opentext™

OpenText™ Records Management

Web Services Guide

Records Management adds functions for managing documents and records in Content Server. This guide describes the Web Services and API functions exposed in the Records Management, Physical Objects, and Security Clearance modules.

LLESRCM160204-PRE-EN-1

OpenText™ Records Management
Web Services Guide
LLESRCM160204-PRE-EN-1
Rev.: 2020-Mar-08

This documentation has been created for software version 16.2.4.
It is also valid for subsequent software versions as long as no new document version is shipped with the product or is published at https://knowledge.opentext.com.

OpenText Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Chapter 1

Overview

This guide includes descriptions for Web Services function calls for the Records Management, Physical Objects, Security Clearance and Classifications modules. This guide also includes information for Service Data Objects, when available.

The Records Management API allows you to update Records Management data, such as Essential code, Status code, or Status Date. The Physical Objects API allows you to retrieve all physical item types and the physical properties for the physical item types that exist in the system. This information is necessary for creating physical items using API.

Note: All types (string, integer, boolean) that are used in the request assoc to define the properties for a Physical Object need to be passed in as string variables. For example, on the **Physical Properties page** tab on the Physical Item Types page, you can define custom fields for Physical Objects. One of the types you can choose is `Boolean`, which translates to a check box in the **Properties** section of a Physical Object. When the browser submits a page with a check box selected, the value server-side is on; if it is not selected, the value server-side does not exist.

Chapter 2

Records Management

This chapter provides information about configuring Records Management Web Services and provides an explanation of each function call in Records Management Web Services. The section also provides details about the Records Management Service Data Objects.

2.1 Configuring Content Web Services and Records Management Web Services

Records Management Web Services allows you to edit Records Management specific metadata.

To configure Content Web Services and Records Management Web Services:

1. Create an Internet Information Services (IIS) application using the `OTCS_Home\webservices\dotnet\les-services` directory.

   Note: For more information, see the *Content Web Services Installation Guide* in the **Content Web Services** section of the **Content Server SDK** in the OpenText Developer NetworkOTDN on the OpenText Knowledge Center.

2. Open Internet Information Services (IIS) Manager, click the application name, and then do the following:

   - If you did not previously set the Application Pool value to `Classic .NET AppPool`, click the **Basic Settings** link in the **Actions** panel. In the Edit Application window, click the **Select** button, choose `Classic .NET AppPool` in the **Application Pool** drop-down list, and then click **OK**.
   - If Content Server uses SSL, click the **Advanced Settings** link in the **Actions** panel. In the Advanced Settings window, type `https` in the **Enabled Protocols** list in the **Behavior** section.
   - In the **IIS** section of the main IIS Manager panel, double-click **Directory Browsing**. In the **Actions** panel of the Directory Browsing window, click the **Enable** link.
   - If Content Server does not user SSL, double-click **SSL Settings** in the **IIS** section of the main IIS Manager panel. In the SSL Settings window, clear the **Require SSL** check box.

3. If you are using SSL, edit the `web.config` file in the *<ContentServer_Home>*`\webservices\dotnet\les-services` folder.

   Note: The comments in the `web.config` file provide instructions for enabling SSL.

4. Copy the `OpenText.Livelink.Service.RecMan.dll` file from the *<ContentServer_Home>*`\webservices\dotnet\les-services-recordsmanagement\bin` directory to the *<ContentServer_Home>*`\webservices\dotnet\les-services\bin` directory.
5. Copy the `RecordsManagement.svc` file from the *<ContentServer_Home>*`\webservices\dotnet\les-services-recordsmanagement` directory to the *<ContentServer_Home>*`\webservices\dotnet\les-services` directory.
6. Copy the following sections from the *<ContentServer_Home>*`\webservices\dotnet\les-services-recordsmanagement\Web.config` directory to the specified sections of the *<ContentServer_Home>*`\webservices\dotnet\les-services\Web.config` directory:

   - **ServiceBehaviors**

     ```
     <behavior name="RecordsManagementBehavior">
     <serviceMetadata httpGetEnabled="true" httpGetUrl="" />
     <serviceDebug includeExceptionDetailInFaults="true"
     httpHelpPageUrl="" />
     </behavior>
     ```

   - **basicHttpBinding**:

     ```
     <binding name="RecordsManagementBinding"
     maxReceivedMessageSize="1024000" messageEncoding="Text"
     transferMode="Buffered">
     <security mode="None">
     <transport clientCredentialType="None"/>
     </security>
     </binding>
     <binding name="RecordsManagementBinding_SSL"
     maxReceivedMessageSize="1024000"
     messageEncoding="Text" transferMode="Buffered">
     <security mode="None">
     <transport clientCredentialType="None"/>
     </security>
     </binding>
     ```

   - **services**

     ```
     <service behaviorConfiguration="RecordsManagementBehavior"
     name="OpenText.Livelink.Service.RecMan.RecordsManagement_WCF">
     <endpoint address="" binding="basicHttpBinding"
     bindingConfiguration="RecordsManagementBinding"
     bindingNamespace="urn:RecMan.service.livelink.opentext.com"
     contract="OpenText.Livelink.Service.RecMan.IRecordsManagement_
     WCF" />
     <!-- SSL disabled
     <endpoint address="" binding="basicHttpBinding"
     bindingConfiguration="RecordsManagementBinding_SSL"
     bindingNamespace="urn:RecMan.service.livelink.opentext.com"
     contract="OpenText.Livelink.Service.RecMan.IRecordsManagement_
     WCF" />
     -->
     </service>
     ```

7. Restart IIS.

Note: You can test to ensure that Records Management is properly deployed by navigating to the following URL in a browser: `http://<server>:<port>/<ContentServer_Home>/RecordsManagement.svc?wsdl`

2.2 Records Management Web Services Functions

2.2.1 HoldCollectInfo()

The HoldCollectInfo() function collects information about distribution of items that are on hold.

Parameters

{"Long HoldID, "ID of a hold to collect information for" , 'IN'}

{"HoldDistribution", "SDO definition" , 'OUT'}

2.2.2 HoldRetrieveItems()

The HoldTrieveItems() function retrieves a specified type and number of items on hold.

Parameters

{"Long HoldID", "ID of a hold to retrieve items" , 'IN'}

{"Integer ItemsPerPage", "Number of items to be returned in one call" , 'IN'}

{"Integer PageNumber", "Page number to be returned" , 'IN'}

{"Boolean itemsAndChildren", "Boolean indicating to return items directly on hold and their children" , 'IN'}

{"Boolean parents", "Boolean indicating to return parents of items directly on hold" , 'IN'}

{"Boolean classifiedItems", "Boolean indicating to return items classified with RM Classification on hold" , 'IN'}

{"Boolean boxContents", "Boolean indicating to return items in box on hold" , 'IN'}

{"Boolean boxesOfParent", "Boolean indicating to return boxes on hold because of parent object" , 'IN'}

{"Boolean boxesOfChildren", "Boolean indicating to return boxes on hold because of child object" , 'IN'}

{"Boolean classParents", "Boolean indicating to return items on hold because one of children has RM Classification on hold" , 'IN'}

{"Boolean classChildren", "Boolean indicating to return items on hold because one of parents has RM Classification on hold" , 'IN'}

{"HoldPage", "SDO definition" , 'OUT'}

2.2.3 RMApplyClassification()

The RMApplyClassification() function classifies an item and updates RM metadata.

Parameters

{"Long nodeID", "DataID of a node to be classified" , 'IN'}

{"Long classID", "DataID of the primary classification" , 'IN'}

{"RMAdditionalInfo AdditionalInfo", "SDO definition" , 'IN'}

{"Integer[] secClassIDs", "List of secondary class IDs to be applied" , 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.4 RMApplyHold()

The RMApplyHold( ) function applies holds to managed objects. The user performing this action must be a Records Manager or have functional access.

Parameters

{"Long nodeID", "DataID of node."}

{"Long holdID", "HoldID of the hold to be applied."}

{"Boolean-return true if success", 'OUT'}

2.2.5 RMApplyProvenances()

The RMApplyProvenances() function assigns a Provenance to managed Content Server objects.

Parameters

{"Long NodeID", "DataID of a node."},

{"Long[] provenanceIDs" , "list of DataIDs of provenances."},

{"Boolean-return true if success", 'OUT'}

2.2.6 RMApplySecondaryClassification()

The RMApplySecondaryClassification() function applies a secondary RM Classification to the object.

Parameters

{"Long NodeID", DataID of a node", 'IN'}

{"Long ClassID", " DataID of classification to be applied", 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.7 RMBatchApplyHold()

The RMBatchApplyHold() function applies Holds to multiple items.

Parameters

{"Long[] NodeIDList", "List of DataID of nodes to have hold assigned.", 'IN' }

{"Long holdID", "HoldID of the hold to be applied." , 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.8 RMBatchMarkOfficial()

The RMBatchMarkOfficial() function marks multiple items as official.

Parameters

{ "Long[] NodeIDList", "List of IDs of the nodes to mark official.", 'IN' }

{"Boolean-return true if success", 'OUT'}

2.2.9 RMBatchRemoveClassificatio()

Removes RM Classifications from managed objects.

Parameters

{"Long[] NodeIDList", "List of IDs of the nodes to remove classification.", 'IN' }

{"Long[] ClassIDList", "List of classifications ID to remove", 'IN' }

{"Boolean-return true if success", 'OUT'}

2.2.10 RMBatchRemoveHold()

The RMBatchRemoveHold() function removes Holds from multiple items.

Parameters

{"Long[] NodeIDList", "List of DataID of nodes to have hold removed." , 'IN'}

{"Long holdID", "HoldID of the hold to be applied." , 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.11 RMBatchRemoveOfficial()

The RMBatchRemoveOfficial() function removes Holds from multiple items.

Parameters

{ "Long[] NodeIDList", "List tof IDs of the nodes to remove official." , 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.12 RMBatchUpdate()

The RMBatchUpdate() function classifies and updates RM metadata for multiple items.

Parameters

{ "Long[] NodeIDList", "List of DataID of nodes to be classified." , 'IN'}

{ "Long classID", "DataID of the classification." , 'IN'}

{ "RMAdditionalInfo AdditionalInfo","SDO definition", 'IN' }

{"Boolean-return true if success", 'OUT'}

2.2.13 RMClassificationFavorites()

The RMClassificationFavorites() function returns the RM Classifications from the Favorites list of the user logged in to Content Server.

Parameters

{"RMClassDefInfo[]", "SDO definition" , 'OUT'}

2.2.14 RMClassificationPickList()

The RMClassificationPickList() function returns the RM Classifications from the picklist of the user logged in to Content Server. The GroupPickList parameter indicates whether the list of RM Classifications is returning a user picklist or a group picklist.

Parameters

{'Integer PickListType', 'User = 0 or Group =1 PickList', 'IN' }

{RMClassDefInfo[]", "SDO definition" , 'OUT'}

2.2.15 RMCreateHold()

The RMCreateHold() function creates a Disposition Hold.

Parameters

{ "RMHoldInfo HoldInfo","SDO definition"}

2.2.16 RMCreateXReference()

The RMCreateXReference() function creates cross references between two objects.

Parameters

{"Long NodeID", "DataID of node.", 'IN'}

{"Long xNodeID", "DataID of node to be cross referenced.", 'IN'}

{"String DocXRefTyp", "Cross Reference code to be used" , 'IN'}

{"String xComments", "Additional comments", 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.17 RMDeclareRecor()

The RMDeclareRecord() function finalizes an item as a record

Parameters

{

{"Long NodeID", "ID of a node.", 'IN' }

{"Boolean-return true if success", 'OUT'}

2.2.18 RMDeleteHold()

The RMDeleteHold() function deletes a Disposition Hold if it is not in use.

Parameters

{"Long HoldID", "ID of a hold to be deleted." , 'IN' }

{"Boolean-return true if success", 'OUT'}

2.2.19 RMGetClassificationChildren()

The RMGetClassificationChildren( ) function returns all classifications created under a specified classification node. Information returned is NodeID, Name, FileNumber, HasChildren, Type, and Selectable.

Parameters

{"Long ClassID", "ID of classification." , 'IN' }

{RMClassDefInfo[]", "SDO definition" , 'OUT'}

2.2.20 RMGetClassificationInfo()

The RMGetClassificationInfo() function retrieves metadata of an RM Classification.

Parameters

{"Long NodeID", "DataID of a classification" , 'IN'}

{"RMClassDefInfo", "SDO definition." , 'OUT'}

2.2.21 RMGetClassificationRoot()

The RMGetClassificationRoot( ) function returns information about the Classification Volume. Information returned is NodeID, Name, FileNumber, HasChildren, Type = -1, Selectable = false.

Parameters

{ "RMClassDefInfo", "SDO definition" , 'OUT'}

2.2.22 RMGetClassifyInfo()

The RMGetClassifyInfo( ) function returns information about RM Classifications assigned to an item. This function fails if the user performing the action does not have See and See Contents permissions for the object.

Parameters

{"Long NodeID", "ID of a node." , 'IN'}

{"RMAdditionalInfo", "SDO definition." , 'OUT'}

2.2.23 RMGetCodes()

The RMGetCodes( ) function returns all attributes for Status, Essential, Storage, RSI, Hold Type, Cross Reference, Accession, and Cycle Period.

Parameters

{"RMCodesInfo", "SDO definition.", 'OUT'}

2.2.24 RMGetFieldInfos()

The RMGetFieldInfos() function returns RM settings for the creation process.

Parameters

{ "String objectType", "Sub type of an object being created. Example of values: '144' or 'Document' ", 'IN'} }

{"RMAttributeGroupDefinition", "SDO definition" 'OUT'}

2.2.25 RMGetHolds()

The RMGetHolds() function retrieves all existing holds.

Parameters

{"Boolean Active", "Boolean indication to return only active holds" , 'IN'}

{"RMHoldInfo", "SDO definition" , 'OUT'}

2.2.26 RMGetManagedTypes()

The RMGetManagedTypes( ) function returns a list of managed subtypes (for example, 136, 144, 0).

Parameters

{"RMManagedTypeInfo[]", "SDO definition" , 'OUT'}

2.2.27 RMGetRSIRetention()

The RMGetRSIRetention() function returns the retention types on an RSI schedule.

Parameters

{"String RSI", "The name of RSI." , 'IN'}

{"RMRSIRetention", "SDO definition" , 'OUT'}

2.2.28 RMGetXReferenc()

The RMGetXReference() function obtains the Cross Reference of an object.

Parameters

{ "Long NodeID", "DataID of node." , 'IN'}

{"RMXRefInfo[]", "SDO definition" , 'OUT'}

2.2.29 RMListProvenances()

The RMListProvenances() function lists Provenances assigned to managed objects.

Parameters

{'NodeID', { IntegerType }, 'DataID of item to have provenance listed', 'IN' }

{"ProvenanceInfo", "definition", 'OUT' }

2.2.30 RMMarkOfficial()

The RMMarkOfficial( ) function assigns an official mark to single Content Server objects. It will not assign official mark to sub-items. If there are inheritance rules set up in the system each of the children will need to have official mark assigned in separate calls for that function.

This function fails if the user does not have Edit Permissions on the object, or the object does not have an RM classification assigned.

Parameters

{"Long NodeID", "ID of a node." , 'IN' }

{"Boolean-return true if success", 'OUT'}

2.2.31 RMRemoveClassification()

The RMRemoveClassification( ) function removes classifications from managed objects. The user performing this action must be a Records Manager or have functional access.

Parameters

{"Long NodeID", "ID of a node." , 'IN'}

{"Long[] ClassIDList", "list of classID to remove", 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.32 RMRemoveHold()

The RMRemoveHold( ) function removes holds from managed objects. The user performing this action must be a Records Manager or have functional access.

Parameters

{"Long nodeID", "DataID of node." , 'IN'}

{"Long holdID", "HoldID of the hold to be removed." , 'IN'}

{"Long userid", "User ID of User Based Hold to be removed. Pass 0 if not a User Based Hold.", 'IN'}

{"String docxreftyp", "Cross-Reference Type of Cross-Reference associated hold to be removed. Pass empty string if not a Cross-Reference associated hold" , 'IN'}

{"Long queryid", "Query ID of Search Enabled Hold to be removed. Pass 0 if not a Search Enabled Hold." , 'IN'}

{"String removeComments", "comments for this removal." , 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.33 RMRemoveOfficia()

The RMRemoveOfficial() function removes the official marking from the single Content Server object. This function will fail if a user does not have Edit permissions on the object.

Parameters

{"Long NodeID", " DataID of the object which will have official mark removed" , 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.34 RMRemoveProvenances()

The RMRemoveProvenances() function removes Provenances from managed objects.

Parameters

{'Long NodeID', 'ID of a node.', 'IN' }

{'Long[] provenanceIDs', 'List of DataIDs of provenances IDs.', 'IN' }

{"Boolean-return true if success", 'OUT'}

2.2.35 RMRemoveXReferenc()

The RMRemoveXReference() function removes cross references between two objects.

Parameters

{"Long NodeID", "DataID of node." , 'IN'}

{"Long xNodeID", "DataID of node to be cross referenced." , 'IN'}

{"String DocXRefTyp", "Cross Reference code to be used" , 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.36 RMShowAllHolds()

The RMShowAllHolds( ) function returns all holds assigned directly to items or to parent objects, child objects, or RM classifications.

Parameters

{"Long nodeID", "DataID of node.", 'IN'}

{"RMHoldDocInfo[]", "SDO definition" , 'OUT'}

2.2.37 RMShowParentHolds()

The RMShowParentHolds() function returns all Holds applied to parents of the item.

Parameters

{"Long nodeID", "DataID of node." , 'IN'}

{"RMHoldDocInfo[]", "SDO definition" , 'OUT'}

2.2.38 RMUpdateDetails()

The RMUpdateDetails() function classifies a node and updates RM metadata.

Parameters

{"Long nodeID", "DataID of node to be classified." , 'IN'}

{"Long classID", "DataID of the primary classification." , 'IN'}

{"RMAdditionalInfo AdditionalInfo", "SDO definition" , 'IN'}

{"Boolean-return true if success", 'OUT'}

2.2.39 RMUserFunctionalAccess()

The RMUserFunctionalAccess() function returns all functions that the current user has assigned.

Parameters

{ "Integer[] functionsList", "Array of funcID to be checked. If empty return" , 'IN'}

{ "RMFuncAccess[]", "definition " , 'OUT'}

2.3 Records Management Service Data Objects

This section provides details for Records Management Service Data Objects (SDO).

2.3.1 HoldDistribution

The HoldDistribution SDO returns a count of items on hold with the type (direct, indirect, or from a Classification).

Name	Type	Data Type	Description
fCountDirectAndChildren	Integer	Integer	The count of items directly on hold and its children
fCountParents	Integer	Integer	The count of parent items indirectly on hold
fCountFromClassification	Integer	Integer	The count items indirectly on hold from RM Classification
fCountBoxContents	Integer	Integer	The count of items indirectly on hold from the box
fCountBoxOfParents	Integer	Integer	The count of boxes indirectly on hold from parents
fCountBoxOfChildren	Integer	Integer	The count of boxes indirectly on hold from children
fCountParentOfRMClass	Integer	Integer	The count of items indirectly on hold because they are parents of item which RM Classification is on hold.
fCountChildrenOfRMClass	Integer	Integer	The count of items indirectly on hold because they are children of item which RM Classification is on hold.

2.3.2 HoldPage

The HoldPage SDO returns a list of DataID returned for a specified page when retrieving a large number of items on hold. The list includes information about the action of retrieving the items on hold.

Name	Type	Data Type	Description
fDataIDs	Long	List	The list of Data IDs of items returned for specified page
fCompleted	Boolean	Boolean	The boolean indicating that there are no more items to retrieve.

fTypeReturned	Integer	Integer	The integer indicating which type of items are being returned 1-items and children,2-parents,3-classified items,4-contents of a box 5-boxes of parents,6-boxes of children

2.3.3 ProvenanceInfo

The ProvenanceInfo SDO returns a list of Provenances assigned to the item.

Name	Type	Data Type	Description
fMyNode	Node	Node	The provenance node object which describe all meta data of Content Server object.
fStatus	String	String	The status of provenance describing if provenance is accepted, pending or rejected.
fSelectable	Boolean	Boolean	The flag indicating if provenance is selectable

2.3.4 RMAccession

The RMAccession SDO represents information about an Accession code.

Name	Type	Data Type	Description
fAccession	String	String	The accession code
fDescription	String	String	The description of a code

2.3.5 RMAdditionalInfo

The RMAdditionalInfo SDO represents Records Management metadata associated with an item.

Name	Type	Data Type	Description
fOfficial	Boolean	Boolean	The flag indicating if item is official
fDocDate	Date	Date	The Record Date of the node
fLastReviewDate	Date	Date	The date when node was review last time
fNextReviewDate	Date	Date	The date when node will be review next time
fReceivedDate	Date	Date	The date when node was received

fStatusDate	Date	Date	The date when status was applied to the node
fClassID	Long	Integer	The classification ID assigned to the node
fCyclePeriod	Integer	Integer	An integer specifying how often node needs to be reviewed, 1- monthly,2- semi annually, 3- quarterly, 12- annually
fRMClassDefInfo	List	RMClassDefInfo	The detailed information about classifications assigned to the node
fAccession	String	String	The code, from Accession table, created in table maintenance
fAddressee	String	String	The string stored as Addressee
fEssential	String	String	The code, from Essential table, created in table maintenance
fEstablishment	String	String	The string stored as Originating Organization
fOriginator	String	String	The string stored as Author or Originator
fRSI	String	String	The code created in Records Managements workspace, Record Series Identifier used for dispositions
fSentTo	String	String	The string stored as Other Addressee
fStatus	String	String	The code, from FILE_STATUS table, created in table maintenance
fStorage	String	String	The code, from STORAGE table, created in table maintenance
fSubject	String	String	The string stored as subject
fOfficialRemove	Boolean	Boolean	The flag indicating if item will have official removed
fRM_RetentionDays	Integer	Integer	The number of days left for retention
fRM_RetentionMode	String	String	The possible values READ_ONLY, INFINITE, FIXED

2.3.6 RMAttributeGroupDefinition

The RMAttributeGroupDefinition SDO returns all needed information about RM fields used in the create process.

Name	Type	Data Type	Description
fUseRecordVersion	Boolean	Boolean	Aa flag indicating if Xreference should be used for record versions
fXreference	Boolean	Boolean	A flag indicating if XReference should be shown on create page
fRMVitalCode	List	String	An array of Essential codes designated as vital
fVersionCode	Boolean	Boolean	The XReference code designated for use as record version code.

2.3.7 RMClassDefInfo

The RMClassDefInfo SDO represents information about an RM Classification.

Name	Type	Data Type	Description
fHasChildren	Boolean	Boolean	The flag indicating if classification has any children
fSelectable	Boolean	Boolean	The flag indicating if classification can be assigned to new items
fClosedDate	Date	Date	The date when classification was closed
fLastAdditionDate	Date	Date	The last date when item was classified with the classification
fStatusDate	Date	Date	The date when status was assigned or updated
fNodeID	Long	Long	The ID of classification
fClosed	Integer	Integer	The flag indicating if classification is closed
fCyclePeriod	Integer	Integer	An integer specifying how often node needs to be reviewed, 1-monthly,2- semi annually, 3-quarterly, 12- annually
fType	Integer	Integer	The flag indicating type of classification. 1 - Classification Tree, 2- Classification, 3- RM Classification or RM Folder or RM Part, 4- Provenance

fRMRSIRetention	Object	RMRSIRetention	Object returning retention information
fFileNumber	String	String	The full file number of classification
fName	String	String	The name of classification
fDescription	String	String	The description of classification
fDispAuthority	String	String	The disposition authority of classification
fDisposition	String	String	Currently not used
fEssential	String	String	The code, from Essential table, created in table maintenance
fRSI	String	String	The code created in Records Managements workspace, Record Series Identifier used for dispositions
fStatus	String	String	The code, from FILE_STATUS table, created in table maintenance
fStorage	String	String	The code, from STORAGE table, created in table maintenance

2.3.8 RMCodesInfo

The RMCodesInfo SDO represents the Essential, RSI, Status, Storage, Accession, Cycle Period, Xreference, and Hold Type codes information.

Name	Type	Data Type	Description
fRMEssCode	List	RMEssCode	An array of essential codes
fRMHoldType	List	RMHoldType	An array of hold type codes
fRMRSI	List	RMRSI	An array of RSI codes
fRMStatusCode	List	RMStatusCode	An array of status codes
fRMStorage	List	RMStorage	An array of storage codes
fRMAccession	List	RMAccession	An array of accession codes
fRMCyclePeriod	List	RMCyclePeriod	An array of cycle period values and description

fRMDocXRefTyp	List	RMDocXRefTyp	An array of cross reference codes

2.3.9 RMCyclePeriod

The RMCyclePeriod SDO represents information about the Cycle Period code.

Name	Type	Data Type	Description
fCyclePeriod	Integer	Integer	The integer value of cycle period code
fDescription	String	String	The description of a code

2.3.10 RMDocXRefTyp

The RMDocXRefTyp SDO represents information about the Cross Reference code.

Name	Type	Data Type	Description
fDocXRef	String	String	The cross reference code
fDescription	String	String	The description of a code
fRelatedCode	String	String	The related code

2.3.11 RMEssCode

The RMEssCode SDO represents information about an Essential code.

Name	Type	Data Type	Description
fEssential	String	String	The essential code
fDescription	String	String	The description of a code

2.3.12 RMField

The RMField SDO represents information about the create process for one of the RM fields.

Name	Type	Data Type	Description
fAdd	Boolean	Boolean	The flag to display that field on create page
fAddReq	Boolean	Boolean	The flag indicating if that field is required on create page
fTabReq	Boolean	Boolean	The flag indicating if that field is required on Records Details tab

fOrder	Integer	Integer	The number indicating the position of a field on create page
fInputType	String	String	The type of input field, possible values are : SELECT, TEXTAREA,TEXT,DateType,BooleanType
fLabel	String	String	The label of a field as shown on create page
fName	String	String	The name of a field as passed to create process
fCreateName	String	String	The name of a table which keeps code and is used in <Create New Code> functionality
fSize	String	String	The size of field for TEXT and TEXTAREA fields

2.3.13 RMFuncAccess

The RMFuncAccess SDO represents information about functions that are assigned to a user.

Name	Type	Data Type	Description
fFuncID	Integer	Integer	The function ID
fValue	Boolean	Boolean	The value of true or false if functions is not assigned

2.3.14 RMHoldDocInfo

The RMHoldDocInfo SDO represents hold information associated with a node.

Name	Type	Data Type	Description
fDateApplied	Date	Date	date when hold was applied (defaulted to todays date when hold was created)
fActiveHold	Integer	Integer	The flag indicating if hold is active
fDocDataID	Long	Long	The ID of a node which has hold assigned
fDocParentID	Long	Long	The Parent ID of a node which has hold assigned
fDocSubType	Integer	Integer	The sub type of a node which has hold assigned
fHoldID	Long	Long	The ID of hold assigned to The node
fApplyPatron	String	String	The name of Content Server user who created The hold

fDocName	String	String	The name of Content Server node which has hold assigned
fHoldComment	String	String	The comments of the hold
fHoldName	String	String	The name of hold assigned
fHoldType	String	String	The hold type

2.3.15 RMHoldInfo

The RMHoldInfo SDO represents information about a hold in Content Server.

Name	Type	Data Type	Description
fDateApplied	Date	Date	The date when hold was applied (default to todays date when hold was created)
fDateRemoved	Date	Date	The date when hold was suspended
fDateToRemove	Date	Date	The date when hold was planned to be suspended
fEditDate	Date	Date	not used
fActiveHold	Integer	Integer	The flag indicating if hold is active
fHoldID	Long	Long	The ID of a hold
fAlternateHoldID	String	String	The string value of alternate ID of a hold
fApplyPatron	String	String	The name of Content Server user who created the hold
fEditPatron	String	String	not used
fHoldComment	String	String	The comments added during creation of a hold
fHoldName	String	String	The name of a hold
fHoldType	String	String	The type of a hold
fOBJECT	String	String	not used
fRemovalComment	String	String	The comments added when hold was suspended
fRemovalPatron	String	String	The name of Content Server user who suspended the hold
fParentID	Long	Long	The Id of Holder Group

2.3.16 RMHoldType

RMHoldType SDO represents information about a type of hold

Name	Type	Data Type	Description
fHoldType	String	String	The hold type code
fDescription	String	String	The description of a code

2.3.17 RMManagedTypeInfo

The RMManagedTypeInfo SDO represents an item's subtype that is managed by Records Management.

Name	Type	Data Type	Description
fSubType	Integer	Integer	An integer corresponding to sub type of managed node

2.3.18 RMRSI

The RMRSI SDO represents information about a RSI.

Name	Type	Data Type	Description
fRSI	String	String	The name of RSI
fDescription	String	String	The description of RSI

2.3.19 RMRSIRetention

The RMRSIRetention SDO represents information about retention on a specific RSI.

Name	Type	Data Type	Description
fRM_RetentionDays	Integer	Integer	The number of the days
fRM_RetentionMode	String	String	The type of retention with possible values READ_ONLY,INFINITE, FIXED

2.3.20 RMStatusCode

The RMStatusCode SDO represents information about a Status code.

Name	Type	Data Type	Description
fStatus	String	String	The status code
fDescription	String	String	The description of a code

2.3.21 RMStorage

The RMStorage SDO represents information about a storage codes.

Name	Type	Data Type	Description
fStorage	String	String	The storage code
fDescription	String	String	The description of a code

2.3.22 RMXRefInfo

The RMXRefInfo SDO represents information about cross references.

Name	Type	Data Type	Description
fDocXRef	String	String	The string corresponding to Xreference code
fLocation	String	String	The string corresponding to Location of object in Content Server
fMyNode	Node	Node	The node object which describe all meta data of Content Server object
fXComment	String	String	t he string corresponding to Xreference comment

Chapter 3

Physical Objects

This chapter provides an explanation of each function call in Physical Objects Web Services.

3.1 Configuring Content Web Services and Physical Objects Web Services

Physical Objects Web Services allow to create physical items, and borrow, return, or update physical item related metadata.

To configure Content Web Services and Records Management Web Services:

1. Create an Internet Information Services (IIS) application using the `OTCS_Home\webservices\dotnet\les-services` directory.

   Note: For more information, see the *Content Web Services Installation Guide* in the **Content Web Services** section of the **Content Server SDK** in the OpenText Developer NetworkOTDN on the OpenText Knowledge Center.

2. Open Internet Information Services (IIS) Manager, click the application name, and then do the following:

   - If you did not previously set the Application Pool value to `Classic .NET AppPool`, click the **Basic Settings** link in the **Actions** panel. In the Edit Application window, click the **Select** button, choose `Classic .NET AppPool` in the **Application Pool** drop-down list, and then click **OK**.
   - If Content Server uses SSL, click the **Advanced Settings** link in the **Actions** panel. In the Advanced Settings window, type `https` in the **Enabled Protocols** list in the **Behavior** section.
   - In the **IIS** section of the main IIS Manager panel, double-click **Directory Browsing**. In the **Actions** panel of the Directory Browsing window, click the **Enable** link.
   - If Content Server does not user SSL, double-click **SSL Settings** in the **IIS** section of the main IIS Manager panel. In the SSL Settings window, clear the **Require SSL** check box.

3. If you are using SSL, edit the `web.config` file in the `<ContentServer_Home>\webservices\dotnet\les-services` folder.

   Note: The comments in the `web.config` file provide instructions for enabling SSL.

4. Copy the `OpenText.Livelink.Service.PhysObj.dll` file from the *<ContentServer_Home>*`\webservices\dotnet\les-services-physicalobjects\bin` directory to the *<ContentServer_Home>*`\webservices\dotnet\les-services\bin` directory.
5. Copy the `PhysicalObjects.svc` file from the *<ContentServer_Home>*`\webservices\dotnet\les-services-physicalobjects` directory to the *<ContentServer_Home>*`\webservices\dotnet\les-services` directory.
6. Copy the following sections from the *<ContentServer_Home>*`\webservices\dotnet\les-services-physicalobjects\Web.config` directory to the specified sections of the *<ContentServer_Home>*`\webservices\dotnet\les-services\Web.config` directory:

   - **ServiceBehaviors**

     ```
     <behavior name="PhysicalObjectsBehavior">
     <serviceMetadata httpGetEnabled="true" httpGetUrl="" />
     <serviceDebug includeExceptionDetailInFaults="true"
     httpHelpPageUrl="" />
     </behavior>
     ```

   - **basicHttpBinding**:

     ```
     <binding name="PhysicalObjectsBinding"
     maxReceivedMessageSize="1024000" messageEncoding="Text"
     transferMode="Buffered">
     <security mode="None">
     <transport clientCredentialType="None"/>
     </security>
     </binding>
     <binding name="PhysicalObjectsBinding_SSL"
     maxReceivedMessageSize="1024000"
     messageEncoding="Text" transferMode="Buffered">
     <security mode="None">
     <transport clientCredentialType="None"/>
     </security>
     </binding>
     ```

   - **services**

     ```
     <service behaviorConfiguration="PhysicalObjectsBehavior"
     name="OpenText.Livelink.Service.PhysObj.PhysicalObjects_WCF">
     <endpoint address="" binding="basicHttpBinding"
     bindingConfiguration="PhysicalObjectsBinding"
     bindingNamespace="urn:PhysObj.service.livelink.opentext.com"
     contract="OpenText.Livelink.Service.PhysObj.IPhysicalObjects_W
     CF" />
     <!-- SSL disabled
     <endpoint address="" binding="basicHttpBinding"
     bindingConfiguration="PhysicalObjectsBinding_SSL"
     bindingNamespace="urn:PhysObj.service.livelink.opentext.com"
     contract="OpenText.Livelink.Service.PhysObj.IPhysicalObjects_W
     CF" />
     -->
     </service>
     ```

7. Restart IIS.

Note: You can test to ensure that Physical Objects is properly deployed by navigating to the following URL in a browser: `http://<server>:<port>/<ContentServer_Home>/PhysicalObjects.svc?wsdl`

3.2 Creating Physical Objects Using Web Services

You can create a Physical Item container, Physical Item Box, or Physical Item (non-container) using Web Services, by calling the *CreateNode* function from the *DocumentManagement* function in Content Server.

Note: The *CreateNode()* function in Content Server only works for registered node types. Physical Item container, Physical Item Box, or Physical Item (non-container) are registered node types.

You can update a Physical Item container, Physical Item Box, or Physical Item (non-container) using Web Services by called the *UpdateNode* function from the *DocumentManagement* function in Content Server.

3.2.1 To Create or Update Physical Items, Containers, and Boxes:

1. Specify one of the following:
   - For a Physical Item (non-container): `nodeSDO.fType="PhysicalItem"`
   - For a Physical Item (container): `nodeSDO.fType="PhysicalItemContainer"`
   - For a Physical Item Box: `nodeSDO.fType="PhysicalItemBox"`
2. To pass all metadata needed to create the Physical Item, you must call the Attribute Group, by specifying the following:
   - `theAttributeGroup.fKey="POCreateInfo"`
   - `theAttributeGroup.fType="POCreateInfo"`
3. In the Attribute Group, you can set the metadata that is passed to the Physical Item, container, or Box, as described in "Attribute Group Metadata" on page 34.
4. To pass Physical Properties, specify the following:
   - `mtField_<property_name>`

     Note: The *<property_name>* value depends on the Physical Property definition that is specified on the Physical Item type.

Table 3-1: Attribute Group Metadata

Metadata value	Value Type	Mandatory Value	Description
selectedMedia	Integer	Yes (valid only for creation)	DataID of Physical Object Type
poLocation	String	Yes	Corresponds to Home Location
poUniqueID	String	No (valid only for creation)	Unique ID of an item. If value is missing, the system will generate a new one.
poKeywords	String	No	Corresponds to PhysItemKeywords.PhysItemData
LocatorType	String	No	Corresponds to LocatorType.PhysItemData
RefRate	String	No	Corresponds to RefRate.PhysItemData
OffsiteStorID	String	No	Corresponds to OffsiteStorID.PhysItemData
Client_Name	String	No	Corresponds to Client.PhysItemData
TemporaryID	String	No	Corresponds to TemporaryID.PhysItemData
LabelType	String	No	Used to generate Label during creation. Corresponds to LabelDefName.LabelDefSpecs
Client_ID	Integer	No	ID of Content Server user assigned to Physical Item as client. Corresponds to ClientID.PhysItemData
NumberOfCopies	Integer	No (valid only for creation)	Number of copies of Physical Item (non-container) to be created. For Physical Item container and Physical Item Box, the value is always set to `1`.
NumberOfLabels	Integer	No (valid only for creation)	Number of Labels to be generated.
NumberOfItems	Integer	No (valid only for creation)	Number of Physical Items to be created.
generateLabel	Integer	No (valid only for creation)	Valid values: `0` – do not generate Label `1` – generate Label

poFromDate	Date	No	From Date for Physical Item container and Physical Item Box. Corresponds to FromDate.PhysItemData
poToDate	Date	No	To Date for Physical Item container and Physical Item Box. Corresponds to ToDate.PhysItemData

3.3 Physical Object Web Services Functions

3.3.1 PhysObjAssignLocator( )

The PhysObjAssignLocator( ) function assigns locators to physical item containers, physical item non-containers, and physical item boxes. This function will fail if any of the following occur:

- The item does not have a locator type assigned to it
- The selected locator is managed and the item does not fit the conditions
- The locator does not have enough space.

Parameters

{ "PhysObjLocatorAssignInfo AssignLocator","All metadata required to assign a locator."}

{"Boolean-return true if success", 'OUT'}

3.3.2 PhysObjAssignToBox( )

The PhysObjAssignToBox( ) function assigns physical item containers, physical item non-containers, and physical item copies to boxes. This function will fail if the box contains restrictions and the item is not a permitted type, the item is already assigned to a box, the parent of the item is already assigned to a box, the date range for the item does not fit within the date range for the box (depending upon the settings), or the user does not have permission to assign the item to the selected box.

Parameters

{"Long ItemID", "Data ID of physical item to be assigned to a box." 'IN'

{"PhysObjBoxingInfo BoxingInfo","Metadata needed to assign item to a box (Box ID,update RSI,update Status,update Location).", 'IN'

{"Boolean-return true if success", 'OUT'}

3.3.3 PhysObjAssignToTransfer( )

The PhysObjAssignToTransfer( ) function assigns transfers to physical item containers, physical item non-containers, and physical item copies. This function will fail if the item does not have a locator type assigned to it.

Parameters

{"String TransferID","Transfer ID of a transfer to be assigned.", 'IN'

{"Long ItemID", "Data ID of physical item to be assigned to a transfer.",'IN' }

{"Boolean bSynchDates","Flag indicating if From and To dates of a Transfer should be synchronized with From and To dates of the physical item.",'IN'

{"Boolean-return true if success", 'OUT'}

3.3.4 PhysObjAssignToTransferUID

The PhysObjAssignToTransferUID function assigns transfers to physical items using the item's unique ID.

Parameters

{"String TransferID","Transfer ID of a transfer to be assigned."}

{"String UniqueID", "Unique ID of physical item to be assigned to a transfer."}

{"Boolean bSynchDates","Flag indicating if From and To dates of a Transfer should be synchronized with From and To dates of the physical item."}

{"Boolean-return true if success", 'OUT'}

3.3.5 PhysObjBorrowedItems()

The PhysObjBorrowedItems() function return items borrower by specific user. If a user ID is not specified, the function returns items borrowed by a user who is logged in.

Parameters

{ "Long UserID","ID of a user for whom borrowed physical items are retrieved.",'IN'

{"PhysObjItemInfo[] PhysicalItemInfo","List of borrowed physical items with metadata.", 'OUT'}

3.3.6 PhysObjChargeIn( )

The PhysObjChargeIn( ) function cancels a physical item borrow or returns a physical item.

Parameters

{"Long ItemID","DataID of physical item to be borrowed.",'IN'

{"Integer Action", "Action to be performed. Expected values are: 2 - Cancel borrow,3 - Return",'IN'

{"PhysObjBorrowInfo ReturnInfo","Metadata needed to return physical item or cancel borrow action.",'IN'

{"Boolean-return true if success", 'OUT'}

3.3.7 PhysObjChargeOut( )

The PhysObjChargeOut( ) function charges out physical items to a Content Server user or an external client. This function will fail if any of the following occur:

- BorrowInfo.userName is missing
- The item is already borrowed
- The parent of the item is borrowed and the system settings do not allow the borrowing of children
- The borrower is a Content Server user who does not have See permissions
- The security override is required (because the external client or Content Server user does not have See Contents permissions) and BorrowInfo.SecurityOverride is not set to true.

Parameters

{"Long ItemID","DataID of physical item to be borrowed.",'IN'

{"PhysObjBorrowInfo BorrowInfo","Metadata needed to borrow physical item.",'IN'

{"Boolean-return true if success", 'OUT'}

3.3.8 PhysObjCreateLocator( )

The PhysObjCreateLocator( ) function creates locators for physical items.

Parameters

{"PhysObjLocatorInfo LocatorInfo","Metadata needed to create box locator.",'IN'

{"Boolean-return true if success", 'OUT'}

3.3.9 PhysObjCreateTransfer( )

The PhysObjCreateTransfer( ) function creates transfers for physical items.

Parameters

{ "PhysObjTransferInfo TransferInfo","Metadata needed to create a transfer.",'IN'

{"Boolean-return true if success", 'OUT'}

3.3.10 PhysObjFlagForPickUp()

The PhysIObjFlagForPickUp() function flags borrowed items listed in DataIDList parameter as ready to be picked up.

Parameters

{"Long[] ItemIDList", "List of Data ID of physical items for which flag for pickup should be updated.",'IN'

{"Boolean bAdd","Flag indicating if Flag For Pick Up should be added or removed.",'IN'

{"Boolean-return true if success", 'OUT'}

3.3.11 PhysObjFunctionMenu()

The PhysObjFunctionMenu() returns a list of functions available for physical items from the **Circulation** tab.

Parameters

{"Long ItemID", "Data ID of physical items for which circulation menu should be generated.",'IN'

{"String nextUrl","nextUrl to be appended to function menu items.",'IN'

{"String[]", "Array of strings representing menu items." OUT}

3.3.12 PhysObjGetAllRequests()

The PhysObjGetAllRequests() function returns all requests created for a specific item and an item's parent or box to which the item is assigned.

Parameters

{ "Long ItemID","Data ID of physical item for which requests should be returned.",'IN'

{"PhysObjItemRequests","List of item's requests (created for item, parent or box)" OUT}

3.3.13 PhysObjGetBoxContent()

The PhysObjGetBoxContent function returns all items that are assigned to a specific box.

Parameters

{"Long BoxID", "Data ID of physical item box for which content should be returned", 'IN'

{"Boolean bSubItems", "Flag indicating if only items directly assigned to a box should be returned (FALSE) or also sub-items (TRUE)." , 'IN'

{"DocMan.Node[]","Returns an array of nodes that are in the box." OUT}

3.3.14 PhysObjGetInfo()

The PhysObjGetInfo function returns additional borrow information.

Parameters

{ "Long ItemID", "The dataID of physical item.",'IN'

{"PhysObjItemInfo ItemInfo","All metadata of specific physical item." OUT}

3.3.15 PhysObjGetItemsHistory()

Parameters

{ "Long[] ItemIDList", "The list of physical items DataID.",'IN'

{"PhysObjHistoryInfo[] HistoryList","List of history info for specific physical items." OUT}

3.3.16 PhysObjGetMediaType( )

The PhysObjGetMediaType( ) function retrieves all Physical Item types information, such as DataID, Name, SubType (Physical Item Container, Physical Item Non-container, Physical Item Box), and Physical Properties.

Parameters

{"PhysObjMediaTypeInfo[] MediaTypeInfo", "Array of metadata about physical item types." OUT}

3.3.17 PhysObjGenerateLabel( )

The PhysObjGenerateLabel( ) function generates labels for locators or physical items.

Parameters

{"Long ItemID", "Data ID of physical items for which label should be generated",'IN'

{"String LabelType","The type of a label to be generated.",'IN'

{"Integer NumberOfCopies","Number of copies of the label to be generated.",'IN'

{"Boolean bLocator","Flag indicating for what object label should be generated. TRUE- label for locator, FALSE - label for physical item",'IN'

{"String Facility","Facility of locator for which label will be generated.",'IN'

{"String Area","Area of a locator for which label will be generated.",'IN'

{"String BoxLocator","Locator for which label will be generated.",'IN'

{"Boolean-return true if success", 'OUT'}

3.3.18 PhysObjGetAllLocators( )

The PhysObjGetAllLocators( ) function returns all locators.

Parameters

{"Integer StartAt", "The starting record of return array.",'IN'

{"Integer NumberOfRows","The number of locators to return.",'IN'

{"Boolean bAll","Flag indicating if all locators should be returned in one call.",'IN'

{"PhysObjLocatorInfo[] LocatorInfo","List of locators with metadata." OUT}

3.3.19 PhysObjGetAllTransfers( )

The PhysObjGetAllTransfers( ) function returns all transfers.

Parameters

{"Integer StartAt", "The starting record of return array.",'IN'

{"Integer NumberOfRows","The number of transfers to return.",'IN'

{"Boolean bAll","Flag indicating if all transfers should be returned in one call.",'IN'

{"PhysObjTransferInfo[] TransfersList","List of transfers with metadata." OUT}

3.3.20 PhysObjGetCodes( )

The PhysObjGetCodes( ) function returns the following physical item codes:

- Location
- Facility
- Area
- Locator Type
- Reference Rate
- Request Type
- External Client
- Recurrence
- Custodian Site

Parameters

{"Integer codesMask","the flag indicating if all codes or specific codes should be returned. values:"0 = all codes, 2 = Location,4 = Facility,8 = Area, 16 = LocatorType,32 = RefRate 64 = RequestType, 128 = ExternalClient, 264 = Recurrences, 512 =CustodianSite",'IN'}

{"PhysObjCodesInfo", "Codes from physical objects tables maintenance." OUT}

3.3.21 PhysObjRemoveFromBox( )

Parameters

{ "Long[] ItemIDList", "The list of physical items DataID.",'IN'

{"Long BoxID","The DataID of a box.",'IN'

{"Boolean-return true if success", 'OUT'}

3.3.22 PhysObjRemoveFromTransfer()

The PhysObjRemoveFromTransfer() function removes a physical item/items from a transfer.

Parameters

{"String TransferID","Transfer ID of a transfer to be removed.",'IN'

{"Long[] ItemIDList", "List of Data ID of physical items to be removed from a transfer. If the list is empty then all items are removed from this transfer.",'IN' }

{"Boolean-return true if success", 'OUT'}

3.3.23 PhysObjRemoveFromTransferUID

The PhysObjRemoveFromTransferUID function removes a physical item/items from a transfer using the Unique ID of the item.

Parameters

{"String TransferID","Transfer ID of a transfer to be removed.",'IN'

{"String[] UniqueIDList", "List of Unique ID of physical items to be removed from a transfer. If UniqueID is empty then all items are removed from the transfer.",'IN' }

{"Boolean-return true if success", 'OUT'}

3.3.24 PhysObjRequest( )

Parameters

{"Long ItemID","Data ID of physical item for which requests should be created or deleted.",'IN'

{"Integer Action","Flag indicating which action should be taken. Values: 5 = Create, 6 = Cancel Request",'IN'

{"PhysObjRequestInfo RequestInfo","Metadata needed to create or delete an requests.",'IN'

{"Boolean-return true if success", 'OUT'}

3.3.25 PhysObjGetInfoUID()

The PhysObjGetInfoUID function returns additional borrow information.

Parameters

{ "String UniqueID", "The UniqueID of physical item.",'IN'

{"PhysObjItemInfo ItemInfo","All metadata of specific physical item." OUT}

3.4 Physical Objects Service Data Objects

This section provides details for Physical Objects Service Data Objects (SDO).

3.4.1 PhysObjAreaCodes

The PhysObjAreaCodes SDO returns all area codes from the **Table Maintenance** section in the Physical Objects workspace.

Name	Type	Data Type	Description
fArea	String	String	The area code
fDescription	String	String	The description of the area code
fFacility	String	String	The facility code for a specific area

3.4.2 PhysObjBorrowInfo

The PhysObjBorrowInfo SDO represents all metadata needed to borrow a physical item.

Name	Type	Data Type	Description
fAcknowledge	Integer	Integer	The flag indicating if acknowledge is required. Possible values: Required = 1, Not Required = 2, Verified = 3, Manually verified = 4
fBorrowComments	String	String	The comments or reasons for borrowing
fBorrowDate	Date	Date	The date when item was borrowed
fLocation	String	String	The new location to be updated when item is borrowed
fObtainedBy	String	String	The name of a user who obtained physical item on behalf of another user

fObtainedByID	Long	Long	The ID of a user who obtained physical item on behalf of another user
fPass	Boolean	Boolean	The flag indicating if this is an action of new borrowing or a pass of already borrowed item
fReturnByDate	Date	Date	The date when the item is supposed to be returned
fReturnedComments	String	String	The comments added when an item was returned
fReturnedDate	Date	Date	The date when an item was actually returned
fReturnHome	Boolean	Boolean	A flag indicating if an item that is being returned should have the current location updated to its home location
fSecurityOverride	Boolean	Boolean	A flag indicating that the system should only allow users with the See permission to borrow the item
fUserID	Long	Long	The ID of a user who is borrowing the physical item
fUserName	String	String	The name of a user who is borrowing the physical item

3.4.3 PhysObjBoxingInfo

The PhysObjBoxingInfo SDO represents all metadata needed to add an item to the box.

Name	Type	Data Type	Description
fBoxID	Long	Long	The ID of a box to which the item will be assigned
fUpdateLocation	Boolean	Boolean	A flag indicating if the item location should be updated with the box location
fUpdateRSI	Boolean	Boolean	A flag indicating if an item's RSI should be updated with the box RSI
fUpdateStatus	Boolean	Boolean	A flag indicating if an item's Status should be updated with the box status.

3.4.4 PhysObjCodesInfo

The PhysObjCodesInfo SDO represents all codes information from the **Table Maintenance** section in the Physical Objects workspace.

Name	Type	Data Type	Description
fAreaCodes	List	PhysObjAreaCodes	An array of Area codes
fCustodianCodes	List	PhysObjCustodian Codes	An array of Custodian Site codes
fExternalClientCodes	List	PhysObjExternalClientCodes	An array of External Borrower or Box Client codes
fFacilityCodes	List	PhysObjFacilityCodes	An array of Facility codes
fLocationCodes	List	PhysObjLocationCodes	An array of codes used for Home and Current locations
fLocatorTypeCodes	List	PhysObjLocatorTypeCodes	An array of Locator Type codes
fRecurrenceCodes	List	PhysObjRecurrence Codes	An array of Recurrence Type codes
fRefRateCodes	List	PhysObjRefRateCodes	An array of Reference Rate codes
fRequestTypeCodes	List	PhysObjRequestTypeCodes	An array of Request Type codes

3.4.5 PhysObjCustodianCodes

The PhysObjCustodianCodes SDO returns all Custodian Site codes from the **Table Maintenance** section in the Physical Objects workspace.

Name	Type	Data Type	Description
fCustodianSite	String	String	The Custodian Site code
fDescription	String	String	The description of the Custodian Site code

3.4.6 PhysObjExternalClientCodes

The PhysObjExternalClientCodes SDO returns all codes for the Box Client and External Client from the **Table Maintenance** section in the Physical Objects workspace.

Name	Type	Data Type	Description
fAddress	String	String	The address of a client
fClient	String	String	The External Client code
fClientType	Integer	Integer	The type of a client. Possible values: 0-external borrower, 1-box client
fDescription	String	String	Description of a client code
fEmail	String	String	The email of a client
fPhone	String	String	The phone number of a client

3.4.7 PhysObjFacilityCodes

The PhysObjFacilityCodes SDO returns all facility codes from the **Table Maintenance** section in the Physical Objects workspace.

Name	Type	Data Type	Description
fAddress	String	String	The address of the facility
fCheckBoxFit	Integer	Integer	The mask indicating the value by which the facility is managed. Possible values: Depth=1, Length=2, Height=4, LocatorType=8, RefRate=16, Auto increment space=32
fDescription	String	String	The description of facility code
fFacility	String	String	The facility code
fManaged	Integer	Integer	The flag indicating if the facility is managed. Possible values: 0- not managed, 1- managed

3.4.8 PhysObjHistoryInfo

The PhysObjHistoryInfo SDO represents all metadata for the history of the circulation for a physical item.

Name	Type	Data Type	Description
fBorrowComment	String	String	The comments or reasons for borrowing
fBorrowPerformerID	Long	Long	The Content Server user who performed the borrow action
fBorrowPerformerName	String	String	The name of the Content Server user who performed the borrow action
fChargeDate	Date	Date	The date when the item was borrowed
fChargedTo	String	String	The name of the user who borrowed the physical item
fHID	Long	Long	The ID of the row in the History table
fName	String	String	The name of the physical item
fNodeID	Long	Long	The dataID of the physical item
fObtainedBy	String	String	The name of the Content Server user or external client who obtained the item on behalf of another user when borrowing
fObtainedByID	Long	Long	The ID of a user who obtained the item on behalf of another user when borrowing
fReturnBy	Date	Date	The date the item is to be returned
fReturnComment	String	String	The comments added when returning the item
fReturnDate	Date	Date	The date when the item was returned
fSecurityOverride	Integer	Integer	A flag indicating if an item was borrowed or obtained by a user with only the See permission
fUniqueID	String	String	the Unique ID of the item
fUserID	Long	Long	The ID of the user who borrowed the item

3.4.9 PhysObjItemInfo

PhysObjItemInfo represents all metadata of physical item.

Name	Type	Data Type	Description
fArea	String	String	The area of the locator if an item is assigned to a locator
fBorrowComment	String	String	Comments or reasons for borrowing
fBoxID	Long	Long	The ID of a box if item is assigned to a box
fBoxLocator	String	String	The Box Locator in specified Facility and Area if item is assigned to a locator
fBoxName	String	String	The name of a box if item is assigned to a box
fCanBeBorrowed	Boolean	Boolean	A flag indicating if item can be borrowed
fChargeDate	Date	Date	The date when item was borrowed
fChargedTo	String	String	The name of a user who borrowed physical item
fClient	String	String	The name of a box client (name of Content Server user or Client code from table maintenance)
fClientID	Long	Long	The ID of box client if box client is Content Server user
fCopyNumber	Integer	Integer	The copy number if item is of a non container type
fCParentID	Long	Long	The ID of a node which is a parent of physical item copy
fCurrentLocation	String	String	The temporary location of the item (different than default location in case item was borrowed)
fDateSent	Date	Date	The date when item was sent with the transfer
fDefaultLocation	String	String	The location where item resides
fFacility	String	String	The facility of a locator if item is assigned to a locator
fFromDate	Date	Date	The From date if item is container or a box

fLocatorType	String	String	The locator type code
fMediaTypeID	Long	Long	The ID of physical item type from which item was created
fMediaTypeName	String	String	The name of physical item type from which item was created
fMyNode	Node	Node	The node object which describe all meta data of Content Server object
fObtainedBy	String	String	The name of Content Server user or external client who obtained item on behalf of another user when borrowing
fObtainedByID	Long	Long	The ID of a user who obtained item on behalf of another user when borrowing
fOffsiteStorID	String	String	The additional storage ID
fPhysicalProperties	List	PhysObjProperties	The list of all physical properties assigned to the item and defined by physical item type
fPickUp	Integer	Integer	The flag indicating if item is ready to be returned
fRefRate	String	String	The reference rate code
fReturnedBy	Date	Date	The date item is to be returned
fSecurityOverride	Integer	Integer	The flag indicating if item was borrowed or obtained by the user with only See permissions
fTemporaryID	String	String	The temporary id assigned to the item
fToDate	Date	Date	The To date if item is container or a box
fTransferID	String	String	The ID of a transfer assigned to the item
fTransferReceived	Date	Date	The date when item was received with the transfer
fNodeID	Long	Long	The ID of the item
fUniqueID	String	String	The Unique ID of the item
fUserID	Long	Long	The ID of a user who borrowed the item

3.4.10 PhyObjItemRequest

PhysObjItemRequests represents all informations about requests created for item, item's parent and box."

Name	Type	Data Type	Description
fBoxRequests	List	PhysObjRequestInfo	An array of request information created for item's box
fItemRequests	List	PhysObjRequestInfo	An array of request information created for the item
fParentRequests	List	PhysObjRequestInfo	An array of request information created for the parent of the item

3.4.11 PhysObjLocationCodes

PhysObjLocationCodes returns all location codes from table maintenance"

Name	Type	Data Type	Description
fCustodianSite	String	String	The custodian site of specific location
fDescription	String	String	The description of location code
fDisabled	Integer	Integer	The flag indicating if location code is disabled
fEmail	String	String	The email of location
fLocation	String	String	The location code used for home location and current location

3.4.12 PhysObjLocatorAssignInfo

PhysObjLocatorAssignInfo represents all metadata needed to assign Locator to a box

Name	Type	Data Type	Description
fArea	String	String	The Area of a locator to be assigned.
fBoxID	Long	Long	The ID of physical item which will have locator assigned.
fFacility	String	String	The Facility of a locator to be assigned.
fLocator	String	String	The Box Locator in specified Facility and Area to be assigned.
fOffsiteStorID	String	String	The additional storage ID.

3.4.13 PhysObjLocatorInfo

"PhysObjLocatorInfo represents all metadata of a locator"

Name	Type	Data Type	Description
fArea	String	String	The area of a locator
fFacility	String	String	The facility of a locator
fFreeSpace	Real	Real	The free space in a locator
fGenerateLabel	Boolean	Boolean	The flag indicating if label should be generated when new locator is created
fLabelType	String	String	The type of a label to be generated
fLocator	String	String	The name of a locator
fLocatorSortKey	String	String	The zero filled string generated from Locator field, used for sorting
fLocatorType	String	String	The locator type code to match with box locator type
fLocatorType_List	String	String	The list of locator type codes if multiple values specified
fNumOfCopies	Integer	Integer	The number of label copies if label will be generated
fRefRate	String	String	The reference rate code to match with box reference rate
fStockType	String	String	The stock type of a label to be generated
fStorLocDepth	Real	Real	The depth of a locator to match with a depth of a box
fStorLocHeight	Real	Real	The height of a locator to match with a height of a box
fStorLocLength	Real	Real	The length of a locator to match with lenght of a box
fTotalSpace	Real	Real	The total space in a locator

3.4.14 PhysObjLocatorTypeCodes

PhysObjLocatorTypeCodes returns all locator type codes from table maintenance

Name	Type	Data Type	Description
fDepth	Real	Real	The depth of locator type
fDescription	String	String	The description of locator type code
fHeight	Real	Real	The height of locator type
fLength	Real	Real	The length of locator type
fLocatorType	String	String	The locator type code used when defining boxes
fTotal	Real	Real	The total space of locator type

3.4.15 PhysObjMediaTypeInfo

PhysObjMediaTypeInfo returns all information about physical item types

Name	Type	Data Type	Description
fMyNode	Node	Node	The node object which describe all meta data of Content Server object
fPhysicalProperties	PhysObjProperties	PhysObjProperties	The additional physical properties
fPhysObjSubType	Integer	Integer	The physical object sub type which can be created from that object. Possible values:424-Box, 412-Container,411- Non Container.

3.4.16 PhysObjProperties

PhysObjProperties represent all data assigned to physical object and defined in Physical Properties of Physical Item Type"

Name	Type	Data Type	Description
fDefaultVal	String	String	The default value for a popup field
fFieldDisplay	String	String	The text that the user sees for this field in the browser
fFieldName	String	String	The name of physical properties defined for physical item type and assigned to the item

fFieldNumber	Integer	Integer	The unique identifier of a field for any givin physical item type
fFieldOrder	Integer	Integer	The integer that identifies the order in which fields are displayed
fFieldRequired	Integer	Integer	The flag indicating if field is required
fFieldSize	Integer	Integer	The size of the field (relevant only for string and integer)
fFieldType	Integer	Integer	The flag that defines the type of field. Values: 1=String, 2=Integer, 3=Popup, 4=Date, 5= Boolean
fFieldValue	String	String	The value of physical properties assigned to the item
fMediaTypeID	Long	Long	The node ID of physical item type
fPopupVals	String	String	The list of user defined popup values

3.4.17 PhysObjRecurrenceCodes

PhysObjRecurrenceCodes returns all recurrence codes used during request creation

Name	Type	Data Type	Description
fDescription	String	String	The description of recurrence code
fNumberOfDays	Integer	Integer	The number of days between each request
fRecurrence	String	String	The recurrence code

3.4.18 PhysObjRefRateCodes

PhysObjRefRateCodes returns all reference rate codes from table maintenance

Name	Type	Data Type	Description
fDescription	String	String	The description of reference rate code
fRefRate	String	String	The reference rate code

3.4.19 PhysObjRequestInfo

PhysObjRequestInfo represents all metadata of a request

Name	Type	Data Type	Description
fAddress	String	String	The contact person's address
fContactPerson	String	String	The contact person's name for whom the request is being made
fDateOfRequest	Date	Date	The date user made a request
fEmail	String	String	The contact person’s email
fFax	String	String	The contact person’s fax
fNodeID	Long	Long	The ID of physical item for which request was created
fNumTimes	Integer	Integer	The number of times request need to be created
fPhone	String	String	The contact person’s phone number
fRecurrenceType	String	String	The recurrence type that defines the number of days between requests
fRecurring	Boolean	Boolean	The flag indicating that request need to be created multiple times
fRequestComment	String	String	The comments added when request was made
fRequestDate	Date	Date	The date user intends to borrow the item
fRequestDateFrom	Date	Date	The date which creates range for request date when deleting multiple requests
fRequestDateTo	Date	Date	The date which creates range for request date when deleting multiple requests
fRequestID	Long	Long	The unique key of a request
fRequestType	String	String	The request type code
fUserID	Long	Long	The ID of a user for whom request was created

3.4.20 PhysObjRequestTypeCodes

PhysObjRequestTypeCodes returns all request type codes from table maintenance

Name	Type	Data Type	Description
fDescription	String	String	The description of request type code
fRequestType	String	String	The request type code

3.4.21 PhysObjTransferInfo

PhysObjTransferInfo represents all metadata of a transfer

Name	Type	Data Type	Description
fTransferID	String	String	The ID of a transfer
fTransSortKey	String	String	The zero filled string generated from Transfer ID field, used for sorting
fTrsfComment	String	String	The comments about the transfer
fTrsfContent	String	String	The description of a type of content in the transfer
fTrsfCreate	Date	Date	The date when transfer was created
fTrsfDateFrom	Date	Date	The from date of the content of a transfer
fTrsfDateTo	Date	Date	The to date of a transfer
fTrsfEstSize	String	String	The estimated size of a transfer
fTrsfRecd	Date	Date	The date when transfer was received
fTrsfSent	Date	Date	The date when transfer was sent
fUserID	Long	Long	The ID of a user responsible for the transfer

Chapter 4

Security Clearance

4.1 Security Clearance Web Services Functions

This section provides a detailed explanation of each Web Services call in Security Clearance.

4.1.1 DefinedRuleAssign()

The DefinedRuleAssign() function creates and assigns a Security Clearance Defined Rule to an item.

Parameters

Input:

{"Integer NodeID", " node id of Content Server item." },

{"Integer ruleType", " integer indicating what action will be taken:1- Assign Supplemental Marking,2-Remove Supplemental Marking,3-Update Security Level"},

{"Date ruleDate", " date when defined rule is to be executed" },

{"String ruleValue"," for rule of type 1 or 2 this is supplemental marking code, for rule of type 3 this is string representation of security level"},

{"String ruleReason"," max 255 character long reason for the rule" }

Output:

A boolean represents the status of the update. The value is **True** if the update was successful, otherwise the value is **False**.

4.1.2 DefinedRuleEdit()

The DefinedRuleEdit() function edits a Security Defined Rule that is assigned to an item.

Parameters

Input:

{"Integer NodeID", " node id of Content Server item." },

{"Integer RuleID", " ID of a rule to be edited"},

{"Date newRuleDate", " new date when defined rule is to be executed" },

{"String newRuleReason"," new reason for the rule" }

Output:

A boolean represents the status of the update. The value is **True** if the update was successful, otherwise the value is **False**.

4.1.3 DefinedRuleGetAll()

The DefinedRuleGetAll() function returns a list of Security Defined Rules for an item.

Parameters

Input:

{"Integer NodeID", " node id of Content Server item." }

Output:

RMSecDefinedRule - List of security defined rules assigned to the item.

4.1.4 DefinedRuleRemove()

The DefinedRuleRemove() function removes Security Defined Rules from an item.

Parameters

Input:

{"Integer NodeID", " node id of Content Server item." },

{"Integer RuleID", " ID of a rule to be removed"}

Output:

A boolean represents the status of the update. The value is **True** if the update was successful, otherwise the value is **False**.

4.1.5 GetItemSec

The GetItemSec returns a node's Security Clearance information.

Parameters

Input:

Integer NodeID - node ID of a Content Server item.

Output:

An RMSecClearanceInfo object represents security clearance of the item.

4.1.6 GetItemSuppMarks

The GetItemSuppMarks retrieves supplemental markings of Content Server objects.

Parameters

Input:

An integer representing the ID of a Content Server user.

Output:

RMSecSuppMarkInfo - returns an array of supplemental markings assigned to a user.

4.1.7 GetSecSetting

The GetSecSetting returns the Security Clearance module's system settings.

Parameters

Input:

None

Output:

An RMSecSettingInfo object represents the Security Clearance module's settings.

4.1.8 GetSupportedSubTypes

The GetSupportedSubTypes returns supported subtypes information.

Parameters

Input:

None

Output:

RMSecSupportedSubType objects. These objects represent information about supported subtypes.

4.1.9 GetUserSec

The GetUserSec returns security levels available to a specific user.

Parameters

Input:

An integer represents a user's ID.

Output:

RMSecLevelInfo objects. These objects represent information about security levels.

4.1.10 GetUserSuppMarks

The GetUserSuppMarks retrieves the supplemental markings of a Content Server user.

Parameters

Input:

Node ID - An integer representing the Data ID of a Content Server object

Output:

None

4.1.11 IsServiceEnabled

The IsServiceEnabled function returns true if Security Clearance service is enabled.

Parameters

Input:

None

Output:

A boolean represents the status of the update. The value is **True** if the update was successful, otherwise the value is **False**.

4.1.12 SetItemSec

The SetItemSec sets a node's security clearance information.

Parameters

Input:

- An integer represents a user's ID
- A RMSecClearanceInfo object represents the new security clearance information

Output:

A boolean represents the status of the update. The value is **True** if the update was successful; otherwise the value is **False**.

4.1.13 SetItemSuppMarks

The SetItemSuppMarks assigns supplemental markings to a Content Server object.

Parameters

Input:

NodeID - An integer that represents the Data ID of a Content Server object.

Output:

A boolean represents the status of the update. The value is **True** if the update was successful, otherwise the value is **False**.

4.1.14 SetUserSec

The SetUserSec sets a user's security level.

Parameters

Input:

- An integer represents a user's ID
- An integer represents a security level

Output:

A boolean represents the status of the update. The value is **True** if the update was successful; otherwise the value is **False**.

4.1.15 SetUserSuppMarks

The SetUserSuppMarks sets user's supplemental markings.

Parameters

Input:

- An integer represents a user's ID
- Strings represent new supplemental markings

Output:

A boolean represents the status of the update. The value is **True** if the update was successful; otherwise the value is **False**.

4.2 Security Clearance Service Data Objects

This section provides details for Security Clearance Service Data Objects (SDO).

4.2.1 RMSecClearanceInfo

RMSecClearanceInfo is represents security clearance information of an item.

Name	Type	Data Type	Description
fLevel	Integer	Integer	The security level of an item
fDodInfo	RMSecDodInfo	Object	The DoD's metadata of an item
fSuppMarks	List	String	The list of supplemental markings of an item

4.2.2 RMSecDefinedRule

RMSecDefinedRule is represents information about a defined rule assigned to the item.

Name	Type	Data Type	Description
fRuleDate	Date	Date	The date when rule will be executed
fRuleID	Long	Long	The ID of a rule
fRuleType	Integer	Integer	The action to be executed, possible values 1- assign supplemental mark,2-remove supplemental mark,3-update security level

fRuleValue	String	String	The value for the rule, can be supplemental markings or security level
fRuleReason	String	String	Text explaining the reason for applying the rule

4.2.3 RMSecDodInfo

'RMSecDodInfo is represent information specific to DOD''s metadata.

Name	Type	Data Type	Description
fDeclassifiedOn	Date	Date	The date on which the declassification was completed
fDeclassifyDate	Date	Date	The date when declassification will occur
fDowngradeDate	Date	Date	The date when classification drowngrade will occur
fDowngradedOn	Date	Date	The date on which the downgrade was completed
fReviewedOn	Date	Date	The date on which the classification change was reviewed
fUpgradedOn	Date	Date	The date on which the classification was upgraded
fCurrentSecurity	Integer	Integer	The current security level
fDeclassifyType	Integer	Integer	The type of the declassification. Valid values are 1,2,3,4. Where 1 is date type, 2 is event type, 3 is date and event type, 4 is exemption category type'',
fDowngradeType	Integer	Integer	The type of downgrade. Valid values are 0,1,2,3. Where 0 is when DowngradeOn Date and DowngradeOn Event are not specified, 1 is is when DowngradeOn Date is specified and DowngradeOn Event is not specified, 2 is when DowngradeOn Date is not specified and DowngradeOn Date is specified, 3 is DowngradeOn Date and DowngradeOn Event are specified'',
fInitialSecurity	Integer	Integer	The initial security level

fClassifiedBy	String	String	The person responsible for setting security clearance
fClassifyingAgency	String	String	The agency responsible for setting security clearance
fDeclassifiedBy	String	String	The person responsible for the declassification
fDeclassifyEvent	String	String	The event which has triggered declassification
fDerivedFrom	String	String	The item from which clearance was inherited
fDowngradeEvent	String	String	The event which has triggered downgrade
fDowngradeInst	String	String	The instruction associate with the downgrade
fDowngradedBy	String	String	The person responsible for performing the downgrade
fExemptionCats	String	String	The categories for exemptions
fReasonsFor	String	String	The reasons for the change in security clearance
fReviewedBy	String	String	The person responsible for reviewing the change in classification
fUpgradeReasons	String	String	The reasons for the upgrade in classification
fUpgradedBy	String	String	The person responsible for the upgrade in classification

4.2.4 RMSecLevelInfo

'RMSecLevelInfo is represent information about a security level.', \

Name	Type	Data Type	Description
fLevel	Integer	Integer	The security level
fName	String	String	The name of the security level
fDescription	String	String	The description of the security level

4.2.5 RMSecSettingInfo

'RMSecSettingInfo is represents settings for Security Clearance module.

Name	Type	Data Type	Description
fDeclassifyOnFrame	Integer	Integer	The Declassify on time frame in years.
fDodFlag	Integer	Integer	Valid values are 0,1, and 2. Where 0 is when DOD off and it is the default, 1 is when DOD Chapter 2 is enabled, 2 is when DOD Chapter 3 is enabled
fSecOffGroupID	Long	Long	The Security Officer group ID
fFormRSuppMark	String	String	The formerly restricted supplemental marking"
fRestrSuppMark	String	String	The restricted supplemental marking"

4.2.6 RMSecSuppMarkInfo

'RMSecSuppMarkInfo is represent information about a supplemental marking.', \

Name	Type	Data Type	Description
fDescription	String	String	The description of a supplemental marking
fSuppMark	String	String	The supplemental marking

4.2.7 RMSecSupportedSubType

'RMSecSupportedSubtype is represents information about a supported subtype.

Name	Type	Data Type	Description
fContainer	Boolean	Boolean	The flag which mark the subtype is a container
fInheritable	Boolean	Boolean	The flag which allow the subtype to inherit security clearance to sub-items
fMandatoryLevel	Boolean	Boolean	The flag which indicates required security level for the subtype
fMandatoryMarkings	Boolean	Boolean	The flag which indicates required security markings for the subtype"
fObjType	Integer	Integer	The subtype ID
fName	String	String	the name of a subtype

Chapter 5

Classifications

This chapter provides an explanation of each function call in Classifications Web Services.

5.1 Classifications Web Services Functions

5.1.1 ApplyClassifications()

Applies a Classification to an node.

Parameters

{"Long NodeID", "DataID of node to be classified." }, \

{"Long[] ClassIDs", "array of class IDs to be applied." }\

"Boolean", "Returns success." \

5.1.2 GetItemClassifications

Returns a list of Classifications applied to a node.

Parameters

{\

{"Long NodeID", "DataID of node to have classifications removed." }, \

{"Boolean Accepted","Boolean indicating to retrun only accepted classifications"}\

, \

"ClassificationInfo[]", "List of classifications assigned to an object." \

5.1.3 GetManagedTypes

Returns a list of objects managed by a Classification.

Parameters

{\

, \

"ManagedTypeInfo[]", "List of managed types." \

5.1.4 RemoveClassifications

Removes a list of Classifications from a node.

Parameters

{\

{"Long NodeID", "DataID of node to have classifications removed." }, \

{"Long[] secClassIDs", "array of class IDs to be removed." }\

, \

"Boolean", "Returns success." \

5.2 Classifications Service Data Objects

This section provides details for Classifications Service Data Objects (SDO).

5.2.1 ClassificationInfo

Classification node assigned to the Content Server item

Name	Type	Data Type	Description
fSelectable	Boolean	Boolean	The flag indicating if classification is selectable
fMyNode	Node	Node	The classification node object which describe all meta data of Content Server object
fStatus	String	String	The status of classification describing if classification is accepted, pending or rejected

5.2.2 ManagedTypeInfo

ManagedTypeInfo represents a item's subtype that is being managed by Classification module

Name	Type	Data Type	Description
fSubType	Integer	Integer	An integer corresponding to sub type of managed node