NEXTLABS®

NextLabs CADRMX Client SDK Documentation

2020.08

Confidentiality Notice

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

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 third-party module that uses CADRMX Client SDK to extend Rights Management functionality.
Rights Protection Manager (RPM)	A Windows driver running in the kernel to handle Rights Management transparently.
Rights Management Desktop (RMD)	An Windows desktop application for Rights Management.
RPM Folder	A secure folder to let native applications handle NXL file transparently.
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 2020.08
- NextLabs SkyDRM Rights Management Server 2020.08
- NextLabs Control Center 9.1

3.3 Development Environment

- Microsoft Visual Studio 2015 with Update 3
- CADRMX Client SDK 2020.08

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 libeay32.dll, CADRMXLib.dll and log4cplus.properties provided in SDK to the same directory of RMX plugin in 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;
  }
}
```

4.2.3 Example 3: Protect file

```
//
// Protect file with the specified tags
//
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;
}
```

4.2.4 Example 4: Evaluate right

```
//
// Check if edit permission is granted or not
//
bool allowEdit = false;
const wchar_t* filePath = L"c:\\users\\rpmtest\\HelloWorld.txt";
RMXResult result = pInst->CheckFileRight(filePath, RMX_RIGHT_EDIT, allowEdit);
if (!result) {
  std::wcout « result.GetErrMessage() « std::endl;
}
```

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

What's New

5.1 2020.08 Release

5.1.1 New API

- IRMXInstance::CheckFileRight
- IRMXInstance::RPMGetFileInfo
- IRMXInstance::RPMSetFileAttributes

5.1.2 Changed API

- IRMXInstance::AddRPMDir

Chapter 6

Module Documentation

6.1 Global Functions

Functions

- DWORD RMX_GetLibVersion (void)
- DWORD RMX_GetSDWLibVersion (void)
- RMXResult RMX_GetCurrentLoggedInUser (IRMXInstance *&pInstance, IRMXUser *&pUser)
- void RMX_DeleteRMXInstance (IRMXInstance *pInstance)

6.1.1 Detailed Description

6.1.2 Function Documentation

6.1.2.1 RMX_GetLibVersion()

```
DWORD RMX_GetLibVersion (
            void  )
```

Returns the version number of the CADRMX Client SDK

Returns

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

6.1.2.2 RMX_GetSDWLibVersion()

```
DWORD RMX_GetSDWLibVersion (
            void  )
```

Returns the version number of the SkyDRM Client SDK

Returns

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

6.1.2.3 RMX_GetCurrentLoggedInUser()

```
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.

6.1.2.4 RMX_DeleteRMXInstance()

```
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

Chapter 7

Class Documentation

7.1 IRMXInstance Class Reference

Public Member Functions

- virtual RMXResult IsInitFinished (bool &finished)=0
- virtual RMXResult AddRPMDir (const std::wstring &dirPath, uint32_t option=(RMXRPMFolderOption::RMX_RPMFOLDER_OVE
- 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 AddTrustedApp (const std::wstring &appPath)=0
- virtual RMXResult RemoveTrustedApp (const std::wstring &appPath)=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 CheckFileRight (const std::wstring &filePath, RMXFileRight right, bool &allowed, bool audit=false)=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 RPMGetFileInfo (const std::wstring &filepath, std::string &duid, std::string &tags, std::string &tokengroup, std::string &creatorid, std::string &infoext, DWORD &attributes, RMX_RPMFolderRelation &dirRelation, BOOL &isNXLFile)=0
- virtual RMXResult RPMSetFileAttributes (const std::wstring &nxlFilePath, DWORD dwFileAttributes)=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 22 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 AddRPMDir()

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

Adds RPM secure folder

Parameters

in	*dirPath*	Directory full path.
in	*option*	RMXRPMFolderOption. Defaults to RMX_RPMFOLDER_OVERWRITE|RMX_RPMFOLDER_API

Returns

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

See also

RMXRPMFolderOption

7.1.2.3 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. Defaults to true.

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 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

7.1.2.5 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.

7.1.2.6 UnregisterApp()

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

Unregisters an application as rpm trusted app

Precondition

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

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 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.

7.1.2.8 NotifyRMXStatus()

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

Notifies RPM driver to update the trusted status of caller process

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.

7.1.2.9 AddTrustedApp()

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

Adds an application to the non-persistent trusted application list

Precondition

The caller process is a trusted process.

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.

7.1.2.10 RemoveTrustedApp()

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

Removes an application from the non-persistent trusted application list

Precondition

The caller process is a trusted process.

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.

7.1.2.11 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.

7.1.2.12 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. Defaults to false.
in	*exitEdit*	Defaults to 0. 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.

7.1.2.13 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. Defaults to false.

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.

7.1.2.14 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.

7.1.2.15 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

7.1.2.16 CheckFileRight()

```
virtual RMXResult IRMXInstance::CheckFileRight (
            const std::wstring & filePath,
            RMXFileRight right,
            bool & allowed,
            bool audit = false )  [pure virtual]
```

Checks if the specified rights are granted to the file

Parameters

in	*filePath*	File path to be evaluated.
in	*right*	RMXFileRight to be checked.
out	*allowed*	True if the rights are granted.
in	*audit*	True to add activity log into SkyDRM. Defaults to false. For auditing, either call IRMXUser::AddActivityLog on demand or set this parameter to true when checking the rights.

Returns

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

Since

2020.08

See also

RMXFileRight

IRMXUser::AddActivityLog

7.1.2.17 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

7.1.2.18 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.

7.1.2.19 RPMGetFileInfo()

```
virtual RMXResult IRMXInstance::RPMGetFileInfo (
            const std::wstring & filepath,
            std::string & duid,
            std::string & tags,
            std::string & tokengroup,
            std::string & creatorid,
            std::string & infoext,
            DWORD & attributes,
            RMX_RPMFolderRelation & dirRelation,
            BOOL & isNXLFile )  [pure virtual]
```

Gets information for the specified file

Parameters

in	*filePath*	File path to be evaluated.
out	*duid*	DUID of NXL file.
out	*tags*	File tags(json-format string), such as {"ip_classification":["secret"],"gov_classification:["test"]}
out	*tokengroup*	Token group name.
out	*creatorid*	owner Id.
out	*infoext*	File information.
out	*attributes*	File system attributes. Refer to Win32 API GetFileAttributes for more information.
out	*relation*	RMX_RPMFolderRelation type.
out	*isNXLFile*	True if file is protected.

Returns

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

Since

2020.08

See also

RMX_RPMFolderRelation

7.1.2.20 RPMSetFileAttributes()

```
virtual RMXResult IRMXInstance::RPMSetFileAttributes (
            const std::wstring & nxlFilePath,
            DWORD dwFileAttributes )  [pure virtual]
```

Sets the attributes for a NXL file

Parameters

in	*nxlFilePath*	The name of the file whose attributes are to be set.
in	*dwFileAttributes*	The file attributes to set for the file. This parameter can be one or more values, the values same like Win32 API SetFileAttributes: FILE_ATTRIBUTE_ARCHIVE 32 (0x20) FILE_ATTRIBUTE_HIDDEN 2 (0x2) FILE_ATTRIBUTE_NORMAL 128 (0x80) FILE_ATTRIBUTE_NOT_CONTENT_INDEXED 8192 (0x2000) FILE_ATTRIBUTE_OFFLINE 4096 (0x1000) FILE_ATTRIBUTE_READONLY 1 (0x1) FILE_ATTRIBUTE_SYSTEM 4 (0x4) FILE_ATTRIBUTE_TEMPORARY 256 (0x100)

Returns

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

Since

2020.08

See also

RPMGetFileInfo

7.1.2.21 SetViewOveraly()

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

Creates 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

7.1.2.22 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.

7.2 IRMXUser Class Reference

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
- virtual RMXResult AddActivityLog (const std::wstring &nxlFilePath, RMXActivityLogOperation op, RMXActivityLogResult result)=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.

7.2.2.10 AddActivityLog()

```
virtual RMXResult IRMXUser::AddActivityLog (
            const std::wstring & nxlFilePath,
            RMXActivityLogOperation op,
            RMXActivityLogResult result )  [pure virtual]
```

Adds file activity log

Parameters

in	*nxlFilePath*	File path of the NXL file.
in	*op*	Activity operation type.
in	*result*	Operation result such as denied or allowed

Returns

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

7.3 RMX_CENTRAL_RIGHT Struct Reference

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 126 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 128 of file RMXLTypeDef.h.

7.3.2.2 obligations

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

List of obligations

Definition at line 129 of file RMXLTypeDef.h.

7.4 RMX_EVAL_ATTRIBUATE Struct Reference

Public Attributes

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

7.4.1 Detailed Description

Structure to store an attribute to be evaluated

Definition at line 97 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 99 of file RMXLTypeDef.h.

7.4.2.2 value

```
std::wstring RMX_EVAL_ATTRIBUATE::value
```

Attribute value

Definition at line 100 of file RMXLTypeDef.h.

7.5 RMX_OBLIGATION_INFO Struct Reference

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 118 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 119 of file RMXLTypeDef.h.

7.5.2.2 options

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

Attributes of obligation

Definition at line 120 of file RMXLTypeDef.h.

7.6 RMX_VIEWOVERLAY_PARAMS Struct Reference

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 107 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 109 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 110 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 111 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 112 of file RMXLTypeDef.h.

7.7 RMXResult Class Reference

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.

```
00024 : 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.

```
00034         : m_code(code), m_message(errMessage)
00035     {
00036     }
```

7.7.2.3 ~RMXResult()

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

Destructor

Definition at line 38 of file RMXLResult.h.

```
00038 {}
```

7.7.2.4 RMXResult() [3/3]

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

Copy constructor

Definition at line 45 of file RMXLResult.h.

```
00046         : m_code(other.m_code), m_message(other.m_message)
00047     {
00048     }
```

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.

```
00056     {
00057         if (this != &other)
00058         {
00059             m_code = other.m_code;
00060             m_message = other.m_message;
00061         }
00062
00063         return *this;
00064     }
```

References m_code, and m_message.

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.

```
00072 { return (0 == m_code); }
```

References 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.

```
00082 { return (code == m_code); }
```

References 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.

```
00091 { return (code != m_code); }
```

References 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.

```
00099 { return m_code; }
```

References 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.

```
00107 { return m_message; }
```

References 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.

Referenced by GetErrCode(), operator bool(), operator!=(), operator=(), and operator==().

7.7.4.2 m_message

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

The message

Definition at line 113 of file RMXLResult.h.

Referenced by GetErrMessage(), and operator=().

Chapter 8

File Documentation

8.1 RMXLGlobal.h File Reference

Functions

- DWORD RMX_GetLibVersion (void)
- DWORD RMX_GetSDWLibVersion (void)
- RMXResult RMX_GetCurrentLoggedInUser (IRMXInstance *&pInstance, IRMXUser *&pUser)
- void RMX_DeleteRMXInstance (IRMXInstance *pInstance)

8.1.1 Detailed Description

This head file contains all gloal functions

Definition in file RMXLGlobal.h.

8.2 RMXLGlobal.h

```
00001 //
00002 // This is a part of the NextLabs CADRMX Client SDK.
00003 // Copyright (C) 2020 NextLabs, Inc. All Rights Reserved.
00004 //
00005
00006
00013 #pragma once
00014
00015 #include <wtypes.h>
00016
00017 #include "RMXLResult.h"
00018 #include "RMXLTypeDef.h"
00019
00020 class IRMXInstance;
00021 class IRMXUser;
00022
00033 CADRMX_API DWORD RMX_GetLibVersion(void);
00034
00041 CADRMX_API DWORD RMX_GetSDWLibVersion(void);
00042
00054 CADRMX_API RMXResult RMX_GetCurrentLoggedInUser(IRMXInstance*& pInstance, IRMXUser*& pUser);
00055
00068 CADRMX_API void RMX_DeleteRMXInstance(IRMXInstance* pInstance);
```

8.3 RMXLInc.h File Reference

8.3.1 Detailed Description

This header file includes all headers of SDK.

Definition in file RMXLInc.h.

8.4 RMXLInc.h

```
00001 //
00002 // This is a part of the NextLabs CADRMX Client SDK.
00003 // Copyright (C) 2020 NextLabs, Inc. All Rights Reserved.
00004 //
00005
00012 #pragma once
00013
00014 #include "RMXLInstance.h"
00015 #include "RMXLGlobal.h"
00016 #include "RMXLUser.h"
```

8.5 RMXLInstance.h File Reference

Classes

- class IRMXInstance

8.6 RMXLInstance.h

```
00001 //
00002 // This is a part of the NextLabs CADRMX Client SDK.
00003 //
00004 // Copyright (C) 2020 NextLabs, Inc. All Rights Reserved.
00005 //
00006 #pragma once
00007
00008 #include <wtypes.h>
00009
00010 #include "RMXLResult.h"
00011 #include "RMXLTypeDef.h"
00012
00013 class IRMXUser;
00014
00022 class CADRMX_API IRMXInstance
00023 {
00024 public:
00025     IRMXInstance() {};
00026     virtual ~IRMXInstance() {};
00027
00028 public:
00037     virtual RMXResult IsInitFinished(bool& finished) = 0;
00038
00049     virtual RMXResult AddRPMDir(const std::wstring& dirPath, uint32_t option =
       (RMXRPMFolderOption::RMX_RPMFOLDER_OVERWRITE | RMXRPMFolderOption::RMX_RPMFOLDER_API)) = 0;
00050
00060     virtual RMXResult RemoveRPMDir(const std::wstring& dirPath, bool force = true) = 0;
00061
00073     virtual RMXResult IsRPMFolder(const std::wstring& dirPath, bool& isRPMDir, RMX_RPMFolderRelation*
      relation = nullptr) = 0;
00074
00083     virtual RMXResult RegisterApp(const std::wstring& appPath) = 0;
00084
00096     virtual RMXResult UnregisterApp(const std::wstring& appPath) = 0;
00097
00107     virtual RMXResult IsAppRegistered(const std::wstring& appPath, bool& registered) = 0;
00108
```

```
00120     virtual RMXResult NotifyRMXStatus(bool running) = 0;
00121
00132     virtual RMXResult AddTrustedApp(const std::wstring &appPath) = 0;
00133
00144     virtual RMXResult RemoveTrustedApp(const std::wstring &appPath) = 0;
00145
00156     virtual RMXResult EditCopyFile(const std::wstring& nxlFilePath, std::wstring& destPath) = 0;
00157
00174     virtual RMXResult EditSaveFile(const std::wstring& filePath, const std::wstring&
      originalNXLfilePath = L"", bool deleteSource = false, uint32_t exitEdit = 0) = 0;
00175
00188     virtual RMXResult RPMCopyFile(const std::wstring& existingNxlFilePath, const std::wstring&
      newNxlFilePath, bool deleteSource = false) = 0;
00189
00199     virtual RMXResult RPMDeleteFile(const std::wstring& filePath) = 0;
00200
00211     virtual RMXResult GetFileRights(const std::wstring& filePath, std::vector<RMXFileRight>& rights) =
      0;
00212
00229     virtual RMXResult CheckFileRight(const std::wstring& filePath, RMXFileRight right, bool& allowed,
      bool audit = false) = 0;
00230
00242     virtual RMXResult GetFileStatus(const std::wstring& filePath, bool& isProtected,
      RMX_RPMFolderRelation* dirRelation = nullptr) = 0;
00243
00252     virtual RMXResult ReadFileTags(const std::wstring& filePath, std::string& tags) = 0;
00253
00273     virtual RMXResult RPMGetFileInfo(const std::wstring & filepath, std::string &duid, std::string
      &tags,
00274         std::string &tokengroup, std::string &creatorid, std::string &infoext, DWORD &attributes,
      RMX_RPMFolderRelation & dirRelation, BOOL &isNXLFile) = 0;
00275
00296     virtual RMXResult RPMSetFileAttributes(const std::wstring &nxlFilePath, DWORD dwFileAttributes) =
      0;
00297
00314     virtual RMXResult SetViewOveraly(const RMX_VIEWOVERLAY_PARAMS& params) = 0;
00315
00324     virtual RMXResult ClearViewOverlay(void* hTargetWnd) = 0;
00325
00326 };
```

8.7 RMXLResult.h File Reference

Classes

- class RMXResult

Typedefs

- typedef unsigned long RMXErrCode_t

8.7.1 Typedef Documentation

8.7.1.1 RMXErrCode_t

```
typedef unsigned long RMXErrCode_t
```

Error code

Definition at line 12 of file RMXLResult.h.

8.8 RMXLResult.h

```
00001 //
00002 // This is a part of the NextLabs CADRMX Client SDK.
00003 // Copyright (C) 2020 NextLabs, Inc. All Rights Reserved.
00004 //
00005
00006 #pragma once
00007
00008 #include <string>
00009 #include "RMXLTypeDef.h"
00010
00012 typedef unsigned long RMXErrCode_t;
00013
00020 class RMXResult
00021 {
00022 public:
00024     RMXResult() : m_code(0) {};
00025
00033     explicit RMXResult(RMXErrCode_t code, const std::wstring& errMessage)
00034         : m_code(code), m_message(errMessage)
00035     {
00036     }
00038     virtual ~RMXResult() {}
00039
00045     RMXResult(const RMXResult& other)
00046         : m_code(other.m_code), m_message(other.m_message)
00047     {
00048     }
00049
00055     RMXResult& operator = (const RMXResult& other)
00056     {
00057         if (this != &other)
00058         {
00059             m_code = other.m_code;
00060             m_message = other.m_message;
00061         }
00062
00063         return *this;
00064     }
00065
00072     operator bool() const { return (0 == m_code); }
00073
00082     bool operator == (int code) const { return (code == m_code); }
00083
00091     bool operator != (int code) const { return (code != m_code); }
00092
00099     RMXErrCode_t GetErrCode() const { return m_code; }
00100
00107     std::wstring GetErrMessage() const { return m_message; }
00108
00109 protected:
00111     RMXErrCode_t m_code;
00113     std::wstring m_message;
00114 };
00115
```

8.9 RMXLTypeDef.h File Reference

Classes

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

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 }
- enum RMXRPMFolderOption {
  RMX_RPMFOLDER_NORMAL = 1, RMX_RPMFOLDER_OVERWRITE = 4, RMX_RPMFOLDER_USER = 8, RMX_RPMFOLDER_EXT = 0x10,
  RMX_RPMFOLDER_API = 0x20 }
- enum RMXActivityLogOperation {
  RMX_OProtect = 1, RMX_OShare, RMX_ORemoveUser, RMX_OView,
  RMX_OPrint, RMX_ODownload, RMX_OEdit, RMX_ORevoke,
  RMX_ODecrypt, RMX_OCopyContent, RMX_OCaptureScreen, RMX_OClassify,
  RMX_OReshare, RMX_ODelete }
- enum RMXActivityLogResult { RMX_RDenied = 0, RMX_RAllowed }

8.9.1 Detailed Description

Declarations used throughout the CADRMX Client SDK.

Definition in file RMXLTypeDef.h.

8.9.2 Enumeration Type Documentation

8.9.2.1 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.

```
00027 {
00028     RMX_NonRPMFolder,
00029     RMX_RPMFolder,
00030     RMX_RPMAncestralFolder,
00031     RMX_RPMInheritedFolder,
00032 } RMX_RPMFolderRelation;
```

8.9.2.2 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.

```
00038 {
00039     RMX_RIGHT_VIEW = 1,
00040     RMX_RIGHT_EDIT = 2,
00041     RMX_RIGHT_PRINT = 4,
00042     RMX_RIGHT_CLIPBOARD = 8,
00043     RMX_RIGHT_SAVEAS = 0x10,
00044     RMX_RIGHT_DECRYPT = 0x20,
00045     RMX_RIGHT_SCREENCAPTURE = 0x40,
00046     RMX_RIGHT_SEND = 0x80,
00047     RMX_RIGHT_CLASSIFY = 0x100,
00048     RMX_RIGHT_SHARE = 0x200,
00049     RMX_RIGHT_DOWNLOAD = 0x400,
00050     RMX_RIGHT_WATERMARK = 0x40000000
00051 } RMXFileRight;
```

8.9.2.3 RMXRPMFolderOption

```
enum RMXRPMFolderOption
```

RMX RPM folder type

Since

2020.08

Enumerator

RMX_RPMFOLDER_NORMAL	Normal RPM folder. When copying a NXL file to this folder, do not allow overwriting unprotected file with same name if exists
RMX_RPMFOLDER_OVERWRITE	When copying a NXL file to this folder, allow overwriting unprotected file with same name if exists
RMX_RPMFOLDER_USER	User defined RPM folder, which can be managed via "My SkyDRM Folder" UI in SkyDRM Desktop
RMX_RPMFOLDER_EXT	Automatically add ".nxl" extension if it's missing when copying a NXL file to this folder
RMX_RPMFOLDER_API	RPM folder set by RMX, which can be managed and reset via "Managed SkyDRM Folder" UI in SkyDRM Desktop.

Definition at line 57 of file RMXLTypeDef.h.

```
00058 {
00059     RMX_RPMFOLDER_NORMAL = 1,
00060     RMX_RPMFOLDER_OVERWRITE = 4,
00061     RMX_RPMFOLDER_USER = 8,
00062     RMX_RPMFOLDER_EXT = 0x10,
00063     RMX_RPMFOLDER_API = 0x20
00064 } RMXRPMFolderOption;
```

8.9.2.4 RMXActivityLogOperation

```
enum RMXActivityLogOperation
```

File activity type

Enumerator

RMX_OProtect	Protect file
RMX_OShare	Share a protected file
RMX_ORemoveUser	Remove user
RMX_OView	View a protected file
RMX_OPrint	Print a protected file
RMX_ODownload	Download a protected file
RMX_OEdit	Save or Edit file a protected file
RMX_ORevoke	Revoke operation
RMX_ODecrypt	Decrypt a protected file
RMX_OCopyContent	Copy content from a protected file
RMX_OCaptureScreen	Capture screen for a protected file
RMX_OClassify	Reclassify a protected file
RMX_OReshare	Reshare a protected file
RMX_ODelete	Delete a protected file

Definition at line 69 of file RMXLTypeDef.h.

```
00069             {
00070     RMX_OProtect = 1,
00071     RMX_OShare,
00072     RMX_ORemoveUser,
00073     RMX_OView,
00074     RMX_OPrint,
00075     RMX_ODownload,
00076     RMX_OEdit,
00077     RMX_ORevoke,
00078     RMX_ODecrypt,
00079     RMX_OCopyContent,
00080     RMX_OCaptureScreen,
00081     RMX_OClassify,
00082     RMX_OReshare,
00083     RMX_ODelete
00084 } RMXActivityLogOperation;
```

8.9.2.5 RMXActivityLogResult

```
enum RMXActivityLogResult
```

File activity result

Enumerator

RMX_RDenied	Operation is denied
RMX_RAllowed	Operation is allowed

Definition at line 89 of file RMXLTypeDef.h.

```
00089                 {
00090     RMX_RDenied = 0,
00091     RMX_RAllowed
00092 } RMXActivityLogResult;
```

8.10 RMXLTypeDef.h

```
00001 //
00002 // This is a part of the NextLabs CADRMX Client SDK.
00003 // Copyright (C) 2020 NextLabs, Inc. All Rights Reserved.
00004
00011 #pragma once
00012
00013 #include <stdint.h>
00014 #include <string>
00015 #include <vector>
00016
00017 #ifdef CADRMXLIB_EXPORTS
00018 #define CADRMX_API __declspec(dllexport)
00019 #else
00020 #define CADRMX_API __declspec(dllimport)
00021 #endif
00022
00026 typedef enum
00027 {
00028     RMX_NonRPMFolder,
00029     RMX_RPMFolder,
00030     RMX_RPMAncestralFolder,
00031     RMX_RPMInheritedFolder,
00032 } RMX_RPMFolderRelation;
00033
00037 typedef enum
00038 {
00039     RMX_RIGHT_VIEW = 1,
00040     RMX_RIGHT_EDIT = 2,
00041     RMX_RIGHT_PRINT = 4,
00042     RMX_RIGHT_CLIPBOARD = 8,
00043     RMX_RIGHT_SAVEAS = 0x10,
00044     RMX_RIGHT_DECRYPT = 0x20,
00045     RMX_RIGHT_SCREENCAPTURE = 0x40,
00046     RMX_RIGHT_SEND = 0x80,
00047     RMX_RIGHT_CLASSIFY = 0x100,
00048     RMX_RIGHT_SHARE = 0x200,
00049     RMX_RIGHT_DOWNLOAD = 0x400,
00050     RMX_RIGHT_WATERMARK = 0x40000000
00051 } RMXFileRight;
00052
00057 typedef enum
00058 {
00059     RMX_RPMFOLDER_NORMAL = 1,
00060     RMX_RPMFOLDER_OVERWRITE = 4,
00061     RMX_RPMFOLDER_USER = 8,
00062     RMX_RPMFOLDER_EXT = 0x10,
00063     RMX_RPMFOLDER_API = 0x20
00064 } RMXRPMFolderOption;
00065
00069 typedef enum {
00070     RMX_OProtect = 1,
00071     RMX_OShare,
00072     RMX_ORemoveUser,
00073     RMX_OView,
00074     RMX_OPrint,
00075     RMX_ODownload,
00076     RMX_OEdit,
00077     RMX_ORevoke,
00078     RMX_ODecrypt,
00079     RMX_OCopyContent,
00080     RMX_OCaptureScreen,
00081     RMX_OClassify,
00082     RMX_OReshare,
00083     RMX_ODelete
00084 } RMXActivityLogOperation;
```

```
00085
00089 typedef enum {
00090     RMX_RDenied = 0,
00091     RMX_RAllowed
00092 } RMXActivityLogResult;
00093
00097 struct RMX_EVAL_ATTRIBUATE
00098 {
00099     std::wstring key;
00100     std::wstring value;
00101 };
00102
00107 struct RMX_VIEWOVERLAY_PARAMS
00108 {
00109     void* hTargetWnd;
00110     std::vector<std::string> vecTags;
00111     std::vector<RMX_EVAL_ATTRIBUATE> vecCtxAttrs;
00112     int displayOffsets[4];
00113 };
00114
00118 struct RMX_OBLIGATION_INFO {
00119     std::wstring                    name;
00120     std::vector<RMX_EVAL_ATTRIBUATE>  options;
00121 };
00122
00126 struct RMX_CENTRAL_RIGHT
00127 {
00128     RMXFileRight right;
00129     std::vector<RMX_OBLIGATION_INFO> obligations;
00130 };
```

8.11 RMXLUser.h File Reference

Classes

- class IRMXUser

8.12 RMXLUser.h

```
00001 //
00002 // This is a part of the NextLabs CADRMX Client SDK.
00003 // Copyright (C) 2020 NextLabs, Inc. All Rights Reserved.
00004 //
00005 #pragma once
00006
00007 #include "RMXLResult.h"
00008 #include "RMXLTypeDef.h"
00009
00017 class CADRMX_API IRMXUser
00018 {
00019 public:
00020     IRMXUser() {};
00021     virtual ~IRMXUser() {};
00022
00023 public:
00029     virtual const std::wstring GetName() = 0;
00030
00036     virtual const std::wstring GetEmail() = 0;
00037
00043     virtual uint32_t GetUserID() = 0;
00044
00055     virtual RMXResult ProtectFile(const std::wstring& filePath, const std::wstring& targetDir, const
      std::string& tags, std::wstring& newNxlFile) = 0;
00056
00067     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;
00068
00076     virtual bool GetDefaultPolicyBundle(std::string& policyBundle) = 0;
00077
00083     virtual const std::string GetSystemProjectTenantId() = 0;
00084
00090     virtual const std::string GetDefaultTokenGroupName() = 0;
00091
00104     virtual RMXResult MergeTags(const std::string& unionTags, std::string& mergedTags) = 0;
00105
00115     virtual RMXResult AddActivityLog(const std::wstring & nxlFilePath, RMXActivityLogOperation op,
      RMXActivityLogResult result) = 0;
00116 };
```

Index