.

As a developer, you can use CADRMX Client SDK to accomplish the following tasks:

- Protect file with tags.
- Evaluate rights for the rights-projected file.
- Access and edit rights-protected files.
- Manage RPM folders.
- Manage RPM trusted applications.
- Create view overlay with dynamic watermark.
- Evaluate resource rights from central policies.

Next topics:
Platform and Environment Support
Getting Started

Chapter 3

Platform and Environment Support

3.1 Operating System

- Windows 7 64-bit
- Windows 10 64-bit

3.2 Runtime Environment

- NextLabs SkyDRM Desktop for Windows
- NextLabs SkyDRM Rights Protection Manager
- NextLabs SkyDRM Rights Management Server
- NextLabs Control Center

3.3 Development Environment

- Microsoft Visual Studio 2015 with Update 3
- CADRMX Client SDK

Next topics:
Getting Started

Chapter 4

Getting Started

4.1 Setting up Development Environment

- Create a Visual Studio C++ project for RMX module.
- Configure the following Studio project properties:
  - **Include Directories**: {SDK_INSTALLDIR}/include
  - **Additional Library Directories**: {SDK_INSTALLDIR}/bin/x64/release
  - **Additional Dependencies**: CADRMXLib.lib;
- Implement RMX functionalites via CADRMX Client SDK and CAD appliation SDK.
- Build and deploy RMX plugin on the target application machine.
- Deploy the {SDK_INSTALLDIR}/bin/x64/release/libeay32.dll file in the installation directory of the target application.

4.2 Using APIs to implement Rights Management features

See below examples which illustrate how to use APIs.

4.2.1 Example 1: Intialize RMX

```
#include <RMXLInc.h>
IRMXInstance* pInst;
IRMXUser* pUser;
//
// Initialize RMX instance with current logged users via RMD
//
RMXResult result = RMX_GetCurrentLoggedInUser(pInst, pUser);
if (!result) {
  std::wcout « result.GetErrMessage() « std::endl;
  return;
}
wchar_t szProcessName[MAX_PATH];
GetModuleFileName(NULL, &szProcessName[0], MAX_PATH);
//
// Register caller app as trusted app
//
bool registered;
if (pInst->IsAppRegistered(szProcessName, registered) == 0 && !registered) {
  result = pInst->RegisterApp(szProcessName);
  if (!result) {
      std::wcout « result.GetErrMessage() « std::endl;
```

```
      return;
  }
}
//
// Notify RPM driver the RMX gets started to run
//
result = pInst->NotifyRMXStatus(true);
if (!result) {
  std::wcout « result.GetErrMessage() « std::endl;
  return;
}
// ....Do other operations
//
// Uninitialize RMX instance when app or plugin terminates
//
RMX_DeleteRMXInstance(pInst);
```

4.2.2 Example 2: Handle RPM folder

```
//
// Check if specified directory is RPM folder
//
bool isRPMDir = false;
const wchar_t* dir = L"c:\\users\\rpmtest";
RMXResult result = pInst->IsRPMFolder(dir, isRPMDir);
if (!result) {
  std::wcout « result.GetErrMessage() « std::endl;
  return;
}

//
// Add new RPM folder
//
if (!isRPMDir) {
  result = pInst->AddRPMDir(dir, 1);
  if (!result) {
      std::wcout « result.GetErrMessage() « std::endl;
      return;
  }
}
```

4.2.3 Example 3: Handle file protection

```
//
// Protect file
//
std::wstring newNXLFile;
const wchar_t* filePath = L"c:\\users\\rpmtest\\HelloWorld.txt";
const char* tags = "{\"ip_classification\":[\"secret\"]}";
RMXResult result = pUser->ProtectFile(filePath, L"c:\\users\\rpmtest", tags, newNXLFile);
if (!result) {
  std::wcout « result.GetErrMessage() « std::endl;
  return;
}
//
// Evaluate file rights
//
std::vector<RMXFileRight> rights;
result = pInst->GetFileRights(newNXLFile, rights);
if (!result) {
  std::wcout « result.GetErrMessage() « std::endl;
  return;
}
//
// Check if edit operation is allowed or not
//
bool allowEdit = false;
for (const auto& r : rights) {
  if (r == RMX_RIGHT_EDIT) {
      allowEdit = true;
      break;
  }
}
```

4.3 General Notes

- APIs return the RMXResult that contains integer code and error message. The error code is 0 if the operation is successful.

- Must call RMX_DeleteRMXInstance to terminate the RMX session when the RMX plugin is unloadled.

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 7

Class Documentation

7.1 IRMXInstance Class Reference

`#include <RMXLInstance.h>`

Public Member Functions

- virtual RMXResult IsInitFinished (bool &finished)=0
- virtual RMXResult GetLoginUser (IRMXUser *&pUser)=0
- virtual RMXResult AddRPMDir (const std::wstring &dirPath, uint32_t option=1)=0
- virtual RMXResult RemoveRPMDir (const std::wstring &dirPath, bool force=true)=0
- virtual RMXResult IsRPMFolder (const std::wstring &dirPath, bool &isRPMDir, RMX_RPMFolderRelation *relation=nullptr)=0
- virtual RMXResult RegisterApp (const std::wstring &appPath)=0
- virtual RMXResult UnregisterApp (const std::wstring &appPath)=0
- virtual RMXResult IsAppRegistered (const std::wstring &appPath, bool &registered)=0
- virtual RMXResult NotifyRMXStatus (bool running)=0
- virtual RMXResult EditCopyFile (const std::wstring &nxlFilePath, std::wstring &destPath)=0
- virtual RMXResult EditSaveFile (const std::wstring &filePath, const std::wstring &originalNXLfilePath=L"", bool deleteSource=false, uint32_t exitEdit=0)=0
- virtual RMXResult RPMCopyFile (const std::wstring &existingNxlFilePath, const std::wstring &newNxlFile↩ Path, bool deleteSource=false)=0
- virtual RMXResult RPMDeleteFile (const std::wstring &filePath)=0
- virtual RMXResult GetFileRights (const std::wstring &filePath, std::vector< RMXFileRight > &rights)=0
- virtual RMXResult GetFileStatus (const std::wstring &filePath, bool &isProtected, RMX_RPMFolderRelation *dirRelation=nullptr)=0
- virtual RMXResult ReadFileTags (const std::wstring &filePath, std::string &tags)=0
- virtual RMXResult SetViewOveraly (const RMX_VIEWOVERLAY_PARAMS &params)=0
- virtual RMXResult ClearViewOverlay (void *hTargetWnd)=0

7.1.1 Detailed Description

This interface defines the RMX instance

Definition at line 20 of file RMXLInstance.h.

7.1.2 Member Function Documentation

7.1.2.1 IsInitFinished()

```
virtual RMXResult IRMXInstance::IsInitFinished (
            bool & finished )  [pure virtual]
```

Checks if the initialization of RMX instance has finished

Parameters

out	*finished*	true if intialization has finished.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

7.1.2.2 GetLoginUser()

```
virtual RMXResult IRMXInstance::GetLoginUser (
            IRMXUser *& pUser )  [pure virtual]
```

Gets current login user in the session

Parameters

out	*pUser*	Pointer to IRMXUser.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

7.1.2.3 AddRPMDir()

```
virtual RMXResult IRMXInstance::AddRPMDir (
            const std::wstring & dirPath,
            uint32_t option = 1 )  [pure virtual]
```

Adds RPM secure folder

Parameters

in	*dirPath*	Directory full path.
in	*option*	0 When copying a NXL file to this RPM folder, do not allow to overwrite unprotected file if exists 1 Set OVERWRITE_NORMAL flag on the new RPM folder, to allow overwriting unprotected file.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

7.1.2.4 RemoveRPMDir()

```
virtual RMXResult IRMXInstance::RemoveRPMDir (
            const std::wstring & dirPath,
            bool force = true )  [pure virtual]
```

Removes RPM secure folder

Parameters

in	*dirPath*	Directory path.
in	*force*	True to force to remove even though the trusted process is still running. Otherwise, remove will fail.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

7.1.2.5 IsRPMFolder()

```
virtual RMXResult IRMXInstance::IsRPMFolder (
            const std::wstring & dirPath,
            bool & isRPMDir,
            RMX_RPMFolderRelation * relation = nullptr )  [pure virtual]
```

Checks if the specified directory is a RPM secure folder

Parameters

in	*dirPath*	Directory path.
out	*isRPMDir*	True if it is RPM folder
out	*relation*	RMX_RPMFolderRelation type

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

See also

RMX_RPMFolderRelation

7.1.2.6 RegisterApp()

```
virtual RMXResult IRMXInstance::RegisterApp (
            const std::wstring & appPath )  [pure virtual]
```

Registers an application as rpm trusted app

Parameters

in	*appPath*	Full path of an executable image.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

7.1.2.7 UnregisterApp()

```
virtual RMXResult IRMXInstance::UnregisterApp (
            const std::wstring & appPath )  [pure virtual]
```

Unregisters an application as rpm trusted app

Parameters

in	*appPath*	Full path of an executable image.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

7.1.2.8 IsAppRegistered()

```
virtual RMXResult IRMXInstance::IsAppRegistered (
            const std::wstring & appPath,
            bool & registered )  [pure virtual]
```

Checks if an application is registered as rpm trusted application

Parameters

in	*appPath*	Full path of an executable image.
out	*registered*	True if the executable image is registered as rpm trusted application.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

7.1.2.9 NotifyRMXStatus()

```
virtual RMXResult IRMXInstance::NotifyRMXStatus (
            bool running )  [pure virtual]
```

Notifies RMX statux in the caller process to RPM driver

Precondition

RegisterApp is called to register caller application as RPM trusted application.

Parameters

in	*running*	True to notify RMX starts to run. False to notify RMX gets stopped.

Returns

RMXResult::GetCode() returns 0 if succeeded. Otherwise, returns error code.

7.1.2.10 EditCopyFile()

```
virtual RMXResult IRMXInstance::EditCopyFile (
            const std::wstring & nxlFilePath,
            std::wstring & destPath )  [pure virtual]
```

Copies NXL file to an RPM folder for editing with native file name.

Parameters

in	*nxlFilePath*	File path of the NXL file to be edited.
in,out	*destPath*	As input, specify an RPM folder path. If empty, use RPM secret directory. As output, return the generated NXL file path without .nxl extension.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

Note

EditCopyFile is used to copy NXL file into RPM folder, so that native application can edit protected file transparently.

7.1.2.11 EditSaveFile()

```
virtual RMXResult IRMXInstance::EditSaveFile (
            const std::wstring & filePath,
            const std::wstring & originalNXLfilePath = L"",
            bool deleteSource = false,
            uint32_t exitEdit = 0 )  [pure virtual]
```

Re-protects the protected file in RPM folder back to original NXL file

Parameters

in	*filePath*	Native file path to be updated.
in	*originalNXLfilePath*	File path of original NXL file. If empty, search local mapping or check current folder.
in	*deleteSource*	True to delete original NXL file.
in	*exitEdit*	0: Not exit and save 1: Not exit but save 2: Exit but not save 3: Exit and save

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

Note

EditSaveFile is used to re-protect the protected file to have recent modification.

7.1.2.12 RPMCopyFile()

```
virtual RMXResult IRMXInstance::RPMCopyFile (
            const std::wstring & existingNxlFilePath,
            const std::wstring & newNxlFilePath,
            bool deleteSource = false )  [pure virtual]
```

Copies Nxl file in RPM folder

Parameters

in	*existingNxlFilePath*	The name of an existing NXL file to be copied.
in	*newNxlFilePath*	The name of the new NXL file. If the specified new file already exists, the function overwrites the existing file and succeeds.
in	*deleteSource*	True to delete the source NXL file.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

Note

RPMCopyFile is used to copy nxl file by trusted process in RPM folder. By default, using Windows API to copy file by trusted process will copy the corresponding plain file.

7.1.2.13 RPMDeleteFile()

```
virtual RMXResult IRMXInstance::RPMDeleteFile (
            const std::wstring & filePath )  [pure virtual]
```

Deletes NXL file in RPM folder

Parameters

in	*filePath*	File path to be deleted.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

\note RPMDeleteFile must be called to let RPM driver delete NXL file in RPM folder clearly, including cache.

7.1.2.14 GetFileRights()

```
virtual RMXResult IRMXInstance::GetFileRights (
            const std::wstring & filePath,
            std::vector< RMXFileRight > & rights )  [pure virtual]
```

Gets rights for the specified file

Parameters

in	*filePath*	File path to be evaluated.
out	*rights*	Vector of RMXFileRight.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

See also

RMXFileRight

7.1.2.15 GetFileStatus()

```
virtual RMXResult IRMXInstance::GetFileStatus (
            const std::wstring & filePath,
            bool & isProtected,
            RMX_RPMFolderRelation * dirRelation = nullptr )  [pure virtual]
```

Gets status for the specified file, such as if it is in RPM folder, and if it is protected.

Parameters

in	*filePath*	File path to be evaluated.
out	*isProtected*	True if file is protected.
out	*dirRelation*	RMX_RPMFolderRelation type.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

See also

RMX_RPMFolderRelation

7.1.2.16 ReadFileTags()

```
virtual RMXResult IRMXInstance::ReadFileTags (
            const std::wstring & filePath,
            std::string & tags )  [pure virtual]
```

Gets tags for the specified file

Parameters

in	*filePath*	File path to be evaluated.
out	*tags*	File tags(json-format string), such as {"ip_classification":["secret"],"gov_classification:["test"]}

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

7.1.2.17 SetViewOveraly()

```
virtual RMXResult IRMXInstance::SetViewOveraly (
            const RMX_VIEWOVERLAY_PARAMS & params )  [pure virtual]
```

Create a view overlay if dynamic watermark can be found for specified tags parameters.

Parameters

in	*params*	RMX_VIEWOVERLAY_PARAMS object, containing all below parameters: • hTargetWnd Top window handler to show view overlay. • vecTags Vector of tags to be evaluated for dynamic watermark. • vecCtxAttrs Vector of context attributes to be evaluated for dynamic watermark, like appPath, fileName, username, etc. • displayOffsets Integer array to specify the offset effect in overlay display area, with {left,upper,right,bottom} sequence.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

Note

Call ClearViewOverlay to remove view overlay

See also

RMX_VIEWOVERLAY_PARAMS

ClearViewOverlay

7.1.2.18 ClearViewOverlay()

```
virtual RMXResult IRMXInstance::ClearViewOverlay (
            void * hTargetWnd )  [pure virtual]
```

Removes view overlay

Parameters

in	*hTargetWnd*	Top window handler to show view overlay

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

The documentation for this class was generated from the following file:

- RMXLInstance.h

7.2 IRMXUser Class Reference

```
#include <RMXLUser.h>
```

Public Member Functions

- virtual const std::wstring GetName ()=0
- virtual const std::wstring GetEmail ()=0
- virtual uint32_t GetUserID ()=0
- virtual RMXResult ProtectFile (const std::wstring &filePath, const std::wstring &targetDir, const std::string &tags, std::wstring &newNxlFile)=0
- virtual RMXResult GetResourceRightsFromCentralPolicies (const std::wstring &resourceName, const std::wstring &resourceType, const std::vector< RMX_EVAL_ATTRIBUATE > &attrs, std::vector< RMX_CENTRAL_RIGHT > &centralRights)=0
- virtual bool GetDefaultPolicyBundle (std::string &policyBundle)=0
- virtual const std::string GetSystemProjectTenantId ()=0
- virtual const std::string GetDefaultTokenGroupName ()=0
- virtual RMXResult MergeTags (const std::string &unionTags, std::string &mergedTags)=0

7.2.1 Detailed Description

This interface provides information about a RMX user.

Definition at line 17 of file RMXLUser.h.

7.2.2 Member Function Documentation

7.2.2.1 GetName()

```
virtual const std::wstring IRMXUser::GetName ( )  [pure virtual]
```

Gets current login user name

Returns

User name string.

7.2.2.2 GetEmail()

```
virtual const std::wstring IRMXUser::GetEmail ( )  [pure virtual]
```

Gets current login user email address

Returns

User email string.

7.2.2.3 GetUserID()

```
virtual uint32_t IRMXUser::GetUserID ( )  [pure virtual]
```

Get current login user id.

Returns

User id.

7.2.2.4 ProtectFile()

```
virtual RMXResult IRMXUser::ProtectFile (
            const std::wstring & filePath,
            const std::wstring & targetDir,
            const std::string & tags,
            std::wstring & newNxlFile )  [pure virtual]
```

Protects a local file

Parameters

in	*filePath*	File path to be protected.
in	*targetDir*	Target directory path to place new protected file.
in	*tags*	Document classification metadata.
out	*newNxlFile*	File path of NXL file to be created.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

7.2.2.5 GetResourceRightsFromCentralPolicies()

```
virtual RMXResult IRMXUser::GetResourceRightsFromCentralPolicies (
            const std::wstring & resourceName,
            const std::wstring & resourceType,
            const std::vector< RMX_EVAL_ATTRIBUATE > & attrs,
            std::vector< RMX_CENTRAL_RIGHT > & centralRights )  [pure virtual]
```

Gets the resource rights with their associated obligations from central policies

Parameters

in	*resourceName*	Name of resource
in	*resourceType*	Type of resource (e.g. "fso", "portal")
in	*attrs*	Resource attributes to be evaluated
out	*centralRights*	Rights assigned to the resource and their associated obligations to be returned.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

7.2.2.6 GetDefaultPolicyBundle()

```
virtual bool IRMXUser::GetDefaultPolicyBundle (
            std::string & policyBundle )  [pure virtual]
```

Gets policy boudle for default tenant

Parameters

out	*policyBundle*	Policy bundle to be returned.

Returns

True if success, else return false.

7.2.2.7 GetSystemProjectTenantId()

```
virtual const std::string IRMXUser::GetSystemProjectTenantId ( )  [pure virtual]
```

Gets system project tenantId

Returns

System project tenantId

7.2.2.8 GetDefaultTokenGroupName()

```
virtual const std::string IRMXUser::GetDefaultTokenGroupName ( )  [pure virtual]
```

Gets token group name of default tenant

Returns

Token group name of default tenant

7.2.2.9 MergeTags()

```
virtual RMXResult IRMXUser::MergeTags (
            const std::string & unionTags,
            std::string & mergedTags )  [pure virtual]
```

Merges tags in case tags come from multiple source NXL files, according to the priority obligation defined in central policy.

Parameters

in	*unionTags*	Multiple tags strings grouped by "{}", such as {"ip_classification":["secret"]}{"ip_classification":["supper-securet"]}.
out	*mergedTags*	Single group of tags to be returned, such as {"ip_classification":["supper-securet"]}.

Note

The algorithm to merge tags groups are:

- Duplicated value should be removed for same key.
- Different values are combined as multiples values for same key.
- Lower priorit values will be removed, and leave higher one as single value for same key, if priority obligation is found in central policy.

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

The documentation for this class was generated from the following file:

- RMXLUser.h

7.3 RMX_CENTRAL_RIGHT Struct Reference

`#include <RMXLTypeDef.h>`

Public Attributes

- RMXFileRight right
- std::vector< RMX_OBLIGATION_INFO > obligations

7.3.1 Detailed Description

Structure to store a central right and obligations

Definition at line 85 of file RMXLTypeDef.h.

7.3.2 Member Data Documentation

7.3.2.1 right

`RMXFileRight RMX_CENTRAL_RIGHT::right`

File right

Definition at line 87 of file RMXLTypeDef.h.

7.3.2.2 obligations

`std::vector<RMX_OBLIGATION_INFO> RMX_CENTRAL_RIGHT::obligations`

List of obligations

Definition at line 88 of file RMXLTypeDef.h.

The documentation for this struct was generated from the following file:

- RMXLTypeDef.h

7.4 RMX_EVAL_ATTRIBUATE Struct Reference

`#include <RMXLTypeDef.h>`

Public Attributes

- std::wstring key
- std::wstring value

7.4.1 Detailed Description

Structure to store an attribute to be evaluated

Definition at line 56 of file RMXLTypeDef.h.

7.4.2 Member Data Documentation

7.4.2.1 key

`std::wstring RMX_EVAL_ATTRIBUATE::key`

Attribute name

Definition at line 58 of file RMXLTypeDef.h.

7.4.2.2 value

`std::wstring RMX_EVAL_ATTRIBUATE::value`

Attribute value

Definition at line 59 of file RMXLTypeDef.h.

The documentation for this struct was generated from the following file:

- RMXLTypeDef.h

7.5 RMX_OBLIGATION_INFO Struct Reference

`#include <RMXLTypeDef.h>`

Public Attributes

- std::wstring name
- std::vector< RMX_EVAL_ATTRIBUATE > options

7.5.1 Detailed Description

Structure to store obligation information defined in central policy

Definition at line 77 of file RMXLTypeDef.h.

7.5.2 Member Data Documentation

7.5.2.1 name

```
std::wstring RMX_OBLIGATION_INFO::name
```

Name of obligation name

Definition at line 78 of file RMXLTypeDef.h.

7.5.2.2 options

```
std::vector<RMX_EVAL_ATTRIBUATE> RMX_OBLIGATION_INFO::options
```

Attributes of obligation

Definition at line 79 of file RMXLTypeDef.h.

The documentation for this struct was generated from the following file:

- RMXLTypeDef.h

7.6 RMX_VIEWOVERLAY_PARAMS Struct Reference

```
#include <RMXLTypeDef.h>
```

Public Attributes

- void * hTargetWnd
- std::vector< std::string > vecTags
- std::vector< RMX_EVAL_ATTRIBUATE > vecCtxAttrs
- int displayOffsets [4]

7.6.1 Detailed Description

Structure to store parameters required to set view overlay

Definition at line 66 of file RMXLTypeDef.h.

7.6.2 Member Data Documentation

7.6.2.1 hTargetWnd

```
void* RMX_VIEWOVERLAY_PARAMS::hTargetWnd
```

Top window handler to show view overlay

Definition at line 68 of file RMXLTypeDef.h.

7.6.2.2 vecTags

```
std::vector<std::string> RMX_VIEWOVERLAY_PARAMS::vecTags
```

Vector of tags to be evaluated for dynamic watermark

Definition at line 69 of file RMXLTypeDef.h.

7.6.2.3 vecCtxAttrs

```
std::vector<RMX_EVAL_ATTRIBUATE> RMX_VIEWOVERLAY_PARAMS::vecCtxAttrs
```

Vector of context attributes to be evaluated for dynamic watermark, like appPath, fileName, username, etc

Definition at line 70 of file RMXLTypeDef.h.

7.6.2.4 displayOffsets

```
int RMX_VIEWOVERLAY_PARAMS::displayOffsets[4]
```

Integer array to specify the offset effect in overlay display area, with {left,upper,right,bottom} sequence

Definition at line 71 of file RMXLTypeDef.h.

The documentation for this struct was generated from the following file:

- RMXLTypeDef.h

7.7 RMXResult Class Reference

`#include <RMXLResult.h>`

Public Member Functions

- RMXResult ()
- RMXResult (RMXErrCode_t code, const std::wstring &errMessage)
- virtual ~RMXResult ()
- RMXResult (const RMXResult &other)
- RMXResult & operator= (const RMXResult &other)
- operator bool () const
- bool operator== (int code) const
- bool operator!= (int code) const
- RMXErrCode_t GetErrCode () const
- std::wstring GetErrMessage () const

Protected Attributes

- RMXErrCode_t m_code
- std::wstring m_message

7.7.1 Detailed Description

This class provides information about the result of API call.

Definition at line 20 of file RMXLResult.h.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 RMXResult() [1/3]

```
RMXResult::RMXResult ( )  [inline]
```

Default constructor

Definition at line 24 of file RMXLResult.h.

```
24 : m_code(0) {};
```

7.7.2.2 RMXResult() [2/3]

```
RMXResult::RMXResult (
            RMXErrCode_t code,
            const std::wstring & errMessage )  [inline], [explicit]
```

Constructor

Parameters

in	*code*	0 if succeeded. Otherwise, error code.
in	*errMessage*	Error message if failed.

Definition at line 33 of file RMXLResult.h.

```
34          : m_code(code), m_message(errMessage)
35      {
36      }
```

7.7.2.3 ~RMXResult()

```
virtual RMXResult::~RMXResult ( )  [inline], [virtual]
```

Destructor

Definition at line 38 of file RMXLResult.h.

```
38 {}
```

7.7.2.4 RMXResult() [3/3]

```
RMXResult::RMXResult (
            const RMXResult & other )  [inline]
```

Copy constructor

Definition at line 45 of file RMXLResult.h.

```
46          : m_code(other.m_code), m_message(other.m_message)
47      {
48      }
```

7.7.3 Member Function Documentation

7.7.3.1 operator=()

```
RMXResult& RMXResult::operator= (
            const RMXResult & other )  [inline]
```

Assignment operator

Definition at line 55 of file RMXLResult.h.

```
56      {
57          if (this != &other)
58          {
59              m_code = other.m_code;
60              m_message = other.m_message;
61          }
62
63          return *this;
64      }
```

7.7.3.2 operator bool()

```
RMXResult::operator bool ( ) const  [inline]
```

Operator to check if the result indicates failure with error

Returns

True if no error occurs.

Definition at line 72 of file RMXLResult.h.

```
72 { return (0 == m_code); }
```

7.7.3.3 operator==()

```
bool RMXResult::operator== (
            int code ) const  [inline]
```

Operator to check if they are same error

Parameters

code	The error code.

Returns

True if error code is same.

Definition at line 82 of file RMXLResult.h.

```
82 { return (code == m_code); }
```

7.7.3.4 operator"!=()

```
bool RMXResult::operator!= (
            int code ) const  [inline]
```

Operator to check if they are same error

Parameters

code	The code.

Returns

False if error code is same.

Definition at line 91 of file RMXLResult.h.

```
91 { return (code != m_code); }
```

7.7.3.5 GetErrCode()

```
RMXErrCode_t RMXResult::GetErrCode ( ) const  [inline]
```

Gets error code

Returns

The error code.

Definition at line 99 of file RMXLResult.h.

```
99 { return m_code; }
```

7.7.3.6 GetErrMessage()

```
std::wstring RMXResult::GetErrMessage ( ) const  [inline]
```

Gets error message

Returns

The error message.

Definition at line 107 of file RMXLResult.h.

```
107 { return m_message; }
```

7.7.4 Member Data Documentation

7.7.4.1 m_code

```
RMXErrCode_t RMXResult::m_code  [protected]
```

The code

Definition at line 111 of file RMXLResult.h.

7.7.4.2 m_message

```
std::wstring RMXResult::m_message  [protected]
```

The message

Definition at line 113 of file RMXLResult.h.

The documentation for this class was generated from the following file:

- RMXLResult.h

Chapter 8

File Documentation

8.1 RMXLGlobal.h File Reference

```
#include <wtypes.h>
#include "RMXLResult.h"
#include "RMXLTypeDef.h"
```

Functions

- CADRMX_API DWORD RMX_GetLibVersion (void)
- CADRMX_API DWORD RMX_GetSDWLibVersion (void)
- CADRMX_API RMXResult RMX_GetCurrentLoggedInUser (IRMXInstance *&pInstance, IRMXUser *&p↩ User)
- CADRMX_API void RMX_DeleteRMXInstance (IRMXInstance *pInstance)

8.1.1 Detailed Description

This head file contains all gloal functions

8.1.2 Function Documentation

8.1.2.1 RMX_GetLibVersion()

```
CADRMX_API DWORD RMX_GetLibVersion (
            void  )
```

Returns the version number of the CADRMX Client SDK

Returns

The format of version is AABBCCCC major.minor.build.

8.1.2.2 RMX_GetSDWLibVersion()

```
CADRMX_API DWORD RMX_GetSDWLibVersion (
            void  )
```

Returns the version number of the SkyDRM Client SDK

Returns

The format of version is AABBCCCC major.minor.build.

8.1.2.3 RMX_GetCurrentLoggedInUser()

```
CADRMX_API RMXResult RMX_GetCurrentLoggedInUser (
            IRMXInstance *& pInstance,
            IRMXUser *& pUser )
```

Gets current login user

Parameters

out	*pInstance*	IRMXInstance pointer.
out	*pUser*	IRMXUser pointer

Returns

RMXResult object. Either call RMXResult::GetCode() to check error code, or call bool() to check if error code is 0.

Note

Call RMX_DeleteRMXInstance to delete instance.

8.1.2.4 RMX_DeleteRMXInstance()

```
CADRMX_API void RMX_DeleteRMXInstance (
            IRMXInstance * pInstance )
```

Deletes an IRMXInstance instance

Precondition

RMX_GetCurrentLoggedInUser is called to get the IRMXInstance instance.

Parameters

in	*pInstance*	IRMXInstance pointer.

Returns

void

8.2 RMXLInc.h File Reference

```
#include "RMXLInstance.h"
#include "RMXLGlobal.h"
#include "RMXLUser.h"
```

8.2.1 Detailed Description

This header file includes all headers of SDK.

8.3 RMXLInstance.h File Reference

```
#include "RMXLResult.h"
#include "RMXLTypeDef.h"
```

Classes

- class IRMXInstance

8.4 RMXLResult.h File Reference

```
#include <string>
#include "RMXLTypeDef.h"
```

Classes

- class RMXResult

Typedefs

- typedef unsigned long RMXErrCode_t

8.4.1 Typedef Documentation

8.4.1.1 RMXErrCode_t

```
typedef unsigned long RMXErrCode_t
```

Error code

Definition at line 12 of file RMXLResult.h.

8.5 RMXLTypeDef.h File Reference

```
#include <stdint.h>
#include <string>
#include <vector>
```

Classes

- struct RMX_EVAL_ATTRIBUATE
- struct RMX_VIEWOVERLAY_PARAMS
- struct RMX_OBLIGATION_INFO
- struct RMX_CENTRAL_RIGHT

Macros

- #define **CADRMX_API** __declspec(dllimport)

Enumerations

- enum RMX_RPMFolderRelation { RMX_NonRPMFolder, RMX_RPMFolder, RMX_RPMAncestralFolder, RMX_RPMInheritedFolder }
- enum RMXFileRight {
  RMX_RIGHT_VIEW = 1, RMX_RIGHT_EDIT = 2, RMX_RIGHT_PRINT = 4, RMX_RIGHT_CLIPBOARD = 8, RMX_RIGHT_SAVEAS = 0x10, RMX_RIGHT_DECRYPT = 0x20, RMX_RIGHT_SCREENCAPTURE = 0x40, RMX_RIGHT_SEND = 0x80,
  RMX_RIGHT_CLASSIFY = 0x100, RMX_RIGHT_SHARE = 0x200, RMX_RIGHT_DOWNLOAD = 0x400, RMX_RIGHT_WATERMARK = 0x40000000 }

8.5.1 Detailed Description

Declarations used throughout the CADRMX Client SDK.

8.5.2 Enumeration Type Documentation

8.5.2.1 RMX_RPMFolderRelation

```
enum RMX_RPMFolderRelation
```

RPM folder relation type

Enumerator

RMX_NonRPMFolder	Not RPM folder
RMX_RPMFolder	RPM folder
RMX_RPMAncestralFolder	Ancestor of RPM folder
RMX_RPMInheritedFolder	Descendant of PRM folder

Definition at line 26 of file RMXLTypeDef.h.

```
27 {
28     RMX_NonRPMFolder,
29     RMX_RPMFolder,
30     RMX_RPMAncestralFolder,
31     RMX_RPMInheritedFolder,
32 } RMX_RPMFolderRelation;
```

8.5.2.2 RMXFileRight

```
enum RMXFileRight
```

RMX file right type

Enumerator

RMX_RIGHT_VIEW	Right to view a protected file
RMX_RIGHT_EDIT	Right to edit a protected file
RMX_RIGHT_PRINT	Right to print a protected file
RMX_RIGHT_CLIPBOARD	Right to copy content of a protected file to clipboard
RMX_RIGHT_SAVEAS	Right to save copy of a protected file
RMX_RIGHT_DECRYPT	Right to remove protection
RMX_RIGHT_SCREENCAPTURE	Right to capture screen
RMX_RIGHT_SEND	Right to send a protected file
RMX_RIGHT_CLASSIFY	Right to reclassify a protected file
RMX_RIGHT_SHARE	Right to share a protected file
RMX_RIGHT_DOWNLOAD	Right to download a protected file
RMX_RIGHT_WATERMARK	Right to display a watermark on a protected file

Definition at line 37 of file RMXLTypeDef.h.

```
38 {
39     RMX_RIGHT_VIEW = 1,
40     RMX_RIGHT_EDIT = 2,
41     RMX_RIGHT_PRINT = 4,
42     RMX_RIGHT_CLIPBOARD = 8,
43     RMX_RIGHT_SAVEAS = 0x10,
44     RMX_RIGHT_DECRYPT = 0x20,
45     RMX_RIGHT_SCREENCAPTURE = 0x40,
46     RMX_RIGHT_SEND = 0x80,
47     RMX_RIGHT_CLASSIFY = 0x100,
48     RMX_RIGHT_SHARE = 0x200,
49     RMX_RIGHT_DOWNLOAD = 0x400,
50     RMX_RIGHT_WATERMARK = 0x40000000
51 } RMXFileRight;
```

8.6 RMXLUser.h File Reference

```
#include "RMXLResult.h"
#include "RMXLTypeDef.h"
```

Classes

- class IRMXUser

Index