Telescope – Integration Broker SDK Manual

Copyright © 2019 North Plains LLC (USA) and North Plains Systems Corp. (Canada), known as “Northplains”. All rights reserved.

North Plains, Telescope, Telescope OnDemand, I-Piece and all associated logos are trademarks or registered trademarks of Northplains. All other third-party product and company names mentioned in this document may be trademarks or registered trademarks of their respective owners.

The contents of this guide are for informational purposes only and are subject to change without notice. Northplains assumes no responsibility for any errors or omissions within this document. The material presented herein should not be construed as a commitment or warranty and it may not be copied or reproduced in any form, electronic or otherwise, without the express written consent of Northplains.

The software (including firmware) addressed in this guide is provided to the US Government under agreement that grants the government the minimum “restricted rights” in the software, as defined in the Federal Acquisition Regulation (FAR) or the Defense Federal Acquisition Regulation Supplement (DFARS), whichever is applicable.

If the software is procured for use by the US Department of Defense, the following legend applies. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

If the software is procured for use by any US Government entity other than the Department of Defense, the following notice applies. Notwithstanding any other lease or license agreement that may pertain to, or accompany the delivery of, this computer software, the rights of the Government regarding its use, reproduction, and disclosure are as set forth in FAR 52.227-19(C).

Unpublished rights reserved under the copyright laws of the United States.

Information and software in this document are proprietary to Northplains, its Distributors, or its Suppliers. No portion of this document may be copied, reproduced, disclosed to others, published, or used, in whole or in part, for any purpose other than that for which it is being made available. Use of software described in this document is subject to the terms and conditions of the North Plains Systems Software License Agreement. Third-party software license acknowledgments also apply, as listed in the Telescope Installation and Configuration Guide.

This document is for information purposes only and is subject to change without notice.

Northplains http://www.northplains.com/contact/ help@northplains.com

Version 9.4.0.17 (February, 2019)

Contents

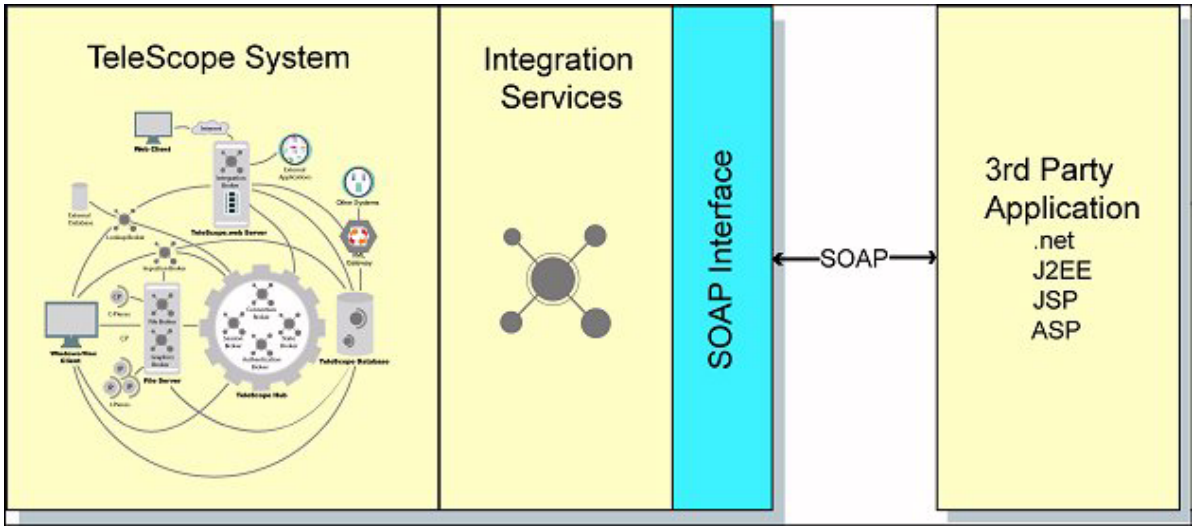
Chapter 1: Integration Broker SDK

This section provides an introduction to the Integration Broker and the Web services, SOAP messages, and the SOAP engine components needed to convey data to and from Telescope.web in a distributed environment.

In this Section:

1.1 Introduction

The Telescope Hub Integration Broker SDK is a functional API into the Telescope application. It is based on Web Services (SOAP) and provides full access to Telescope functionality.



This guide explains how to use the SOAP API and User Interface Service to integrate Telescope.web with other applications. It also explains how to deploy the sample client application and details the application's design. This guide assumes that you have a working knowledge of service-oriented architecture, Java™ or C++, and Simple Object Access Protocol (SOAP).

The APIs described in this document are broken down into major functional groups, based on the types of services they offer. This classification was made for convenience and readability only, and does not represent any physical breakdown of the API.

1.1.1 Why SOAP?

Simple Object Access Protocol is an XML-based model for accessing remote objects over a network. SOAP offers the following advantages:

- API-level access to Telescope functionality
- Authenticated and session-based (login/logout calls)
- Scalable (multiple instances)

About the User Interface Service

The UI Service makes it possible for external web-based applications to integrate with Telescope using a URL-based interface and a web browser on the client machine. The UI Service offers the following advantages:

- URL access to Telescope.web interface components
- Associated with the same session obtained from the SOAP Service (thus, authenticated)

1.2 Components of SOAP Integration

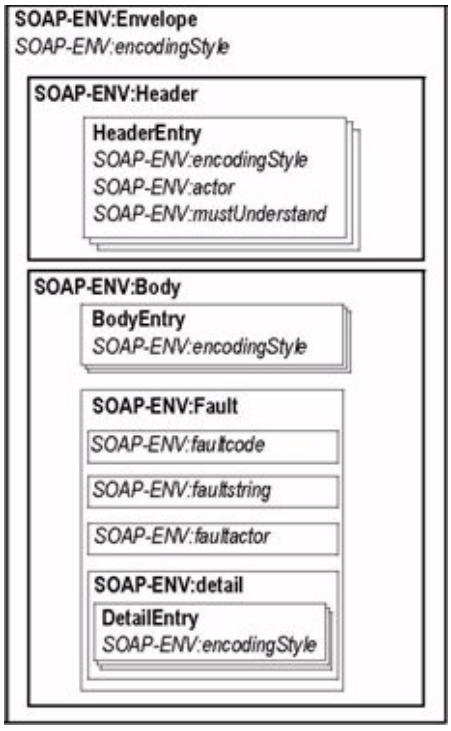
This topic deals with Web services, SOAP messages, and the SOAP engine. These components are needed to convey data to and from Telescope.web in a distributed environment.

1.2.1 Web Services

Web services can be thought of as distributed applications. They provide an implementation-independent way for applications to communicate with each other. Instead of creating an instance of a class and invoking its methods, a Web service consumer locates a Web service and invokes the operations it provides. The Web service provider (the application implementing the Web service) can be on the same Java virtual machine as the consumer using it or it can be thousands of miles away.

1.2.2 SOAP Messages

A SOAP message, represented by the Envelope element, contains a mandatory Body element and an optional Header element. The Body element can contain any number of body entries. The optional Fault element is present only in messages that report a processing exception. The structure of a SOAP message looks like this:



A SOAP message, represented by the Envelope element, contains a mandatory Body element and an optional Header element.

The Body element can contain any number of body entries. The optional Fault element is present only in messages that report a processing exception.

1.2.3 RPC-Style Messages

Telescope Integration Broker supports Remote Procedure Call (RPC) messaging, rather than document-style messaging. RPC messaging provides a standard way of representing method invocations in SOAP messages. Here's an example of an RPC SOAP message:

```
<soapenv:Envelope
     xmlns:soapenv="soap_ns"
     xmlns:xsd="xml_schema_ns"
     xmlns:xsi="type_ns">
<soapenv:Body>
     <ns1:getStockPrice
        xmlns:ns1="app_ns"
        soapenv:encodingStyle="encoding_ns">
     <stockSymbol xsi:type="xsd:string">AAPL</stockSymbol>
</ns1:getStockPrice>
</soapenv:Body>
```

1.2.4 Elements of a SOAP Message

The following table shows the elements of a SOAP message, its parent elements (if any), its uses, and its description:

Element	Parent	Use	Descrption
Envelope	None	Use only once	Encloses the message.
Header	Envelope	Optional	Encloses header entries.
Header Entries	Header	Can be used more than once	Provide additional information about the message's content, for example, digital signatures, authorization data, and so on.
Body	Envelope	Use only once	Encloses the message's body entries.
Body Entries	Body	Can be used more than once	Make up the content of the message. Their element names depend on the message's content.
Fault	Body	Optional	Reports a problem. When used, no other body entry can be present.
Faultcode	Fault	Use only once	Indicates the reason for the fault. Intended for application use.
Faultstring	Fault	Use only once	Provides a human-readable version of the fault description.
Faultactor	Fault	Optional	Indicates which entity along the message path raised the fault.

Element	Parent	Use	Descrption
Detail	Fault	Optional	Encloses detail entries.
Detail Entries	Detail	Can be used more than once	Contain application-specific information about the fault.

1.2.5 Attributes of a SOAP Message

All of the attributes that the SOAP envelope schema defines are global (they are not associated with a particular element). Also, each element in a SOAP message is free to use any attribute, regardless of where it is defined, either in SOAP's schema or another schema. This is one of SOAP's extensibility features because elements are free to use any number of attributes. The following table describes the attributes that the SOAP specification defines.

Attribute	Value	Description
Actor	URI	Specifies the entity that is to process the element. When absent, the actor is the ultimate recipient of the message. This attribute is used mainly to assign header entries to specific entities.
mustUnderstand	0 or 1	Indicates whether the element's actor must process the element. When set to 1 and if the actor is unable to process the element, the actor must respond with a fault.
encodingStyle	List of URIs	Indicates the encoding style used for the element’s content.

1.2.6 The SOAP Engine

A SOAP engine (or processor) helps both consumers of Web services and their providers accomplish tasks without having to know the intricacies of SOAP message handling. As far as the consumer is concerned, it invokes an operation in a way similar to how a remote procedure call is invoked. The Web service provider needs to implement only the logic required by the business problem it solves.

The consumer's SOAP processor converts the method invocation into a SOAP message. This message is transmitted through a transport, such as HTTP or SMTP, to the service provider's SOAP processor, which parses the message into a method invocation. The provider then executes the appropriate logic and gives the result to its SOAP processor, which parses the information into a SOAP response message. The message is transmitted through a transport to the consumer. Its SOAP processor parses the response message into a result object that it returns to the invoking entity.

The Apache Axis SOAP engine provides and consumes Web services. Axis is the third generation of Apache SOAP (an implementation of SOAP from the Apache Software Foundation). Axis is a SOAP engine as well as a code generator and WSDL processing tool. It provides a framework for constructing SOAP processors such as clients, servers, gateways, and so on, which are integrated into the existing application so that message handling, invoking the service, and generating responses are performed using the Axis engine.

Chapter 2: Deploying the Integration Broker

In this Section:

2.1 Integration Broker Support in Telescope

You should specify the Default Group for creating multiple users in the SOAPParams.plist file (by default, this file is located in ...\tsweb.woa\Contents\Resources). In this file, a key called TEMPLATE_GROUP defines the default group that a new user is placed in if the in_strGroupName parameter is not passed with the SOAP API CreateUser() method. The default value for this setting is Default, which causes all new users to be created as members of the Default group.

You might want to use the support files and programming resources provided with the Integration Broker SDK as you develop an application that will interface with Telescope.

2.2 SDK Support Files

The WSDL files in the "programming resources" folder describe the Telescope.web services in a service-specific language. These files can be used with the WSDL2Java utility (available from apache.org) to generate Java stubs. The stubs can then be used to connect to the Integration Broker from a third-party application.

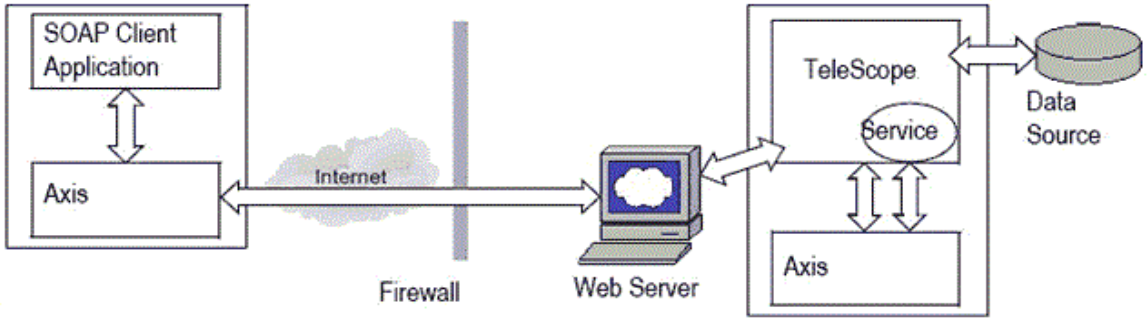
Also included in this folder are Javadocs that describe the main Integration Broker service methods and the North Plains Systems internal data types.

The sample client applications are also a valuable resource for understanding the Integration Broker SDK and how to use it.

2.3 Sample Client Applications

Two sample client applications are provided to demonstrate and test the SOAP and UI Services provided by Telescope.

The following diagram shows the typical deployment architecture for a client application using the SOAP API provided by Telescope:



2.3.1 Simple Command Line Java Application

The Sample_Client folder provided with the Integration Broker ZIP file contains a simple command line application that demonstrates some of the Integration Broker SDK methods in action. To run the application, copy the files in the Sample_Client\executables folder to a location on the local machine and run the run.bat file.

2.3.2 Simple C# Application

The IB2REVISED\samples\DemoApp folder created when you extracted the Integration Broker ZIP file contains a simple C# application with a basic graphical user interface that demonstrates some of the Integration Broker SDK methods in action. To run the application, run the WindowsApplication1.exe file in the DemoApp folder.

2.4 Working WIth Functional Rules

For calls in the Integration Broker API that execute functional rules, if the user's rules return a challenge form, this is treated as a failure, with the error message: "Challenge forms are not supported in functional rules using the SOAP API".

Functional rules are executed for each asset handled by the call, and a functional rule execution failure will result in an error string being added to the array of error strings for return to the caller, but will not raise an exception.

2.5 Accessing Telescope

Since Telescope can define multiple customized "sites", the ability to pass the name of the site when connecting to Telescope is crucial. To pass the site name, include it in the SOAP Login endpoint URL when making the initial login call to the Integration Broker as indicated below. The site name is stored in the session that is created via the Login() method so all UI elements from Telescope accessed via the Integration Broker will reflect the site name that was passed in. If the site parameter is absent, or an invalid site name is passed, the default site is assumed, as dictated by the standard behavior of Telescope.

2.5.1 Passing the Site Parameter When Accessing Telescope

To pass the site parameter, modify the endpoint URL used to access the Integration Broker, as demonstrated by the examples below.

Without site parameter:

http://www.hostserver.com/scripts/WebObjects.dll/tsweb.woa/wa/services/Login

With site parameter:

http://www.hostserver.com/scripts/WebObjects.dll/tsweb.woa/wa/services/Login?site=ABC123

2.5.2 Calling the Integration Broker

Keep the following points in mind when calling the Integration Broker:

The first call must always be the SOAP Login call; this call returns two URLs: a new SOAP endpoint URL, and a URL for accessing the UI Service.

The session ID is automatically embedded in the URLs by the Integration Broker.

All subsequent SOAP and UI service calls must use these endpoint URLs.

The Logout call terminates the session.

The session is maintained on the server side and expires after a period of time if no further calls are made; the greeting call can be used periodically to keep the session alive if necessary.

2.5.3 Functional Rules and the Integration Broker

Once logged in, interacting with Telescope.web using the Integration Broker is identical to working in the application directly. Any functional rules configured for the user you are logged in as will run when you perform the action that triggers them. However, functional rules that use challenge forms or approvals are not supported with the Integration Broker. Calls that trigger such functional rules will return an error.

Chapter 3: Telescope Hub SOAP API

This section provides information about the Telescope SOAP API.

In this Section:

3.1 SOAP API Methods Overview

The chart below lists the classes and methods supported by the SOAP API. The Login method must be used first in any session because it provides a starting point for other SOAP services.

Class:Login

This class provides the SOAP API used as a starting point to log in to the TSWeb domain to use other TSWeb SOAP Services.

Method	Description
EnumerateConnections	Returns the database connection(s) as defined in the 'site.plist' file.
Login	Validates the user's identifier and password in the specified database. This is the first method that must be called in each session.
LoginWithProvider	This method must be called before any other method. It validates the user against the specified data source, using an authentication provider class.

Class: TSWebService

This Java class provides the SOAP API for interface between SOAP clients and the TSWeb domain.

Method	Description
AddToCatalog	Adds one or more assets to a cataog.
CancelCheckout	Cancels the check out action performed on an asset.
Checkin	Checks in a single file.
CheckinWithData	Checks in a single file with data. The actual file must passed as DIME attachment.
Checkout	Checks out an asset.
CheckOutStatus	Checks the value of the editorial.status column for the asset.
ConvertMimix2Onix	Converts the provided MIMiX XML to ONIX using the supplied XLST file. If no XSLT file is indicated, the default file is used
CreateCatalog	Creates a new empty catalog.
CreateDerivativeFromVersion	Creates a derivative of the current version.
CreateUser	Creates a new user with the specified user ID and password.
Delete	Deletes an asset, first making sure that the user is allowed to delete assets.
DeleteCatalog	Deletes a catalog. The logged-in user must either be the owner of the catalog or an administrator who has visibility permission for the catalog owner's group.
DeleteMBMessage	Deletes one or more messages.
DeleteUser	Deletes the Telescope user.

Method	Description
DeleteVersion	Deletes the indicated version.
Download	Downloads one or more assets, executing functional rules if required.
DownloadAged	Downloads a version of one or more files prior to a given date.
DownloadAgedStart	Downloads one or more files asynchronously, attempting to retrieve a version that was checked in prior to a given date.
DownloadAgedStatus	Returns the aged values of the downloading version.
DownloadPageURL	Returns URL of Download Manager page for downloading selected assets.
DownloadStart	Downloads one or more files asynchronously.
DownloadStatus	Returns the asset's download status.
DownloadVersion	Downloads a specific version of an asset.
DownloadVersionStart	Downloads a specific version of one or more assets asynchronously.
DownloadVersionStatus	Returns the asset version's download status.
EnumerateCatalogs	Returns the catalogs nested in the given catalog structure.
EnumerateFields	Returns a list of the metadata fields visible to the logged-in user.
EnumerateGroups	Returns a list of the user groups visible to the logged in user.
EnumerateUsers	Returns a list of the users visible to the logged in user.
GetAssetVersions	Returns a list of versions for the given asset.
GetCatalogAssets	Returns a list of assets in the given catalog; only assets visible to the logged-in user are returned.
GetCatalogProperties	Returns the properties information for the given catalog.
GetData	Returns the value of a metadata field.
GetDataMultiple	Returns the values of one or more metadata fields for one or more assets.
GetFile	Returns a link to a file so that it can be downloaded. Indicates if the file should be zipped on download.
GetIngestStatus	Queries ingest status associated with a unique request/process ID.
GetMBMessageAction	Called when the recipient clicks on one of the message action buttons attached to a 'To Do' message.
GetMBMessageCount	Returns the number of messages in the system for the logged-in user from the given cut-off date.
GetMBMessageList	Returns the message IDs and header information for the messages in the given category for the logged-in user.

Method	Description
GetMBVisibleActions	Returns the message actions (from the M_ACTIONS table) that are visible to the calling user.
GetOnixDataMultiple	Generates an XML string in ONIX format from the data in the given fields.
GetTBCriteriaValues	Collects array values of the tree model for the current user from the Tree Broker.
GetTBLevelData	Returns the data for a given tree search from the TreeSearch Broker.
GetTBSearchNames	
GetTemplateNames	Returns all of the names of the templates currently saved in the database.
greeting	Returns a welcome message and resets the session timeout. Used to 'heartbeat' the application so that the session does not expire.
Ingest	Uploads the specified files, fills in metadata fields, saves the assets, and returns a list of record IDs of the uploaded files.
IngestWithData	Passes the binary data for an ingested file.
IngestWithStatus	Ingests multiple assets into the Telescope database asynchronously.
IngestWithTemplate	Uploads the specified files, fills in metadata fields from the specified template, saves the assets, and returns a list of record IDs.
IngestWithTemplateAndData	Passes the binary data for the ingested file with its template.
InvalidSession	Handles an invalid SOAP session.
IsValidSession	Checks whether a given sessionId is valid.
IsValidUser	Performs a search for a user in the database.
Logout	Terminates the user's session and cleans up temporary files.
PromoteVersion	Promotes the given version of an asset.
ReadMBMessage	Returns the full information from a given message.
RemoveFromCatalog	Removes one or more assets from a catalog.
Search	Validates the criteria list, executes the search, and returns the search results as an array of record IDs.
SendMBApprovalMessage	Sends a Telescope 'approval' message.
SendMBMessage	Sends a Telescope message.
SendMessage	Sends a Telescope message to the specified user with the subject and body provided.
SetCatalogProperties	Sets the given catalog's properties.
SetData	Sets one of an asset's metadata fields to a specified value.
SetDataMultiple	Sets the data for one or more assets.
UpdateUserPassword	Replaces the password of the specified user.

Chapter 4: Asset Maintenance Methods

The asset maintenance methods include methods for checking out, checking in, and deleting assets from the Telescope.web database.

In this Section:

4.1 CheckOut

public void CheckOut	**(**	Integer in numRecordId,
		Integer in numRenditionId)
	throws	Throwable

This call adds a lock to the given file, preventing other users from checking it out. The file is not actually copied, simply locked. The Download method can be used to obtain a copy of the file after it has been locked. It is not necessary to lock a file before downloading, but this is recommended if the file is modified before being returned to Telescope.

Before checking out the file, this method validates the privileges for the current user, and if they are sufficient, proceeds with checkout. If the current user doesn't have enough privileges to proceed with checkout for the given record ID and rendition ID, this method throws an exception.

Parameters:

- in_numRecordId – The record ID of the file to be checked out.
- in_numRenditionId – The rendition ID of the file to be checked out.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

Soap UI example call:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:impl="http://com/northplains/web/tsweb/soap/impl">
   <soapenv:Header/>
   <soapenv:Body>
      <impl:CheckOut soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <in_numRecordId xsi:type="soapenc:int" xmlns:soapenc="http://schemas.xmlsoap.org/soap/
encoding/">34327</in_numRecordId>
        <in_numRenditionId xsi:type="soapenc:int" xmlns:soapenc="http://schemas.xmlsoap.org/soap/
encoding/">1</in_numRenditionId>
      </impl:CheckOut>
   </soapenv:Body>
</soapenv:Envelope>
```

Result:

200 - Asset should be checked out, verify on TSWeb UI

Example Code Call:

The following example code call uses the SOAP example call above.

```
TSWebServiceService service = new TSWebServiceServiceLocator();
service.createCall();
TSWebService telescopeService = service.getsoapservice(new java.net.URL(strEndPoint));
```

```
telescopeService.checkOut(recordId, rendId);
```

4.2 CheckOutStatus

public NPSMap CheckOutStatus	**(**	Integer in numRecordId,)
	throws	Throwable

This method gets the checkout status of the asset.

Parameters:

- in_numRecordId – The record ID of the asset.

Returns:

An NPSMap object that contains the status of the asset being checked out. If an asset is not checked out this method returns an NPSMap with a single key ISCHECKOUT and a Boolean object for it indicating the status.

If an asset is checked out this method returns an NPSMap with the following keys containing the values found in the CHECKOUTS table for the given record ID:

- ISCHECKOUT
- USER_NAME
- CHKOUTDATE
- RENDITIONS

If the asset is checked out but some of the values (or records in the checkouts table) are missing the NPSMap object will have null values for the keys for which the value wasn't found in the database. The value for the key RENDITIONS is a type of java.util.ArrayList that contains all checked out renditions (as java.lang.Number objects) found in the checkouts table. The value for the key CHKOUTDATE is a type of java.lang.String in the format yyyy/MM/dd HH:mm:ss.

NOTE: When accessing any value in the NPSMap, except for the key ISCHECKOUT, the returned result should be checked for null as this method can return some values as null for the keys declared above.

NOTE: This method neither validates visibility of the user name that checked out the asset, nor the visibility of the rendition IDs returned in the NPSMap against the current user name. However, the visibility validation of the asset for the current user is performed.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

4.3 CancelCheckOut

public void CancelCheckOut	**(**	Integer in numRecordId,)
	throws	Throwable

This call removes the lock from a record which had previously been checked out.

This method first validates the current user privileges, and if they are sufficient, removes all entries in the CHECKOUTS table and changes the status of the record in the EDITORIAL table to null.

If the current user doesn't have enough privileges to cancel the checkout, this method throws an exception.

Parameters:

- in_numRecordId – The record ID of the asset whose checkout state should be changed.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

4.4 Checkin

public void Checkin	**(**	(int in_iRecordId,
		int in_iRendId,
		String in_strUri,
		boolean in_bIsMac,
		String in_strVersionName,
		String in_strVersionDesc)
	throws	Throwable

This method performs a check-in process for a single file. The file resides on a server accessible over the Web, and its URL is included so that it can be found and uploaded.

Note:

- This method is overloaded with a deprecated method with the same name (but a different parameter set) that was used before version 9.4.0.11. If the deprecated parameter signature is detected, the checkin will not be completed; instead, a message will be shown to indicate a deprecated method is being used.

Parameters:

- in_iRecordId – Record ID of the asset to check in.
- in_iRendId – Rendition ID. The rendition ID must be visible to the user's group and a record in the doc_renditions table must be present.
- in_strUri - String URL. A URL defining a link to the asset's location where it can be fetched. The file at this link is uploaded. Its name at the original location is used.
- in_bIsMac – True if the file is in the Mac bin format.
- in_strVersionName – A name for the version. This value cannot be null and must not exceed 16 characters in length.
- in_strVersionDesc – (Optional) A version description; if specified, it must not exceed 255 characters in length

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

Sample SOAP UI Call:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:impl="http://com/northplains/web/tsweb/soap/impl">
   <soapenv:Header/>
   <soapenv:Body>
      <impl:Checkin soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
         <in_iRecordId xsi:type="xsd:int">34327</in_iRecordId>
         <in_iRendId xsi:type="xsd:int">1</in_iRendId>
         <in_strUri xsi:type="soapenc:string" xmlns:soapenc="http://schemas.xmlsoap.org/soap/
              encoding/">http://some/path/to/file.jpg</in_strUri>
         <in_bIsMac xsi:type="xsd:boolean">false</in_bIsMac>
         <in_strVersionName xsi:type="soapenc:string" xmlns:soapenc="http://schemas.xmlsoap.org/
              soap/encoding/">1.0</in_strVersionName>
         <in_strVersionDesc xsi:type="soapenc:string" xmlns:soapenc="http://schemas.xmlsoap.org/
              soap/encoding/">version change from soap call</in_strVersionDesc>
      </impl:Checkin>
   </soapenv:Body>
</soapenv:Envelope>
```

Result:

200 - Asset should be checked in, verify on TSWeb UI

Example Code Call:

The following example code call uses the SOAP example call above.

(The first three lines only need to be done once per session.)

```
TSWebServiceService service = new TSWebServiceServiceLocator();
service.createCall();
TSWebService telescopeService = service.getsoapservice(new java.net.URL(strEndPoint));
telescopeService.checkin(intRecordId, intRendId, strFileURL, bIsMac, strVersionName,
    strVersionDescription);
```

4.5 CheckinWithData

```
public void CheckinWithData      (    int in_iRecordId,
                                      int in_iRendId,
                                      String in_strFileName,
                                      boolean in_bIsMac,
                                      String in_strVersionName,
                                      String in_strVersionDesc
                                 throws    Throwable
```

This method performs a check-in process for a single attached file. The file has to be passed as DIME attachment. For more information about passing binary data, see Section 7.1, "API Methods: Ingestion," on page 61.

Notes:

- This method is not recommended, because DIME attachments may have performance issues, especially for large files where some applications (specifically those written in C#) do not use connection timeouts. It is recommended that you use the Checkin method instead (with a URL to an asset in a network-accessible location).
- This method is overloaded with a deprecated method with the same name (but a different parameter set) that was used before version 9.4.0.11. If the deprecated parameter signature is detected, the checkin will not be completed; instead, a message will be shown to indicate a deprecated method is being used.

Parameters:

- in_iRecordId – Record ID of the asset to check in.
- in_iRendId – Rendition ID. The rendition ID must be visible to the user's group and a record in the doc_renditions table must be present.
- in_strFileName – The file name, used as a file name when InputStream data is saved.
- in_bIsMac – True if the file is in the Mac binary format.
- in_strVersionName – A name for the version. This value cannot be null and must not exceed 16 characters in length.
- in_strVersionDesc – (Optional) A version description; if specified, it must not exceed 255 characters in length

Throws

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

Sample SOAP UI call:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:impl="http://com/northplains/web/tsweb/soap/impl">
   <soapenv:Header/>
   <soapenv:Body>
      <impl:CheckinWithData soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <in_iRecordId xsi:type="xsd:int">4623</in_iRecordId>
         <in_iRendId xsi:type="xsd:int">3</in_iRendId>
        <in_strFileName xsi:type="soapenc:string" xmlns:soapenc="http://schemas.xmlsoap.org/soap/
           encoding/">movie.mp4</in_strFileName>
         <in_bIsMac xsi:type="xsd:boolean">false</in_bIsMac>
         <in_strVersionName xsi:type="soapenc:string" xmlns:soapenc="http://schemas.xmlsoap.org/
            soap/encoding/">s</in_strVersionName>
         <in_strVersionDesc xsi:type="soapenc:string" xmlns:soapenc="http://schemas.xmlsoap.org/
            soap/encoding/">0</in_strVersionDesc>
      </impl:CheckinWithData>
   </soapenv:Body>
</soapenv:Envelope>
```

Result:

200 - Asset should be checked in, verify on TSWeb UI

Example Code Call:

The following example code call uses the SOAP example call above.

```
TSWebServiceService service = new TSWebServiceServiceLocator();
service.createCall();
TSWebService telescopeService = service.getsoapservice(new java.net.URL(strEndPoint));
File attachedFile = new File(in_strFileName);
org.apache.axis.client.Stub telescopeStub = (org.apache.axis.client.Stub)telescopeService;
telescopeStub._setProperty(org.apache.axis.client.Call.ATTACHMENT_ENCAPSULATION_FORMAT,
   org.apache.axis.client.Call.ATTACHMENT_ENCAPSULATION_FORMAT_DIME);
String strFileName = attachedFile.getName();
DataHandler attachedFileDH = new DataHandler(new FileDataSource(attachedFile));
telescopeStub.addAttachment(attachedFileDH);
telescopeStub.setTimeout(12000);
iReturnValue = telescopeService.checkinWithData(intRecordId, intRendId, strFileName, bIsMac,
   strVersionName, strVersionDesc);
```

4.6 Delete

```
public String[ ] Delete          (    Integer in_numRecordId,
                                      boolean in_bDeleteOriginalFile)
                          throws      Throwable
```

This method deletes the asset represented by the given record ID. If the current user doesn't have enough privileges to delete the asset, this method throws an exception. This method also deletes the original files if the parameter in_bDeleteOriginalFile is included and set to true.

Parameters

- in_numRecordId – The record ID of the asset to be deleted. (Required)
- in_bDeleteOriginalFile – If set to true, deletes the original files as well. (Optional)

Examples

- Delete (12345) – Deletes asset 12345 (but not the original files)
- Delete (12345, T) – Deletes asset 12345 and its original files

Returns

- Returns an array of files that could not be deleted because they are missing or referenced by other assets. (Occurs when in_bDeleteOriginalFile is set to true and there is an error.)

Throws

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

4.7 SetThumbnailByCode

```
public boolean SetThumbnailByCode ( int record_id,
                                    String type_code)
throws                              Throwable
```

The method sets a thumbnail for an asset to a standard existing thumbnail based upon a preconfigured *type_code*. The *type_code* is created from the TSAdmin interface (File Types section) with thumbnail upload.

Parameters:

- record_id - The asset record_id
- type_code - The thumbnail type code

Returns:

Nothing.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

4.8 AttachRendition

public int AttachRendition	**(**	String in_strUrlToFile
		integer in_numRecordId
		integer in_numRendId
		int in_iIsMac
		String in_strMigrationPolicy)
	throws	Throwable

Attaches a rendition to an existing asset.

This method attaches a single rendition by calling the Ingest Broker. This is a synchronized call and might take some time before the process is finished. A single rendition can be attached only if it is visible to the user and free from another record.

Parameters

- in_strUrlToFile —The URL to the rendition file. In a form like http://[domain name]/[path]/[file name]
- in_numRecordId – The record ID of the asset to which the new rendition should be attached. The record ID value is validated against the user's visible assets, and has to be visible.
- in_numRendId – The rendition ID for the rendition to attach. If this value is null, zero or a negative value, the code will try to attach a provided file to the lowest visible rendition if it exists and is visible to the user. If the rendition ID is not defined (in the renditions table) or not visible to the current user, an exception is thrown. The rendition must be empty. It is not allowed to replace or update an existing rendition in this call.
- in_iIsMac – Pass 1 if the file is in Mac binary format; otherwise, 0. If this value is equal to 1 (one), the downloaded file is assumed to be in Mac binary format and should have a file name in the form [file name].[file extension].bin.
- in_strMigrationPolicy—Any applicable Migration Policy. Use a migration policy if the file has to migrated. If this value is null or an empty string, the privileges for the current user are verified to ensure they can import with no migration. If this value is not null, the current user must have visibility privileges for the provided migration policy. If the user privileges cannot be validated, an exception is thrown.

Returns:

- The attached rendition ID.

Throws

- Throwable in case of any error or if fails to validate input data or user privileges.

Sample SOAP UI call:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:def="http://DefaultNamespace">
  <soapenv:Header/>
```

```
  <soapenv:Body>
    <def:AttachRendition soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <in_strUrlToFile xsi:type="xsd:string">?</in_strUrlToFile>
      <in_numRecordId xsi:type="xsd:int">?</in_numRecordId>
      <in_numRendId xsi:type="xsd:int">?</in_numRendId>
      <in_iIsMac xsi:type="xsd:int">?</in_iIsMac>
      <in_strMigrationPolicy xsi:type="xsd:string">?</in_strMigrationPolicy>
    </def:AttachRendition>
  </soapenv:Body>
</soapenv:Envelope>
```

Example Code Call:

The following example code call uses the SOAP example call above.

```
TSWebServiceService service = new TSWebServiceServiceLocator();
service.createCall();
TSWebService telescopeService = service.getsoapservice(new java.net.URL(strEndPoint));
File attachedFile = new File(in_strFileName);
org.apache.axis.client.Stub telescopeStub = (org.apache.axis.client.Stub)telescopeService;
telescopeStub._setProperty(org.apache.axis.client.Call.ATTACHMENT_ENCAPSULATION_FORMAT,
   org.apache.axis.client.Call.ATTACHMENT_ENCAPSULATION_FORMAT_DIME);
String strFileName = attachedFile.getName();
DataHandler attachedFileDH = new DataHandler(new FileDataSource(attachedFile));
telescopeStub.addAttachment(attachedFileDH);
telescopeStub.setTimeout(12000);
iReturnValue = telescopeService.attachRendition(in_strUrlToFile, in_numRecordId, in_numRendId,
in_iIsMac, in_strMigrationPolicy);
```

4.9 PopulatePopupValues

The `PopulatePopupValues` SOAP API call populates popup values for multilanguage support. The call must be issued separately for every popup menu contained in the `popups` table, for each language. For more information on multilanguage support, see the *Administrator's Guide*.

<impl>`PopulatePopupValues`	`in_arNewPopupsValues`
SEE EXAMPLE BELOW FOR FULL IMPLEMENTATION	(array)
	`in_iColumnId (`
	(integer)
	in_strLangId (standard language code)
	`</impl>PopulatePopupValues`
throws	Throwable

Parameters:

- `in_arNewPopupsValues` array—contains the translations for each menu item. This array must contain exactly the same number of popup items, in exactly the same order, as appear for the default language in the `popup` table.

  Example:

  <in_arNewPopupsValues xsi:type="impl:ArrayOf_soapenc_string">

  <in_arNewPopupsValues xsi:type="xsd:string">un</in_arNewPopupsValues>

  <in_arNewPopupsValues xsi:type="xsd:string">deux</in_arNewPopupsValues>

  <in_arNewPopupsValues xsi:type="xsd:string">trois</in_arNewPopupsValues>

  </in_arNewPopupsValues>

- `in_iColumnId`—identifies the popup menu. Its value should be the integer that appears in the `column_idx` column of the `popups` table beside the default popup menu values. (You will need to perform an SQL query on the `popups` table to discover this value.)

  Example:

  <in_iColumnId xsi:type="xsd:int">12</in_iColumnId>

- `in_strLangId`—identifies the language. Standard language-country codes separated by an underscore are required.

  Example (for Canadian French):

  <in_strLangId xsi:type="soapenc:string" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/ ">fr_CA</in_strLangId>

Sample SOAP API Call to Populate Popup Values

The following SOAP API call will populate the `popups_lang` table with French equivalents for a popup menu with values “one”, “two”, and three”. This popup menu is identified with a column ID value of “12”, which appears in the `column_idx` column of the `popups` table.

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:impl="http://com/northplains/web/tsweb/soap/impl">
   <soapenv:Header/>
   <soapenv:Body>
      <impl:PopulatePopupValues soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <in_arNewPopupsValues xsi:type="impl:ArrayOf_soapenc_string">
            <in_arNewPopupsValues xsi:type="xsd:string">un</in_arNewPopupsValues>
            <in_arNewPopupsValues xsi:type="xsd:string">deux</in_arNewPopupsValues>
            <in_arNewPopupsValues xsi:type="xsd:string">trois</in_arNewPopupsValues>
         </in_arNewPopupsValues>
         <in_iColumnId xsi:type="xsd:int">12</in_iColumnId>
         <in_strLangId xsi:type="soapenc:string"
          xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">fr_CA</in_strLangId>
       </impl:PopulatePopupValues>
   </soapenv:Body>
</soapenv:Envelope>
```

Chapter 5: Download Methods

In this Section:

5.1 Overview

These methods provide a number of ways to download assets.

The download API calls in the Integration Broker can time out if the number and/or sizes of files being downloaded is large, or if there are many complex conversions carried out as a part of the download. This results from the fact that the current Download Integration Broker API call is a blocking call (to make things simple for the caller), but it cannot provide the downloadable URL to the client until all of the requested files have been copied from their source File Broker locations to the Telescope.web staging area (with conversion if necessary), and then optionally compressed into a ZIP archive for serving to the caller.

If this process takes a long time, the caller's SOAP client may time out the call before the Telescope server is finished its preparatory work. To solve this problem, two calls in the API permit the caller to use Integration Broker's download functionality in an asynchronous manner.

The typical way of using these new calls in place of the current, blocking download call would be to simply 'busy loop', waiting for the DownloadStatus call to indicate that it is ready. Ideally, while waiting, the caller should give the user some feedback that the download preparation is in progress. Also, callers should keep in mind that SOAP is not an inexpensive or lightweight protocol, so repeatedly calling DownloadStatus not only uses up network bandwidth unnecessarily, but will result in wasted overhead on client and server.

5.1.1 Example

To replace the following call:

```
HashMap map = tsRemote.Download(listOfAssetItems);
```

The following program logic could be used:

```
long lDownloadKey = tsRemote.DownloadStart(listOfAssetItems);
HashMap map = tsRemote.DownloadStatus(lDownloadKey);
    while (map.containsKey('PERCENT_DONE') == TRUE)
        {
        -- display waiting progress to user (or allow user to cancel)
        -- delay some reasonable time (> 10 seconds recommended)
        map = tsRemote.DownloadStatus(lDownloadKey);
        }
    error handling
    extract and use 'DOWNLOAD_URL' from map
```

5.2 Getfile

```
public NPSMap GetFile    (    Integer in_numRecordId,
                              Integer in_numRenditionId,
                              String in_strConversion,
                              boolean in_bZipYn,
                              boolean in_bIsMacBinary)
throws                        Throwable
```

This method calls DLManager to download a file from the system. It can be used to bypass the ZIP compression, and MacBinary attributes if a direct URL to a file is desired. For example, when displaying a preview of a file, this call could be used to convert to a 72dpi JPEG, and the resulting URL could be used directly in an <IMG> HTML tag.

Parameters:

- in_numRecordId – A record ID for the file to be downloaded.
- in_numRenditionId – A rendition ID for the file to be downloaded; if this value is null, the lowest visible rendition is used.
- in_strConversion – A conversion string if the file should be converted into a different format; can have a null value if no conversion is required.
- in_bZipYn – True if the downloaded file requires to be zipped, false, otherwise.
- in_bIsMacBinary – True if a file has to be downloaded in Mac binary format.

Returns:

An NPSMap object with the following keys:

- DOWNLOAD_URL – A URL to the downloaded files.
- TIMESTAMP – A String value of the number of seconds elapsed since Jan, 1st, 1970 12:00 AM UTC.
- ERRORS – If there were any errors returned by DLManager and gracefully handled by this method, the return value will contain one more ERRORS keys containing an array (ArrayList) of strings that describe the errors. In case of a fatal error returned by DLManager, the return value will contain null value for the key DOWNLOAD_URL.
- If there were no errors during the download process, the ERRORS key still might be present with a null value or an empty array.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

NOTE: No exception is thrown if the DLManager is unable to download a file(s). Such errors are returned as a part of the return value for the key ERRORS.

5.3 Download

```
public NPSMap Download    (    NPSMap[] in_arAssetsData,
                               boolean in_bIsMacBinary)
throws                         Throwable
```

This method downloads one or more assets. It also provides access to the file conversion framework to support on-the-fly conversion during the download. The URL returned can be used to pull the file(s) to the client's computer. The URL is only valid for a single HTTP download. This method always tells the DLManager to zip files as multiple files cannot be downloaded without zipping them first.

Parameters

- in_arAssetsData – An array of NPSMap objects with the following keys and values:
- RECORD_ID: (Required) A Number value representing the record ID of the asset that is downloaded.
- RENDITION: (Optional) A Number value representing the rendition ID of the file that is downloaded; if this value is null or missing, the lowest visible rendition will be used.
- CONV_STRING: (Optional) A String value containing the conversion string if the file is converted.
- COPYCOV_YN: (Optional) A String value which is a flag of one character length that indicates that COV documents should be downloaded as well; this value must either be Y or N or a null value (a null value is equal to N).
- INCLUDE_CONTAINERS: (Optional) An array of String objects indicating the container field names whose contained documents should be downloaded as well. The format of each of the container field name is [table_name].[column_name], where table_name and column_name are the values returned by the EnumerateFields SOAP call. table_name is optional and if it is missing, the column is considered as belonging to the editorial table. If one or more of the columns indicated here are not visible to the current user, or they are not of the Container Field type, they are ignored and the anomaly is recorded in the log file.

NOTE: Any other keys and values in the NPSMap object is ignored.

- in_bIsMacBinary – True if files must be downloaded in Mac binary format; this flag applies to all the files in the in_arAssetsData array.

Returns

An NPSMap object with the following keys:

- DOWNLOAD_URL – A URL to the downloaded files.
- TIMESTAMP – A String value of the number of seconds elapsed since Jan, 1st, 1970 12:00 AM UTC.
- ERRORS – If there were any errors returned by DLManager and gracefully handled by this method, the return value will contain one more ERRORS keys containing an array (ArrayList) of strings that describe the errors. In case of a fatal error returned by DLManager, the return value will contain null value for the key DOWNLOAD_URL.

- If there were no errors during the download process, the ERRORS key still might be present with a null value or an empty array.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

NOTE: No exception is thrown if DLManager is unable to download the file(s). Such errors are returned as a part of the return value in the ERRORS key.

5.4 DownloadStart

```
public NPSMap DownloadStart    (    NPSMap[] in_arAssetsData,
                                    boolean in_bIsMacBinary,
                                    boolean in_bZipSingle)
throws                              Throwable
```

This method calls DLManager to download one or more files asynchronously. This method always tells the DLManager to zip files as multiple files cannot be downloaded without zipping them first.

Parameters

- in_arAssetsData – An array of NPSMap objects with the following keys and values:
- RECORD_ID: (Required) A Number value representing the record ID of the asset to be downloaded.
- RENDITION: (Optional) A Number value representing the rendition ID of the file to be downloaded; if this value is null or missing, the lowest visible rendition is used.
- CONV_STRING: (Optional) A String value containing the conversion string if the file is converted.
- COPYCOV_YN: (Optional) A String value which is a flag of one character length that indicates that COV documents should be downloaded as well; this value must either be Y or N or a null value (a null value is equal to N).
- INCLUDE_CONTAINERS: (Optional) An array of String objects indicating the container field names whose contained documents should be downloaded as well. The format of each of the container field name is [table_name.column_name], where table_name and column_name are the values returned by the EnumerateFields SOAP call. table_name is optional and if it is missing, the column is considered as belonging to the editorial table. If one or more of the columns indicated here are not visible to the current user, or they are not of the Container Field type, they are ignored and the anomaly is recorded in the log file.

NOTE: Any other keys and values in the NPSMap object is ignored.

- in_bIsMacBinary – true if files must be downloaded in Mac binary format; this flag applies to all the files in the in_arAssetsData array.
- in_bZipSingle – true if files must be bundled in one single zip archive (the configuration which applies to Telescope.web's Download Now functionality, controlling whether single files, or single files of specific file types, are compressed, is also adhered to by this call).

Returns:

A HashMap object with the key DOWNLOAD_KEY that contains a key identifying the current download. If there were any errors returned by DLManager and gracefully handled by this method, the return value will contain one more ERRORS keys containing an array (ArrayList) of strings that describe the errors. In case of a fatal error returned by DLManager, the return value will contain null value for the key DOWNLOAD_URL.

If there were no errors during the download process, the ERRORS key still might be present with a null value or an empty array.

Throws

- java.lang.Throwable – In case of errors, including validation errors. For more information, seeSection 15.1, "Exception Handling," on page 132.

NOTE: No exception is thrown if DLManager is unable to download the file(s). Such errors are returned as a part of the return value in the ERRORS key.

5.5 DownloadStatus

```
public NPSMap DownloadStatus  (    String in_strDownloadKey)
throws                             Throwable
```

This method returns assets downloading status from DLManager. If the operation is complete, it returns the information necessary to retrieve the prepared file(s).

If the value of the passed key does not match one of the asynchronous download operations currently being executed by the Integration Broker, the method will raise a SOAP fault.

Parameters

- in_strDownloadKey – A string value of the key for the download operation, obtained as the result returned from a DownloadStart call.

Returns:

If the download operation is ongoing, this method returns a HashMap containing a single PERCENT_DONE key with an integer value between 0 and 100, indicating the approximate percentage complete for the overall operation.

When the file(s) are converted, staged, and ready to be served to the caller, the returned HashMap will instead contain the same keys as the return value for the original Download call.

The Integration Broker automatically cleans up and disposes all of its internal structures associated with the asynchronous download operation as soon as it successfully returns the final, completed HashMap to the caller. For this reason, any attempt by the caller to call DownloadStatus again for the same download key value will result in a TSWeb-0033 SOAP fault being raised. The prepared file(s) are not deleted at this time, as they still need to be retrieved by the caller, and is deleted by DLManager when they have been completely retrieved by the caller.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

NOTE: No exception is thrown if DLManager is unable to download the file(s). Such errors are returned as a part of the return value in the ERRORS key.

5.6 DownloadAged

```
public NPSMap DownloadAged    (    NPSMap[] in_arAssetsData,
                                   boolean in_bIsMacBinary
                                   boolean in_bZipSingle)
throws                             Throwable
```

This method calls DLManager to download a version of one or more files, prior to a given date. The "aged" call scans the ED_VERSIONS table for the given record_id and rend_id looking for a version that was checked in on or before the given date and time. If one is found, that file is downloaded. If not, the file referenced in DOC_RENDITIONS for the passed-in record_id and rend_id is downloaded.

Parameters:

- in_arAssetsData – An array of NPSMap objects with the following keys and values:
- RECORD_ID: (Required) A Number value representing the record ID of the asset to be downloaded.
- RENDITION: (Optional) A Number value representing the rendition ID of the file to be downloaded; if this value is null or missing, the lowest visible rendition is used.
- CONV_STRING: (Optional) A String value containing the conversion string if the file is converted.
- CURRENT_AS_OF: (Optional) A String value containing the number of seconds elapsed since Jan 1st 1970, 12:00:00 AM.
- COPYCOV_YN: (Optional) A String value which is a flag of one character length that indicates that COV documents should be downloaded as well; this value must either be Y or N or a null value (a null value is equal to N).
- INCLUDE_CONTAINERS: (Optional) An array of String objects indicating the container field names whose contained documents should be downloaded as well. The format of each of the container field names is [table_name].[column_name], where table_name and column_name are the values returned by the EnumerateFields SOAP call. table_name is optional and if it is missing, the column is considered as belonging to the editorial table. If one or more of the columns indicated here are not visible to the current user, or they are not of the Container Field type, they are ignored and the anomaly is recorded in the log file.

NOTE: Any other keys and values in the NPSMap object is ignored.

- in_bIsMacBinary – True if files must be downloaded in Mac binary format; this flag applies to all the files in the in_arAssetsData array.
- in_bZipSingle – True if files must be bundled in one single zip archive (the configuration which applies to Telescope.web's Download Now functionality, controlling whether single files, or single files of specific file types, are compressed, is also adhered to by this call).

Returns:

An NPSMap object with the following keys:

- DOWNLOAD_URL – A URL to the downloaded files.
- TIMESTAMP – A String value of the number of seconds elapsed since Jan, 1st, 1970 12:00 AM UTC.
- ERRORS – If there were any errors returned by DLManager and gracefully handled by this method, the return value will contain one more ERRORS keys containing an array (ArrayList) of strings that describe the errors. In case of a fatal error returned by DLManager, the return value will contain null value for the key DOWNLOAD_URL.

If there were no errors during the download process the ERRORS key still might be present with a null value or an empty array.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

NOTE: No exception is thrown if DLManager is unable to download the file(s). Such errors are returned as a part of the return value in the ERRORS key.

5.7 DownloadAgedStart

```
public NPSMap DownloadAgedStart    (    NPSMap[] in_arAssetsData,
                                        boolean in_bIsMacBinary
                                        boolean in_bZipSingle)
throws                                  Throwable
```

This method calls DLManager to download one or more files asynchronously, attempting to retrieve a version that has been checked in prior to a given date. The "aged" call scans the ED_VERSIONS table for the given record_id and rend_id looking for a version that was checked in on or before the given date and time. If one is found, that file will be downloaded. If not, the file referenced in DOC_RENDITIONS for the passed-in record_id and rend_id is downloaded. This method always tells the DLManager to zip files as multiple files cannot be downloaded without zipping them first.

Parameters

- in_arAssetsData: An array of NPSMap objects with the following keys and values:
- RECORD_ID: (Required) A Number value representing the record ID of the asset to be downloaded.
- RENDITION: (Optional) A Number value representing the rendition ID of the file to be downloaded; if this value is null or missing, the lowest visible rendition is used.
- CONV_STRING: (Optional) A String value containing the conversion string if the file is converted.
- CURRENT_AS_OF: (Optional) A String value containing the number of seconds elapsed since Jan 1st 1970, 12:00:00 AM.
- COPYCOV_YN: (Optional) A String value which is a flag of one character length that indicates that COV documents should be downloaded as well; this value must either be Y or N or a null value (a null value is equal to N).
- INCLUDE_CONTAINERS: (Optional) An array of String objects indicating the container field names whose contained documents should be downloaded as well. The format of each of the container field names is [table_name.column_name], where table_name and column_name are the values returned by the EnumerateFields SOAP call. table_name is optional and if it is missing, the column is considered as belonging to the editorial table. If one or more of the columns indicated here are not visible to the current user, or they are not of the Container Field type, they are ignored and the anomaly is recorded in the log file.

NOTE: Any other keys and values in the NPSMap object is ignored.

- in_bIsMacBinary: True if files must be downloaded in Mac binary format; this flag applies to all the files in the in_arAssetsData array
- in_bZipSingle: True if files must be bundled in one single zip archive (the configuration which applies to Telescope.web's Download Now functionality, controlling whether single files, or single files of specific file types, are compressed, is also adhered to by this call).

Returns

An NPSMap object with the following keys:

- DOWNLOAD_URL: A URL to the downloaded files
- TIMESTAMP: A String value of the number of seconds elapsed since Jan, 1st, 1970 12:00 AM UTC
- ERRORS: If there were any errors returned by DLManager and gracefully handled by this method, the return value will contain one more ERRORS keys containing an array (ArrayList) of strings that describe the errors. In case of a fatal error returned by DLManager, the return value will contain null value for the key DOWNLOAD_URL.
- If there were no errors during the download process the ERRORS key still might be present with a null value or an empty array.

Throws

- java.lang.Throwable: In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

NOTE: No exception is thrown if DLManager is unable to download the file(s). Such errors are returned as a part of the return value in the ERRORS key.

5.8 DownloadStatus

```
public NPSMap DownloadAgedStatus  (    String in_strDownloadKey)
throws                                 Throwable
```

This method returns assets downloading status from DLManager. If the operation is complete, it returns the information necessary to retrieve the prepared file(s).

If the value of the passed key does not match one of the asynchronous download operations currently being executed by the Integration Broker, the method will raise a SOAP fault.

Parameters:

- in_strDownloadKey: A string value of the key for the download operation, obtained as the result returned from a DownloadStart call.

Returns:

If the download operation is ongoing, this method returns a HashMap containing a single PERCENT_DONE key with an integer value between 0 and 100, indicating the approximate percentage complete for the overall operation.

When the file(s) are converted, staged, and ready to be served to the caller, the returned HashMap will instead contain the same keys as the return value for the original Download call.

The Integration Broker automatically cleans up and disposes all of its internal structures associated with the asynchronous download operation as soon as it successfully returns the final, completed HashMap to the caller. For this reason, any attempt by the caller to call DownloadStatus again for the same download key value will result in a TSWeb-0033 SOAP fault being raised. The prepared file(s) are not deleted at this time, as they still need to be retrieved by the caller, and are deleted by DLManager when they have been completely retrieved by the caller.

Throws:

- java.lang.Throwable: In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

NOTE: No exception is thrown if DLManager is unable to download the file(s). Such errors are returned as a part of the return value in the ERRORS key.

5.9 DownloadPageURL

```
public NPSMap DownloadPageURL    (    NPSMap[] in_arAssetsData,
                                      boolean in_bIsMacBinary,
                                      string in_strSiteName,
                                      boolean in_bZipSingle)
throws                                Throwable
```

This method returns the URL of the DLManager page for downloading selected assets.

Parameters:

- in_arAssetsData: An array of NPSMap objects with the following keys and values for each asset to be downloaded:
- RECORD_ID: (Required) A Number value representing the record ID of the asset to be downloaded
- RENDITION: (Optional) A Number value representing the rendition ID of the file to be downloaded; if this value is null or missing, the lowest visible rendition is used
- CONV_STRING: (Optional) A String value containing the conversion string if the file is converted
- COPYCOV_YN: (Optional) A String value which is a flag of one character length that indicates that COV documents should be downloaded as well; this value must either be Y or N or a null value (a null value is equal to N)
- INCLUDE_CONTAINERS: (Optional) An array of String objects indicating the container field names whose contained documents should be downloaded as well. The format of each of the container field name is [table_name.column_name], where table_name and column_name are the values returned by the EnumerateFields SOAP call. table_name is optional and if it is missing, the column is considered as belonging to the editorial table. If one or more of the columns indicated here are not visible to the current user, or they are not of the Container Field type, they are ignored and the anomaly is recorded in the log file.

NOTE: Any other keys and values in the NPSMap object is ignored.

- in_bIsMacBinary: True if files must be downloaded in Mac binary format; this flag applies to all the files in the in_arAssetsData array
- in_strSiteName: A string value representing the site name.
- in_bZipSingle: True if files must be bundled in one single zip archive (the configuration which applies to Telescope.web's Download Now functionality, controlling whether single files, or single files of specific file types, are compressed, is also adhered to by this call).

Chapter 6: API Methods: Catalogs

Methods in the Catalogs functional group focus on providing external access to the nested catalog structure and catalog information in the Telescope environment. Given their nested structure, to uniquely identify a catalog in the Telescope system, one must specify the path to the catalog. A catalog path is a string that defines the catalog's within the overall nesting structure, and is composed of the name(s) of all of the catalogs from the root of the catalog tree down to the desired catalog, separated by a pipe character (|).

In this Section:

6.1 AddToCatalog

```
public String[] AddToCatalog    (    String in_strPath,
                                     String in_strPassword,
                                     int[] in_arRecordIds)
throws                               Throwable
```

This method adds one or more assets to a catalog. The logged-in user must have edit permissions on the specified catalog and be able to see the assets. If any of the asset record_ids passed are invalid or invisible to the logged-in user a SOAP fault is raised and none or the records are added to the catalog.

This method will execute any Add To Catalog functional rules for each asset in the list. For more information about how the Integration Broker handles functional rules, see Working with Functional Rules.

Parameters:

- in_strPath – The catalog path for the parent catalog.
- in_strPassword – The current password for validation.
- in_arRecordIds – An array of integers representing the record_id values for the assets to be added to the catalog.

Returns:

An array of strings containing any functional rule error messages that were generated as the result of this execution.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

6.2 CreateCatalog

```
public void CreateCatalog    (    String in_strNestingPath,
                                  String in_strName,
                                  String in_strPassword,
                                  NPSCatalogACLEntry[] in_aACL,
                                  boolean in_bAllowNesting)
throws                            Throwable
```

This method creates a new, empty catalog.

Parameters:

- in_strNestingPath – The catalog path of the catalog under which the new catalog shouldbe created.
- in_strName – The name of the catalog which is created.
- in_strPassword – The new view password for the catalog.
- in_aACL – An array of NPCatalogACLEntry structures, which define a new set of visibility and editing permissions for the catalog.
- in_bAllowNesting – The value that allows nesting (YES or NO) in the added catalog.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

6.3 EnumerateCatalogs

```
public NPSCatalogInfo[] EnumerateCatalogs    (    String in_strPath)
throws                                            Throwable
```

This method returns the catalogs nested within the given catalog structure.

Parameters:

in_strPath – The catalog path for the parent catalog; if this does not refer to a valid catalog, this method will raise a SOAP fault. To obtain the 'root' level catalogs, pass an empty string for this parameter.

Returns:

Returns a list of structures containing the information for each catalog contained in the first level under the parent structure and visible to the logged-in user (or an empty list). Each structure contains information about the catalog including the catalog name, create and modified date, owner, password protection, number of assets contained and number of catalogs nested under the first level below.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

6.4 DeleteCatalog

```
public boolean DeleteCatalog    (    String in_strPath)
throws                               Throwable
```

This method deletes a catalog. The logged-in user must be either the owner of the specified catalog, or an administrator who has group visibility to the catalog owner.

Parameters:

- in_strPath – The catalog path for the parent catalog.

Returns:

Returns true if the catalog and all descendants were able to be deleted or false otherwise.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

6.5 GetCatalogAssets

```
public int[] GetCatalogAssets    (    String in_strPath,
                                      String in_strPassword)
throws                               Throwable
```

This method gets a list of the assets in a given catalog.

Parameters:

- in_strPath – The catalog path for the parent catalog.
- in_strPassword – The current password for validation.

Returns:

An array of integers representing the assets in the catalog. Only those assets that are visible to the logged-in user are returned.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

6.6 GetCatalogProperties

```
public NPSCatalogProperties GetCatalogProperties    (    String in_strPath,
throws                                                   Throwable
```

This method returns the properties information for a given catalog available to the logged-in user.

Parameters:

- in_strPath – The path to the parent catalog.

Returns:

Returns the properties information of the passed-in catalog, provided the user is the owner of the catalog or an administrator who has visibility over the catalog owner, otherwise an SOAP fault is raised. The structure contains information about the parent catalog path, password protection, and the access control list (ACL) defining the visibility and editing permissions for the catalog. Each entry in the ACL contains information about the target entity (either a user name, a user group name, or the special value" !ALL!", which indicates that every user whose user group visibility allows them to see the owner of the catalog, can see the catalog) and a flag indicating either read-only or modify privileges.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

6.7 RemoveFromCatalog

```
public String[] RemoveFromCatalog (  String in_strPath,
                                     String in_strPassword,
                                     int[] in_arRecordIds)
throws                               Throwable
```

This method removes one or more assets from a catalog. The logged-in user must have 'edit' permission for the specified catalog.

This method will execute any Remove From Catalog functional rules for each asset in the list. For more information see Working with Functional Rules.

Parameters:

- in_strPath – The catalog path for the parent catalog.
- in_strPassword – The current password for validation.
- in_arRecordIds – An array of the record_id values for the assets to be added to the catalog.

Returns:

Returns an array of strings containing the record_id values of the assets that were removed from the catalog and any functional rule error messages that were generated as the result of this execution.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

6.8 SetCatalogProperties

```
public void SetCatalogProperties    (    String in_strPath,
                                         String in_strNewNesting,
                                         String in_strNewName,
                                         String in_strPassword,
                                         NPSCatalogACLEntry[] in_aACL
throws                                   Throwable
```

This method sets the catalog's properties information. The logged-in user must be the owner of the catalog, or be an administrator who has visibility over the catalog owner or a SOAP fault is raised.

Parameters:

- in_strPath – The catalog path for the parent catalog.
- in_strNewNesting – A catalog path that defines an existing catalog in the system, under which the catalog should be moved.
- in_strNewName – The new name for the catalog. The new name for the catalog must not already exist at the selected nesting level in the system.
- in_strPassword – The new view password for the catalog.
- in_aACL – An array of NPCatalogACLEntry structures that define a new set of visibility and editing permissions for the catalog.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

Chapter 7: API Methods: Ingestion

This section provides information about the ingesting files:

7.1 Overview

Ingestion and check in methods "WithData" allow a calling application to pass the actual binary data for an ingested file, as a part of the calls themselves, rather than indirectly as URLs that the Integration Broker must then 'pull' the file from (which can be problematic if the client is not local to the Integration Broker, and doesn't have a handy HTTP or FTP server to put the file on, so Integration Broker can retrieve it).

The following methods are available to transfer binaries over SOAP:

- Input parameter to the Web service method - highly inefficient due to the huge overhead caused by the usually BASE64 encoding of data in the SOAP payload.
- SOAP with Attachments - provides a standardized way to send a MIME-encoded message that includes the SOAP envelope in one MIME part, and an arbitrary number of attachments in the other MIME parts. With this technique, the file can be sent as a single MIME part in a two-part MIME message. This method has its own downsides, the primary one being that while MIME boundaries make it easy for the sender of the attachment to add an arbitrary number of attachments, it takes considerably more work to pick out the precise attachments from the receiver's standpoint, since each attachment must be scanned in its entirety, in order to find the MIME boundaries.
- DIME and WS-Attachments – DIME (Direct Internet Message Encapsulation) is a standard for sending a message in explicitly-sized chunks. WS-Attachments is a standard that defines how to use DIME to send attachments in a Web service call. For example, one chunk could contain the SOAP envelope while another chunk could contain the attachment. DIME and WS-Attachments are the preferred technique for sending attachments. Unlike SOAP with Attachments, which uses MIME boundaries to separate the attachments from one another, DIME uses explicitly-sized chunks. Each chunk has a record number and a size associated with it, therefore the receiver of a DIME-formatted message can quickly move from one attachment to another. With SOAP with Attachments, moving from one attachment to another requires examining the first attachment in its entirety, looking for the MIME boundary whereas with DIME, the receiver only needs to check the size of the current chunk and then skip ahead however many bytes specified to get to the beginning of the next chunk.

  Only a single data stream (that is, only the data fork of a multi-fork file) can be sent to Telescope using this method. If the file being ingested or checked-in is a multi-fork file (for example, from a Macintosh), the caller must first convert the file into a single-stream format using the standard MacBinary encoding, before passing the stream to Integration Broker, and indicate in the call that the stream refers to a MacBinary file, by passing TRUE for the in_bIsMac parameter. Integration Broker will automatically decode the MacBinary file during the ingestion.

7.2 Ingest

```
public int[] Ingest      (    String[] in_arUrls,
                              String in_strMimix,
                              int in_iRenditionId,
                              int in_iIsMac)
throws                        Throwable
```

This call adds new assets to the system. Assets must be accessible via the internet, using the passed-in URL. To ingest assets not accessible via the internet (for example assets on a user's desktop) use the IngestWithData method. Integration Broker pulls the assets into the Telescope system and puts them on a File Broker in the same way as Telescope.web. This method doesn't support binary data in the MIMiX file as there is no support for the binary data type in the extra_columns table. This method can be used only inside response-request loop.

Parameters:

- in_arUrls – An array of URLs to the files. If this value is null or an empty array, a single New Document is created.
- in_strMimix – A string that represents a MIMiX XML file with metadata.
- in_iRenditionId – The rendition ID to assign to the ingested files. If the value is less than or equal to 0 or null, the assets is ingested with the lowest visible rendition for the current user.
- in_iIsMac – Pass 1 if the files are in Mac binary format, otherwise, 0. This flag applies to all the files in the in_arUrls array.

Returns:

Returns an array of record IDs for successfully ingested assets.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

7.3 GetIngestStatus

public String[ ] GetIngestStatus	(	int in_strProcessId)
throws		Throwable

Queries ingest status associated with the unique request/process ID obtained in a call to {IngestWithStatus (String[], String, int, boolean, String)}

This method has to be called in a loop with some delay (to avoid high CPU usage) until the status returns "DONE".

The return status "DONE" means that the ingest process is finished, either succeeded, failed or with some errors.

Parameters:

- in_strProcessId — unique request/process id returned by {IngestWithStatus (String[], String, int, boolean, String)}

Returns:

Returns the current status of the ingest process.

The return value might have the following keys. Each key-value pair is represented by {NPSMapItem} in the array in the return value {NPSMap}

- "DONE" — if present, contains a boolean value. "True" value indicates the ingest process is finished. If the value is "false" or missing, a call to this method should continue until the value is "true".
- "ERRORS" — if present, contains an array of strings if there were any error during the ingest process.
- "RECORD_IDS" — if present, contains an array of integers, primary keys of the ingested assets. If this key-value pair is missing or doesn't contain any value when "DONE" value is "true", means that no assets were successfully ingested.
- "MESSAGES" — if present, contains and array of strings. These messages can be displayed for the user indicating progress of the ingest in a readable form.

These messages can be configured or adjusted using SOAPMessages.plist file following under 6xxx range.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information see Section 15.1, "Exception Handling," on page 132.

7.4 GetTemplateNames

```
public String[ ] GetTemplateNames    (      )
throws                                      Throwable
```

This gets a list of the templates available in the system. This is used to support the IngestWithTemplate call.

Returns:

Returns an array of strings containing the names of the templates visible to the user.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

7.5 IngestWithTemplate

```
public int[] IngestWithTemplate    (    String[] in_arUrls,
                                        String in_strTemplateName,
                                        int in_iRenditionId,
                                        int in_iIsMac)
throws                                  Throwable
```

This method adds new assets to the system with the metadata from the passed in template. Assets must be accessible via the internet, using the passed-in URL. Integration Broker pulls the assets into the Telescope system and puts them on a File Broker in the same way that Telescope.web does.

This method doesn't support binary data in the MIMiX file as there is no support for the binary data type in the extra_columns table. This method can be used only inside response-request loop.

Parameters:

- in_arUrls – The URLs to the files. If the value in the array is null, an empty asset is created.
- in_strTemplateName – A string representing a template name in the JOBS table. The template stored under this name must be in MIMiX format. Use the GetTemplateNames method to get a list of available templates in the JOBS table.
- in_iRenditionId – The rendition ID to assign to the ingested files. If the value is less than or equal to 0 or null, the assets is assigned the lowest visible rendition for the current user.
- in_iIsMac – Pass 1 if the files are in Mac binary format, otherwise, 0. This flag applies to all the files in the in_arUrls array.

Returns:

Returns an array of record IDs for successfully ingested assets.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information see Section 15.1, "Exception Handling," on page 132.

7.6 IngestWithData

```
public int[] IngestWithData    (    String in_strMimix,
                                    int in_iRenditionId,
                                    boolean in_bIsMac,
                                    String in_strFileName)
throws                              Throwable
```

This method passes the binary data for an file when it is ingested. The binary data is sent as a DIME attachment to the SOAP call. For more information about passing binary data, see Section 7.1, "API Methods: Ingestion," on page 61.

NOTE: The attachment type of the file attached to the request should be set to "application/octet-stream" in order for the TSWeb SOAP provider to understand it correctly.

Parameters:

- in_strMimix – A string that represents a MIMiX XML file with metadata for the asset.
- in_iRenditionId – The rendition ID to assign to the ingested file. If the value is less than or equal to 0 or null, the asset is assigned the lowest visible rendition for the current user.
- in_bIsMac – True if the files are in Mac binary format, otherwise, false.
- in_strFileName – The input file name.

Returns:

Returns the newly created record_id value.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

7.7 IngestWithTemplateAndData

```
public int IngestWithTemplateAndData    (    String in_strTemplate,
                                             int in_iRenditionId,
                                             boolean in_bIsMac,
                                             String in_strFileName)
throws                                       Throwable
```

This method passes the actual binary data for an ingested file with its template. The binary data is sent as a DIME attachment to the SOAP call. For more information about passing binary data, see Section 7.1, "API Methods: Ingestion," on page 61.

.

Parameters:

- in_strTemplate – A string representing a template name in the JOBS table. The template stored under this name must be in the MIMiX format. Use the GetTemplateNames method to get a list of available templates in the JOBS table.
- in_iRenditionId – The rendition ID to assign to the ingested file. If the value is less than or equal to 0 or null, the asset is assigned the lowest visible rendition for the current user.
- in_bIsMac – True if the file is in Mac binary format, otherwise, false.
- in_strFileName – The input file name.

Returns:

Returns the newly created record_id values.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

7.8 IngestWithStatus

```
public int[] IngestWithStatus    (    String[] in_arUrls,
                                      String in_strMimix,
                                      int in_iRenditionId,
                                      int in_bIsMac
                                      String in_strTemplateName)
throws                                Throwable
```

This call ingests multiple assets into the Telescope database asynchronously.

The binary data is sent as a DIME attachment to the SOAP call. For more information about passing binary data, see Section 7.1, "API Methods: Ingestion," on page 61.

Parameters:

- in_arUrls — An array of URLs to the files. If this value is null or an empty array, a single New Document is created.
- in_strMimix — A string that represents a MIMiX XML file with metadata fields.
- in_iRenditionId — The rendition ID to assign to the ingested files. If the value is less than or equal to 0 or null, the assets is ingested with the lowest visible rendition for the current user.
- in_bIsMac — 1 if a file is in Mac binary format; otherwise, 0.
- in_strTemplateName — a string that represents a template name in the jobs table. The template stored under this name MUST be in the MIMiX format. If this value is present, the MIMiX string is ignored and a template is used instead.

Returns:

Returns a unique ID associated with the current request. The unique ID is returned without waiting while the ingest process is finished.

This unique ID should be used in calls to {@link #GetIngestStatus(String)} to obtain the current status of the ingest.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

7.9 IngestWithDataAndStatus

public int[] IngestWithDataAndStatus	(	String in_strMimix, int in_iRenditionId, int in_bIsMac String in_strTemplateName)
throws		Throwable

This call ingests multiple assets into the Telescope database asynchronously.

The binary data is sent as a DIME attachment to the SOAP call. For more information about passing binary data, see Section 7.1, "API Methods: Ingestion," on page 61.

Parameters:

- in_strMimix — A string that represents a MIMiX XML file with metadata fields.
- in_iRenditionId — The rendition ID to assign to the ingested files. If the value is less than or equal to 0 or null, the assets is ingested with the lowest visible rendition for the current user.
- in_bIsMac — 1 if a file is in Mac binary format; otherwise, 0.
- in_strTemplateName — a string that represents a template name in the jobs table. The template stored under this name MUST be in the MIMiX format. If this value is present, the MIMiX string is ignored and a template is used instead.

Returns:

Returns a unique ID associated with the current request. The unique ID is returned without waiting while the ingest process is finished.

This unique ID should be used in calls to {@link #getIngestWithDataStatus(String)} to obtain the current status of the ingest.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. or more information, see Section 15.1, "Exception Handling," on page 132.

Example:

```
public int[] IngestWithStatus ( String[] in_arUrls,
String in_strMimix,
int in_iRenditionId,
int in_bIsMac
String in_strTemplateName)
throws Throwable
```

7.10 GetIngestWithDataStatus

public String[ ] GetIngestWithDataStatus	(	int in_strProcessId)
throws		Throwable

Queries ingest status associated with the unique request/process ID obtained in a call to {IngestWithStatusAndData (String[], String, int, boolean, String)}

This method has to be called in a loop with some delay (to avoid high CPU usage) until the status returns "DONE".

The return status "DONE" means that the ingest process is finished, either succeeded, failed or with some errors.

Parameters:

- in_strProcessId — unique request/process id returned by {IngestWithStatusAndData (String[], String, int, boolean, String)}

Returns:

Returns the current status of the ingest process.

The return value might have the following keys. Each key-value pair is represented by {NPSMapItem} in the array in the return value {NPSMap}

- "DONE" — if present, contains a boolean value. "True" value indicates the ingest process is finished. If the value is "false" or missing, a call to this method should continue until the value is "true".
- "ERRORS" — if present, contains an array of strings if there were any error during the ingest process.
- "RECORD_IDS" — if present, contains an array of integers, primary keys of the ingested assets. If this key-value pair is missing or doesn't contain any value when "DONE" value is "true", means that no assets were successfully ingested.
- "MESSAGES" — if present, contains and array of strings. These messages can be displayed for the user indicating progress of the ingest in a readable form.

These messages can be configured or adjusted using SOAPMessages.plist file following under 6xxx range.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information see Section 15.1, "Exception Handling," on page 132.

Chapter 8: Login/Out and Session Maintenance

This section provides information about logging in and out and session maintenance:

8.1 Overview

The first call must always be one of the SOAP Login calls; these calls return two URLs: a new SOAP endpoint URL, and a URL for accessing the UI Service.

The session ID is automatically embedded in the URLs by the Integration Broker. This session is maintained on the server and expires after a period of time if no further calls are made. Use the greeting call periodically to keep the session alive if necessary.

The Logout call terminates the session.

8.2 Login

```
public NPSMap Login     (   String inStrConnName,
                            String inStrUsername,
                            String inStrPassword)
throws                      Throwable
```

This must be the first call made by the client to the Integration Broker. The only call supported by the "main" SOAP endpoint is the Login call. All subsequent calls to the SOAP API must be made to the SOAP_SERVICE endpoint returned from the Login call.

Parameters:

- inStrConnName – The name of the connection to log in to.
- inStrUsername – The user name to connect with.
- inStrPassword – The password to connect with.

Returns:

Returns an NPSMap containing the SOAP_SERVICE and UI_SERVICE URLs to use for respective services.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

8.3 LoginWithProvider

```
public NPSMap LoginWithProvider    (    String inStrConnName,
                                        String inStrProviderName,
                                        NPSMap hValue)
throws                                  Throwable
```

This method must be called before any other method. It validates the user against the specified data source, using an authentication provider class. This is the login with provider method that must be called by the TSWeb SOAP SSO service in order to use all other SOAP services.

In case of errors, including validation errors If the login is not successful, the method returns an error message as a SOAP fault indicating the reason for the failure.

Parameters:

- in_strConnName – The connection name. This must match the label for a connection in the sites.plist file. In the event of duplicate connection names across multiple sites, the first one found is used.
- inStrProviderName – A String that identifies the name of the plug-in class used to perform authentication.
- hValues – A transparent HashMap object that is used by the authentication provider class during authentication.

Returns:

Returns an NPSMap containing the SOAP_SERVICE and UI_SERVICE URLs to use for respective services.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

8.3.1 Performing Authentication

The com.northplains.authentication.AuthenticationProviderInterface has the following method which must be implemented by the concrete plug-in class that represents an authentication provider:

```
public String authenticater    (    NPSMap hValue)
throws                              Throwable
```

Parameters:

- hValues – A transparent HashMap object.

Returns:

Returns a String containing a Telescope existing user name, which has been authenticated. If the user does not exist, an empty string is returned. If the information provided into the HashMap object is null or unusable, a null String object is returned.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

8.4 EnumerateConnections

```
public NPSMapArray EnumerateConnectionsr    (    String in_strForSite)
throws                                           Throwable
```

This method returns database connection(s) for a given site as defined in the sites.plist file. This call can be made before the Login or LoginWithProvider calls are made to the Integration Broker.

Parameters:

- in_strForSite – The name of the site (from the sites.plist file).

Returns:

An array of the site connections in an NPSMapArray object.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Section 15.1, "Exception Handling," on page 132.

8.5 Greeting

```
public String greeting    (    )
throws                         Throwable
```

Use this method to verify that the SOAP service is up and running. It can also be used by clients who do not make regular calls to the Integration Broker to prevent their sessions from expiring. How often the call should be made is governed by the session timeout value on the Integration Broker (usually 30 minutes).

Returns:

Returns a welcome message that reads "Hello <user>".

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

8.6 IsValidSession

public boolean IsValidSession (String in_strSessionId)

This method checks whether a given session ID is valid.

Parameters:

- in_strSessionId – A String object representing the sessionId (WOSession) to be looked up.

Returns:

Returns a boolean value: true if the sessionId is valid, and false otherwise.

8.7 Logout

```
public void Logout ( )
throws Throwable
```

This method terminates the active session. The SOAP endpoint URL used for this call, and the equivalent UI Service URL, become invalid as soon as this call returns. After this call the service consumer must log in again to use the services.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

Chapter 9: API Methods: Metadata Methods

The metadata access methods allow a calling application to get and change metadata for one or more assets.

The bulk metadata access methods permit access to multiple fields of an asset in a single call, and the ability to perform Change Multiple operations via the API.

In this Section:

9.1 EnumerateFields

public NPSMapArray EnumerateFieldsr (

throws Throwable

This method provides information about the metadata model that is accessible to the logged-in user. The returned hash map contains complete information about each visible field, including data type, database column and table names, popup menu items, etc. This method gets all fields visible for the current user as defined by the extra_columns and view_fields tables.

Returns:

An NPSMapArray where each NPSMap within the array contains keys and values as a result of two joined tables: EXTRA_COLUMNS and VIEW_FIELDS. The keys and values from the VIEW_FIELDS table are view_order and edit_yn. All the keys are in upper case.

Each NPSMap also contains a DATATYPE_VIEWNAME key whose value is a display name of the data type defined, and if this field has popups values defined in the popups table, a POPUPS key whose value is an array of strings.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

Example:

```
key: TABLE_NAME value: editorial
key: VIEW_ORDER value: 20
key: DATATYPE_VIEWNAME value:
key: VIEWER_NAME value: Date Produced
key: DISTRIBUTE_YN value: Y
key: EDIT_YN value: Y
key: PRIV_LVL value: Null
key: COLCASCADE_YN value: Null
key: REQUIRED_YN value: Null
key: COLUMN_NAME value: date_produced
key: SEARCH_OPERATORS value: (NPSMap Array) {OPERATOR_ID=1, OPERATOR=Is,
   OPERATOR_SHORT==},{OPERATOR_ID=2, OPERATOR=Is Not,
   OPERATOR_SHORT=!=},{OPERATOR_ID=3, OPERATOR=Less Than,
   OPERATOR_SHORT=<},{OPERATOR_ID=4, OPERATOR=Greater Than,
   OPERATOR_SHORT=>},{OPERATOR_ID=5, OPERATOR=Less Than Or Equal,
   OPERATOR_SHORT=<=},{OPERATOR_ID=6, OPERATOR=Greater Than Or Equal,
   OPERATOR_SHORT=>=}
key: MAX_LEN value: 1
key: ID value: 11
key: LOOKUP_YN value: Null
key: CUSTOM_YN value: Null
key: VALIDATE_YN value: Null
key: DATA_TYPE value: 5
key: COL_PROPERTIES value: Null
```

9.2 GetData

```
public Object GetData    (    Integer in_numRecordId,
                              Integer in_numRenditionId,
                              String in_strFieldName)
throws                        Throwable
```

This method gets the metadata stored in a Telescope database field for a given asset. Any metadata (including fields outside of editorial) is accessible. Rendition ID is optional and only applies if asking for file related metadata (such as file_name or file_type).

Parameters:

- in_numRecordId – The record ID of the asset.
- in_numRenditionId – The rendition ID of the file. This value is used only for DOC_RENDITIONS table. If this value is null or less than or equal to 0, the lowest visible rendition ID is used instead.
- in_strFieldName – A fully qualified field name, in the format [table name].[column name]; if there is no table name qualifier, EDITORIAL table is assumed.

Returns:

The object returned depends on the type of data requested:

- If the field is defined in the database as binary, the return value is an array of bytes.
- If the column in the database is defined as timestamp/date, the return value is converted into a string with the format yyyy/MM/dd HH:mm:ss.
- If the record for the given record ID or/and rendition ID is not found or there is no value for the given field, returns null

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

9.3 GetDataMultiple

```
public String GetDataMultiple    (    int[] in_arRecordIds,
                                      String[] in_arFields)
throws                                Throwable
```

This method retrieves the values for one or more asset metadata fields from one or more assets in a single call. The actions retrieved are activated by calling getMBMessageAction.

The passed-in record_id values must refer to valid Telescope asset records the logged-in user has permission to see, otherwise, this method will raise a SOAP fault. Similarly, the passed in field names must be valid Telescope metadata fields or rendition fields.

Parameters:

- in_arRecordIds – An array of the record_id values for which metadata is to be retrieved.
- in_arFields – An array of the Telescope field names to retrieve for each asset.

Returns:

Returns a MIMiX-formatted string containing one or more ASSET elements (one for each requested asset). Each ASSET element will contain a FIELD element for each of the requested fields. The ASSET tags in the returned MIMiX data will always include an ID attribute, containing the asset's record_id identifier, so the multiple requested records can be distinguished from each other. If one of the requested fields passed in to the call via in_arFields was the meta:viewex metafield, and a requested asset has a COV or Video Manager extended view, then the returned ASSET element for that asset will also include the COV or VIDEOMGR elements, containing the asset's extended view data.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.
- "The supplied list of record_id values is too large"—The upper limit for this call is 1000 by default, as defined by the 'MAX_MULTI_RECORDS' parameter in the TSWeb config.plist file. To change this limit, go to the `C:\Telescope\Applications\tsweb.woa\Contents\Resources\Config.plist` file (default location shown) and update the MAX_MULTI_RECORDS key. Since this is an optional key, you may need to add it to the file.

9.4 SetData

```
public void SetData          (   int  in_numRecordId,
                                 int  in_numRenditionId,
                                 String in_strFieldName,
                                 Object in_objValue)
throws                           Throwable
```

This method permits modification of metadata in the Telescope database by the caller. The Rendition ID is only used if file-based metadata is being set. The "object" value is interpreted differently, depending on the data type of the field being set.

This method updates the data for the given field. This method supports all the fields defined in the extra_columns table. In addition, this method also supports all the fields in the doc_renditions, thumbnails and viewex tables.

Parameters:

- in_numRecordId – A record ID of the asset to be updated
- in_numRenditionId – A rendition ID of the file. This value is valid only for the doc_renditions table. If the table name is doc_renditions and this value is less than or equal to 0 or null, the lowest visible rendition for the current user will be used instead.
- in_strFieldName – A fully qualified field name in the format [table name].[column name]. If the table qualifier is missing, the editorial table is assumed.
- in_objValue – A new value for which the given field should be updated. If this value is null and the field is a type of normalized repeating, the normalized repeating record(s) is deleted from the database for the given record ID. If the field is a type of binary, in_objValue must be an array of bytes. If the field is a type of timestamp/date, in_objValue must be a type of string in the format yyyy/MM/dd HH:mm:ss

Throws:

- Throws errors as standard SOAP exceptions
- Throwable - in case of error, or if there's a failure to validate user privileges.

9.5 SetDataMultiple

public NPSMap SetDataMultiple	)	String in_strMimix)
throws		Throwable

This method permits modification of multiple assets elements in the Telescope database.

This method accepts a MIMiX-formatted string containing data to be used for changes. This MIMiX data may contain multiple ASSET elements, to set the data for multiple assets at once.

The 'Change Metadata' functional rules for the currently logged-in user are executed before the database update is done. As with other calls in the Integration Broker API that execute functional rules, if the user's rules return a challenge form, this will be treated as a failure, with the error message: 'Challenge forms are not supported in functional rules using the SOAP API'. Transactions are committed after each asset update.

Parameters:

- in_strMimix – an XML string in MIMiX format

Returns:

- Returns a list of updated records and the eventual Functional Rules errors being raised.
- The return value contains two keys, ERRORS and RECORD_IDS. ERRORS key has an array of strings with error messages. RECORD_IDS has an array of integers for the assets that were updated.

Throws:

- Throwable - in case of error, or if there's a failure to validate user privileges.

This method will raise a SOAP fault in the following situations:

- If the string is empty, or does not contain valid MIMiX data.
- If the passed-in MIMiX data contains ASSET elements without the ID attribute, or if the ID attributes do not refer to valid Telescope asset records, or if they refer to assets which the logged-in user cannot see because of their where clause.
- If any FIELD element (in any ASSET element) does not refer to a valid Telescope metadata field, refers to a Telescope metadata field that the logged-in user does not have permission to see and edit, or refers to one of the 'predefined' or 'rendition' metadata fields which are marked as 'read-only'.

SOAP fault codes are as follows:

- TSWeb-0005 No MIMiX string has been provided.
- TSWeb-0022 Failed to parse MIMiX string.
- TSWeb-0001 No record ID value has been provided.
- TSWeb-0002 The record ID: {0} is invalid.
- TSWeb-2013 The field: {0} is not found or not visible for the user: {1}.
- TSWeb-2037 The field: {0} is not editable for the user: {1}.

- TSWeb-0031 The rendition ID: {0} is invalid.
- TSWeb-2012 The rendition ID {0} is not visible for the user: {1}.
- TSWeb-0032 The viewex_type: {0} does not match the other extended view elements in the MIMiX data.

Chapter 10: API Methods: Messaging Methods

These messaging methods allow a calling application to access, send and change Telescope messages.

In this Section:

10.1 DeleteMBMessage

```
public void DeleteMBMessage ( int[] in_seqMessages)
throws Throwable
```

This method deletes one or more Telescope messages.

Parameters:

- in_seqMessages – A list of the message IDs to be deleted.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

10.2 GetMBMessageAction

```
public void GetMBMessageAction    (    int in_iMsgID,
                                       String in_strActionCode)
throws                                 Throwable
```

This method is called when the recipient clicks on one of the message action buttons attached to a 'To Do' message. It sends the message action corresponding to the action button to the Message Broker which in turn performs the action script attached to the actions table.

Parameters:

- in_iMsgID – The integer message ID of the message whose action is being triggered.
- in_strActionCode – The code name of the message action whose button was clicked by the user.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

10.3 GetMBMessageCount

public com.northplains.IDLmodule.MessageBroker.MBMessageCount[] GetMBMessageCounts	(	int in_cutoffDate)
throws		Throwable

This method returns the number of sent and received messages currently in the system for a logged-in user, the number of unread messages, and a total count of messages after a given cutoff date. Messages whose send dates are later than this value are included in the returned counts, regardless of how many 'real' entries there are in the M_MESSAGES table. If this value is set to a future date/time, Message Broker will return zero for all counts.

Parameters:

- in_cutoffDate – A date/time giving the earliest send date the caller wants to see in seconds since Jan 1, 1970.

Returns:

A list of all messages with a sent date greater than the cutoff date. Each structure contains the category of messages (sent/received), the count of unread messages and a total count.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

10.4 GetMBMessageList

public com.northplains.IDLmodule.MessageBroker.MBMessageHeader[] GetMBMessageList	(	iint in_lCategory, int in_cutoffDate, int in_iSkip, int in_iNumResults)
throws		Throwable

This method returns the message IDs and header information for the messages in a specified category for the logged-in user.

Parameters:

- in_lCategory – The desired category of messages to return. Currently supported values are 1 = 'Received Items', or 2 = 'Sent Items'.
- in_cutoffDate – The earliest sent date the caller wants to see. Only messages whose sent dates are later than this value are returned, regardless of how many real entries there are in the M_MESSAGES table. The value should be provided in seconds since Jan 1, 1970.
- in_iSkip – The number of messages to skip in the returned, sorted list of message from the database, before beginning to return items to the caller. Passing 0 (zero) starts the list at the beginning.
- in_iNumResults – The number of desired result items. Passing 0 (zero) returns all items.

Returns:

Returns a list (sorted by date) of all message headers meeting the criteria. Each structure will contain the message id, subject, sent date, sender, list of recipients, read flag, and a flag indicating whether the message is a "TO DO" message.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

10.5 GetMBVisibleActions

```
public com.northplains.IDLmodule.MessageBroker.MBActionInfo[]    (    )
GetMBVisibleActions
throws                                                           Throwable
```

This method retrieves all of the message actions (from the M_ACTIONS table) that are visible to the calling user.

Returns:

Returns a list of all visible actions (in a code-name pair) for the logged-in user.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

10.6 ReadMBMessage

public com.northplains.IDLmodule.MessageBroker.MBMessageInfo ReadMBMessage	(	int in_iMsgID)
throws		Throwable

This method returns the full information for a given message.

Parameters:

- in_iMsgID – The requested message's ID field from the M_MESSAGES table. If a message cannot be found with this ID, Message Broker will raise an exception.

Returns:

Returns a structure containing the message ID, subject, body, sent date, sender, list of recipients, list of actions attached if any, (for a TODO message only) and list of attached records.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

10.7 SendMessage

```
public void SendMessage    (    String in_strToUser,
                                String in_strSubject,
                                String in_strMsgBody,
                                boolean in_bSendEmail)
throws                          Throwable
```

This call creates a Telescope message and optionally sends an e-mail as well. Attachments are not supported at present.

If the message or e-mail cannot be sent, or in case of any error including a failure for user privileges, this method throws an exception. The Telescope message is sent despite a failure to send the e-mail message.

Parameters:

- in_strToUser – A recipient user name. Must have a valid value.
- in_strSubject – Subject of the message. If this value is null, an empty string is used for the subject.
- in_strMsgBody – The message to be sent. If this value is null, an empty message is sent.
- in_bSendEmail š If true, this method tries to send e-mail to the recipient by using a SMTP host provided for the application instance as a command line argument for WOSMTPHost. WebObjects uses a default SMTP host with the name "SMTP" if no command line argument is provided for WOSMTPHost. The default SMTP host is not accepted by this method and the e-mail is rejected. In Telescope version 8.2 and later the use of this parameter has no effect on sending e-mails. It has been kept only for compatibility reasons.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

10.8 SendMBMessage

```
public int SendMBMessage    (   String[] in_arRecipients,
                                String in_strSubject,
                                String in_strMsgText,
                                String[] in_seqActions,
                                int[] in_seqAttachments)
throws                          Throwable
```

This method sends a Telescope message. This same method is used for sending reply messages and forwarding a message.

NOTE: This method supersedes the SendMessage method. The original method continues to be supported but is deprecated in the future versions of the Integration Broker SDK.

Parameters:

- in_arRecipients – A list of Telescope user names to send the message to.
- in_strSubject – A string representing the subject line of the message. This string can be empty.
- in_strMsgText – A string containing the full text of the message. Line breaks in the message is in the Windows style (CR + LF).
- in_seqActions – A list of valid message action code names, which the sender has permission to see. Setting this value turns the message into a 'To Do' message. Message Broker will raise an exception if an attempt is made to pass message actions to the MBSendMessage call when the MSGTODO license does not exist on the Session Broker.
- in_seqAttachments – A list of record_ids for documents which should be attached to the message.

Returns:

Returns the ID of the message sent.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

10.9 SendMBApprovalMessage

```
public int SendMBApprovalMessage    (    String[] in_seqRecipients,
                                         String in_strSubject,
                                         String in_strMsgText,
                                         int[] in_seqAttachments)
throws                                   Throwable
```

This method sends a Telescope approval message. This is a message which is generated automatically by Telescope when a user needs approval to perform an action, either because of their permissions, or because of a Functional Rule returning the needs approval result code.

Parameters:

- in_seqRecipients – A list of Telescope user names to send the message to.
- in_strSubject – A string representing the subject line of the message. This string can be empty.
- in_strMsgText – A string containing the full text of the message. Line breaks in the message is in the Windows style (CR + LF).
- in_seqAttachments – A list of record_ids for documents which should be attached to the message.

Returns:

Returns the ID of the message sent.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

Chapter 11: Search Methods

You can use the Integration Broker SDK to perform two types of searches: advanced searches based on metadata values and Tree searches that have been predefined in Telescope. Access to the Tree Broker is provided through the Integration Broker SOAP API as a thin layer over the Tree Broker IDL. For more information about creating Tree searches, see the Telescope Administrator's Guide.

In this Section:

11.1 Search

public int[] Search	(	NPSMap[] in_arCriterias)
throws		Throwable

This method searches the Telescope database, based on a set of criteria, and returns the found records. The authenticated user's where clause is applied to the search results.

NOTE: SOAP API search queries use SQL queries against the Telescope database (rather than Solr queries against the Solr database). To prevent timeouts and improve performance, you may want to limit the number of results returned by SQL queries. To do this, use the max_results field in the users table (see the Database Internals guide for details).

Parameters:

- in_arCriterias – An array of NPSMap objects with the following keys and values to build the SQL criteria to be executed against the database:
  - TABLE_NAME or tableName – the name of the table. If empty, assumes EDITORIAL entity.
  - COLUMN_NAME or columnName – the name of the column to execute the search for (Mandatory).
  - DATA_TYPE or dataType – the Telescope data type of the column as defined in the EXTRA_COLUMNS table (Optional).
  - OPERATOR or operator – operator selected for search such as LIKE, IS, IS NOT etc. (Mandatory).
  - CONJUNCTION or conjunction – the conjunction selected in case of multiple criteria such as OR or AND (Optional).
  - OPEN or open – represents an open bracket "(" (Optional). Set to "true" to put an open bracket before this criteria; otherwise set to "false".
  - CLOSED or closed - represents a close braket ")" (Optional). Set to "true" to put an close bracket after this criteria; otherwise set to "false".
  - VALUE or value – the value entered by the user. NULL used if empty.

Returns:

An array of integers representing the record IDs of found assets.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

11.2 Example of Constructing a Query in NPSMap Form

If the user wants to search for ('a' or 'b') in the Title field and 'c' in the Notes field, the application will need to create an array of 3 NPSMap objects, as illustrated in the structures below.

NOTE: The following order is important to keep!

```
NPSMap[1]: Part one of the query: "Editorial.Title IS ('a' OR"
Name = TABLE_NAME, Value = Editorial
Name = COLUMN_NAME, Value = 'Title'
Name = OPERATOR, Value = 'Is'
Name = OPEN_PAREN, Value = 'true'
Name = CONJUNCTION, Value = 'OR'
Name = VALUE, Value = 'a'

NMPSMap[2]: Part one of the query: "Editorial.Title IS 'b') AND"
Name = TABLE_NAME, Value = Editorial
Name = COLUMN_NAME, Value = 'Title'
Name = OPERATOR, Value = 'Is'
Name = CONJUNCTION, Value = 'AND'
Name = CLOSE_PAREN, Value = 'true'
Name = VALUE, Value = 'b'

NPSMap[3]: Part one of the query: "Editorial.Notes IS 'c'"
Name = TABLE_NAME, Value = Editorial
Name = COLUMN_NAME, Value = 'Notes'
Name = OPERATOR, Value = 'Is'
Name = VALUE, Value = 'c'
```

11.3 GetTBCriteriaValues

```
public NPSMapArray GetTBCriteriaValues    (    String in_wsSearchName,
                                               String[] in_seqLevelValues)
throws                                         Throwable
```

This method gets the tree models for the current user from the Tree Broker.

Parameters:

- in_wsSearchName – The tree search name for which we request the data.
- in_seqLevelValues – An array of String values (catalog names), which is the actual "path" of values converted to a String as the current user would "descend" from the top level of the search down to the current level for which the data is searched.

Returns:

Returns a list of structures from the Tree Broker containing the search information (field name, value, conjunction) ready to be passed to the Integration Broker Search() call.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

11.4 GetTBLevelData

```
public NPSLevelData GetTBLevelData    (    String in_wsSearchName,
                                           String[] in_seqLevelValues
throws                                     Throwable
```

This method returns the data from the Tree Broker for a given tree search for the current user.

Parameters:

- in_wsSearchName – The tree search name.
- in_seqLevelValues – An array of String values (catalog names), which is the actual "path" of values converted to String as the current user would "descend" from the top level of the search down to the current level for which the data is searched.

Returns:

Returns a structure containing information about the current level – whether the level is a leaf (terminal) level, whether is informational only or participates in the generated search criteria, and a list of possible nodes to be displayed to the search level.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

11.5 GetTBSearchNames

```
public String[] GetTBSearchNames (
throws                                Throwable
```

This method gets a list of the available tree search names for the logged-in user.

Returns:

Array tree search names.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

Chapter 12: User Maintenance

The SOAP API includes methods for creating and maintaining users, and for determining user visibility permissions.

To make the visibility permissions (required part of the administration of the catalog and message features) explicit in the Integration Broker API, two "enumerate" calls allow the caller to determine the visible groups and users available to the logged-in user.

In this Section:

12.1 EnumerateGroups

```
public String[] EnumerateGroups    (
throws                                  Throwable
```

This method returns a list of all of the user groups visible to the logged-in user.

Returns:

An array of groups to the logged-in user based on the VIEW_GROUPS table.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

12.2 EnumerateUsers

```
public NPSMapArray EnumerateUsers    (
throws                                    Throwable
```

This method returns a list of all of the users visible to the logged-in user.

Returns:

Returns a list of structures containing the information for each user visible to the logged-in user. The key values for each map are the same as the database column names from the Telescope USERS table, and each map will contain keys for every Telescope-defined column in the users table (that is, any fields manually added to the users table that are not defined in the Telescope database design documentation will not be included).

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

12.3 IsValidUser

public boolean IsValidUser (String in_strUserName)

Parameters:

- in_strUserName – A String object representing the user name to be looked up in the database.

Returns:

Returns a boolean value: true if the user exists, and false otherwise.

12.4 CreateUser

```
public void CreateUser    (    String in_strUserName,
                               String in_strPassword,
                               String in_strGroupName)
throws                         Throwable
```

This call creates a new user in the Telescope users table. The new user's permissions are determined by the group they are created in. Only a user authenticated as an administrator can make this call. This method can be used only inside response-request loop.

Parameters:

- in_strUserName – A user name for the new user.
- in_strPassword – A password string for the new user; if this value is null or an empty string, an empty password is created for the new user.
- in_strGroupName – A group name to be used for the new user; if this value is null or an empty string, a new user is created in the Default group. If the group with this value is not found, a template group name, defined in the SOAPParams.plist, is used as a new user group.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

12.5 DeleteUser

public void DeleteUser	(	String in_strUserName)
throws		Throwable

This method deletes a user from the users table in the Telescope database. Only users authenticated as administrators can perform this action.

Parameters:

- in_strUserName – The user name to be deleted.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

12.6 UpdateUserPassword

```
public void UpdateUserPassword    (    String in_strForUserName,
                                       String in_strNewPassword)
throws                                 Throwable
```

This method modifies the password associated with a given user name. Unless the logged-in user has administrative privileges, they can update only his or her own password. If the call is made to update another user password, the logged-in user must have administrative privileges.

Parameters:

- in_strForUserName – The user name of the user whose password is updated.
- in_strNewPassword – A new password for the user. If this value is null or an empty string, an empty password is created.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

Chapter 13: Version Control Methods

These methods allow the calling application to create and maintain asset versions.

In this Section:

13.1 CreateDerivativeFromVersion

```
public void CreateDerivativeFromVersion (  int in_iVersionID,
                                           boolean in_bCopyMetadata,
                                           String in_strMimiXMetadata)
throws                                     Throwable
```

This method creates a derivative from the current version.

Parameters:

- in_iVersionID – The ID of the version used to create a new asset.
- in_bCopyMetadata – A flag indicating whether the metadata is gathered from the version's master record or from the in_strMimiXMetadata parameter representing the MIMiX data; if true, MIMiX data is ignored and the data is taken from the parent asset based on the ed_versions.record_id value.
- in_strMimiXMetadata – Metadata for the version if in_bCopyMetadata is false.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

13.2 DeleteVersion

```
public void DeleteVersion    (    iint in_lVersionID)
throws                            Throwable
```

This method deletes the given version of an asset.

Parameters:

- in_lVersionID – The version ID of the ED_VERSIONS record that is deleted.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

13.3 DownloadVersion

```
public NPSMap DownloadVersion    (    Integer in_nVersionId,
                                      boolean in_bIsMacBinary,
                                      boolean in_bZipSingle)
throws                                Throwable
```

This method calls DLManager to download a specific version of an asset.

Parameters:

- in_nVersionId – The version_id from the ED_VERSIONS table of the asset to download.
- in_bIsMacBinary – Set to 'True' to download a file in Mac binary format; this flag applies to all the files in the in_arAssetsData array.
- in_bZipSingle – Set to 'True' to zip a file.

Returns:

- An NPSMap object with the following keys:

  DOWNLOAD_URL – A URL to the downloaded files

  TIMESTAMP – A String value of the number of seconds elapsed since Jan, 1st, 1970 12:00 AM UTC

  ERRORS – If there were any errors returned by DLManager and gracefully handled by this method, the return value will contain one more ERRORS keys containing an array (ArrayList) of strings that describe the errors. In case of a fatal error returned by DLManager, the return value will contain null value for the key DOWNLOAD_URL.

  If there were no errors during the download process, the ERRORS key still might be present with a null value or an empty array.

Throws:

- java.lang.Throwable – In case of errors, including validation errors. For more information, see Exception Handling.

NOTE: No exception is thrown if DLManager is unable to download the file(s). Such errors are returned as a part of the return value in the ERRORS key.

13.4 DownloadVersionStart

```
public NPSMap DownloadVersionStart    (    Integer in_nVersionId,
                                           boolean in_bIsMacBinary,
                                           boolean in_bZipSingle)
throws                                     Throwable
```

This method calls DLManager to asynchronously download a specific version of an asset.

Parameters:

- in_nVersionId – The version_id from the ED_VERSIONS table for the asset to download.
- in_bIsMacBinary – Set to 'True' to download a file in Mac binary format.
- in_bZipSingle – Set to 'True' to zip a file.

Returns:

A HashMap object with the key DOWNLOAD_KEY that contains a key identifying the current download. If there were any errors returned by DLManager and gracefully handled by this method, the return value will contain one more ERRORS keys containing an array (ArrayList) of strings that describe the errors. In case of a fatal error returned by DLManager, the return value will contain null value for the key DOWNLOAD_URL.

If there were no errors during the download process, the ERRORS key still might be present with a null value or an empty array.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

NOTE: No exception is thrown if DLManager is unable to download the file(s). Such errors are returned as a part of the return value for the key ERRORS.

13.5 DownloadVersionStatus

```
public NPSMap DownloadVersionStatus  (  String in_strDownloadKey)
throws                                  Throwable
```

This method returns the status of a downloading asset version from the download manager.

Parameters:

- in_strDownloadKey – A string value of the key for the download operation, obtained as the result returned from a DownloadStart call.

Returns:

Returns a HashMap similar to the DownloadStatus method.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

NOTE: No exception is thrown if DLManager is unable to download the file(s). Such errors are returned as a part of the return value in the ERRORS key.

13.6 GetAssetVersions

```
public NPSVersInfo[] GetAssetVersions    (    int in_iRecordID)
throws                                        Throwable
```

This method gets a list of all of the versions associated with an asset.

Parameters:

- in_iRecordID – The record ID of the asset whose versions are being requested. If this value doesn't refer to a valid asset in the database, or refers to an asset that the currently logged-in user cannot see, this method will raise a SOAP fault.

Returns:

Returns a list of NPVersInfo structures (mirroring the ED_VERSIONS table) containing the information about all versions associated with an asset. If the asset has no versions, the method will return normally with an empty array.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

13.7 PromoteVersion

public void PromoteVersion	(	int in_lVersionID)
throws		Throwable

This method promotes the version associated with the given version ID.

Parameters:

- in_lVersionID – The version ID of the ED_VERSIONS record to be promoted.

Throws:

- java.lang.Throwable – In case of errors, including validation errors.

Chapter 14: UI Service

This section provides information about the UI Service:

14.1 Overview

The UI Service allows user interface elements to be presented or provided by Telescope. This topic explains the UI Service direct action and its action codes.

A direct action in Telescope provides the functionality required by the UI Service. This direct action is called by a URL interface, with all the necessary parameters passed using the Get or Post method. The calling application is responsible for passing parameters that include:

- an ACTION that tells Telescope what action to perform
- CMDDATA for the command, which differs from action to action
- a RETURL to callback when the action is complete, usually to pass data back to the caller when appropriate

Below is an example of a URL complete with all required parameters using the Get method:

```
http://www.hostserver.com/scripts/WebObjects.dll/tsweb. woa/3/wa/services/
uiservice?wosid=ABCDEFGHIJK0123456789
&action=3&cmddata=FIELD&returl=http://www.return.net/blah
```

14.2 UI Service Actions

14.2.1 Home Page

Takes the user to their home page in Telescope.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/uiservice?wosid=ABCDEF&action=

Action		CMDDATA Parameter (optional)	RETURL Parameter
0	Go to home page	<IGNORED>	<IGNORED>

14.2.2 Browse and Select Files

Opens the passed-in search page, lets the user search and browse the returned results. A button in the standard Telescope result set page is used to return check-boxed assets to the caller.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/
uiservice?wosid=ABCDEF&action=1&cmddata=FIELD&returl=http://www.return.net/blah

Action		CMDDATA Parameter (optional)	RETURL Parameter
1	Browse and select files	{ FIELD | KEYWORD | BROWSE | FORM | CONTENT }	Called with record ID, file type, and file name values appended.

14.2.3 Search and Return Result Set

Opens the passed-in search page, lets the user search and browse the returned results. A button in the standard Telescope.web result set page is used to return the SQL used to arrive at the results to the caller.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/
uiservice?wosid=ABCDEF&action=2&cmddata=BROWSE&returl=http://www.return.net/blah

Action		CMDDATA Parameter (optional)	RETURL Parameter
2	Search and return result set	{ FIELD | KEYWORD | BROWSE | FORM | CONTENT }	Called with search SQL appended.

14.2.4 Return Thumbnail

Returns the thumbnail JPEG data for the passed-in record ID. This UI Service call can be embedded in an HTML <IMG> tag.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/uiservice?wosid=ABCDEF&action=3&cmddata=1234

Action		CMDDATA Parameter (optional)	RETURL Parameter
3	Return thumbnail	The record ID of the asset.	<IGNORED>

14.2.5 Return Rendering

Returns the file data directly in the response. This UI Service call can be embedded in an HTML <IMG> tag. CMDDATA contains record ID, rendition ID and conversion string as shown below.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/uiservice?wosid=ABCDEF&action=4&cmddata=345r1%20JPEG(DPI=72)

Action		CMDDATA Parameter (optional)	RETURL Parameter
4	Return asset rendering	A record ID/rendition ID pair that identifies the asset rendition to act on.	<IGNORED>

14.2.6 Display One Asset

Displays the standard Telescope document info window for the passed-in asset. The user can see document information, extended view, history, notes, etc. depending on their privilege.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/
uiservice?wosid=ABCDEF&action=5&cmddata=1234

Action		CMDDATA Parameter (optional)	RETURL Parameter
5	Display One Asset	The Record ID of the asset	<IGNORED>

14.2.7 Display Multiple Assets

Displays the default results page containing the assets representing the record IDs passed to the action. The result set is subject to standard where clause limitations, therefore, any record IDs that are not visible to the current user are removed from the displayed result set. The operations available to the user in the results view are based on the user's privileges. This view is the same one presented to the user if logging in manually.

This action is called like all other UI Service actions, by using the UI Service endpoint URL. An example URL used to call this service may look as follows:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/
uiservice?wosid=ABCDEF&action=6&cmddata=15687;15987;18954

In the event that the URL exceeds 255 bytes due to the number of record IDs being passed, a form should be used by the client of this service using the POST method. Below is an example of how to construct a form to pass in the same values passed using the URL in the previous example:

```
<form method="post" action="http://www.host.com/scripts/
WebObjects.dll/tsweb.woa/3/wa/services/
uiservice?wosid=ABCDEF">
<input type="hidden" name="action" value="6">
<input type="hidden" name="cmddata"
value="15687;15987;18954">
</form>
```

When the record IDs are displayed using the new service action, the last search results stored in the user session are updated with the current results.

Action		CMDDATA Parameter (optional)	RETURL Parameter
6	Display Multiple Assets	<list of semi-colon delimited record IDs> e.g. "15687;15987;18954"	<IGNORED>

14.2.8 Search Action

This action invokes a search action. It has no parameters and returns nothing.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/uiservice?wosid=ABCDEF&action=7

Action		CMDDATA Parameter (optional)	RETURL Parameter
7	Search action	<NOT USED>	<IGNORED>

14.2.9 File Drop Applet

This action renders the file drop applet for file ingestion. It has no parameters and returns nothing.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/uiservice?wosid=ABCDEF&action=8

Action		CMDDATA Parameter (optional)	RETURL Parameter
8	File drop applet	<NOT USED>	<IGNORED>

14.2.10 Display Assets on Home Page

This action displays assets on the home page. It has no parameters and returns nothing.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/uiservice?wosid=ABCDEF&action=9

Action		CMDDATA Parameter (optional)	RETURL Parameter
9	Display action	<NOT USED>	<IGNORED>

14.2.11 Display Asset Creation Screen

This action is used for rendering the metadata screen for the Telescope Uploader. It has no parameters and returns nothing.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/uiservice?wosid=ABCDEF&action=10

Action		CMDDATA Parameter (optional)	RETURL Parameter
10	Display Asset Creation Screen	<NOT USED>	<IGNORED>

14.2.12 Display Asset Search Results

This action is used to display asset search results. It has no parameters and returns nothing.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/uiservice?wosid=ABCDEF&action=11

Action		CMDDATA Parameter (optional)	RETURL Parameter
11	Display Asset Search Results	<NOT USED>	<IGNORED>

14.2.13 Display Catalog

This action is used for displaying catalogs. It has no parameters and returns nothing.

Example URL:

http://www.host.com/scripts/WebObjects.dll/tsweb.woa/3/wa/services/uiservice?wosid=ABCDEF&action=12

Action		CMDDATA Parameter (optional)	RETURL Parameter
12	Display Catalog	<NOT USED>	<IGNORED>

Chapter 15: Reference

This following sections includes reference materials for Exception Handling and SOAP fault codes:

15.1 Exception Handling

Any exceptions thrown by a web service are passed over the wire as SOAP faults. A SOAP fault typically contains the following:

- **Fault Code**: Identifier for the fault; for more information see SOAP Fault Codes
- **Fault String**: Short description of the fault
- **Fault Actor**: Entity that raised the fault
- **Fault Details**: Detailed description of the fault

Any error or exception that occurs in a method is presented as a SOAPException with the appropriate fault code and fault string and returned to the consumer as part of the SOAP response. It is up to the client of the web service to extract the fault from the SOAP response to notify the user.

All of the fault codes and strings (error messages) returned to the client as a result of incorrect information or failed functional rules are taken from a configuration file (SOAPErrorMessages.strings).

There are two kinds of exceptions that might be sent to the consumer as SOAP faults: Handled (user exceptions) and Unhandled (system/run-time exceptions). Handled exceptions are thrown by the application code intentionally if a validation fails or if the information provided is invalid. Unhandled exceptions are unanticipated run-time exceptions that may occur in the application.

To distinguish Handled from Unhandled exceptions, a fault code can hold a special notation that tells the consumer that the fault was raised as part of the user's input, for example. Fault codes are customized to contain a notation like TSWeb-<<error code>>, where error code is the type of exception. If any fault is received with this notation, it is considered a User Exception and the fault string associated with the fault contains the actual error message.

Unhandled exceptions are processed by the SOAP engine, which packages the exception in its own format and can be considered a general system exception.

15.2 SOAP Error Codes

Errors caused as a result of SOAP are written to the file SOAPMessages.plist, located in the `tsweb.woa\Contents\Resources` directory. SOAP faults are categorized as either User errors, Database errors, Validation errors, System errors, or Broker errors. Each error consists of a fault code and a string that describes the error.

NOTE: Error codes should not be changed or removed. The descriptions may be customized to suit your needs, as long as the meaning of the original string is retained. Only new error codes can be added.

15.2.1 User Errors

User errors occur as a result of invalid method calls (for example, wrong type of parameter, missing or null parameter values, missing or invalid values).

The following list shows original error codes shipped with the product:

0000 = "Your session has expired or invalid. Please login to use the service.";

0001 = "No record id value has been provided.";

0002 = "The record id: {0} is invalid.";

0003 = "No rendition id value has been provided.";

0004 = "No user name has been provided.";

0005 = "No MIMiX string has been provided.";

0006 = "No template name has been provided.";

0007 = "No template is found for the template name: {0} or template is empty.";

0008 = "No field name value has been provided.";

0009 = "No recipient name has been provided.";

0010 = "No asset data has been provided.";

0011 = "No search criteria have been provided.";

0012 = "No column name has been specified for criteria: {0}.";

0013 = "No URLs have been provided for ingest process.";

0014 = "Invalid syntax for URI: {0}.";

0015 = "URI: {0} does not contain a valid file name";

0016 = "The username and password entered are not valid. Please try again.";

0017 = "Unable to find a connection with name {0}.";

0018 = "Invalid date format. The date value must be in the format: {0}.";

0019 = "The value: {0} must be of type Integer.";

0020 = "The value: {0} must be of type Decimal.";

0021 = "The value: {0} must be of type Real.";

0022 = "Failed to parse MIMiX string.";

0023 = "Your where clause contains parameter replacements that cannot be resolved.\\nThis must be corrected before you will be permitted to log in.";

0024 = "There was an error contacting the Session Broker. Logins are disabled until the Session Broker is back in service.";

0025 = "The {0} user class is not licensed at this installation of TeleScope.web. This login has been disabled until an appropriate license is obtained.";

0026 = "There is already 1 {0} user logged in. Please try again later.";

0027 = "There are already {0} {1} users logged in. Please try again later.";

0028 = "License pool {0} is not recognized by the Session Broker. This login has been disabled until an appropriate license pool is set.";

0029 = "This Content Consumer user record has been tampered with in the database. Please contact your system administrator.";

0030 = "You currently have more Content Creator users logged in than your license allows. You will not be allowed to login until someone else logs out.";

0031 = "There are more Content Creator users listed in this database than are allowed by your software license. Please contact your system administrator.";

0032 = "The provided url at index {0} is empty.";

0034 = "Could not find or instantiate the authentication provider class: {0}.";

0035 = "Invalid or empty username was returned by provider.";

0036 = "No sufficient information has been provided to upload a file.";

0037 = "Missing or invalid version";

0038 = "Version name cannot be longer than 16 characters.";

0039 = "Version description cannot be longer than 255 characters.";

0040 = "MIMiX xml string doesn't contain any assets.";

0041 = "Record ID: {0} has more than one occurrence in the MIMiX string.";

0042 = "Version ID must be greater than 0.";

0043 = "No version found for the version id: {0}.";

0044 = "No versions found for record id: {0}";

0045 = "Missing or invalid parameter {0}";

0046 = "Authentication failed.";

0047 = "No file broker url has been provided.";

0048 = "Invalid file broker url: {0}";

0049 = "No attached data to request is found.";

15.2.2 Database Errors

Database errors occur as a result of database transactions (for example, unsupported data types or fields, failure to determine a group for the user).

The following list shows original error codes shipped with the product:

1001 = "Unable to create a user: {0}. Error: {1}.";

1002 = "Failed to determine a group for the user: {0}.";

1003 = "Unable to determine the data type for table name: {0} and column name: {1}.";

1004 = "Unable to convert data type: {0} into java.sql.Types.";

1005 = "Unable to convert value: {0} for java.sql.Types data type: {1}.";

1006 = "The data type value in the extra_columns table for the field: {0} is not defined.";

1007 = "The Telescope data type {0} is not supported.";

1008 = "The field: {0} is not supported.";

1009 = "Failed to create a download_queue record for record id: {0} and rendition id: {1}.";

1010 = "More than one record has been updated in the database for record id: {0}, rendition id: {1}, table name: {2}, column name: {3}.This is not allowed. The transaction has been rolled back.";

1011 = "Invalid data type for criteria: {0}.";

1012 = "Failed to get primary keys.";

1013 = "The given site name does not exist in the 'sites.plist' file.";

1014 = "There are no visible connections.";

15.2.3 Validation Errors

Validation errors occur when the Ingest Broker attempts to verify that the requested operation is allowed (for example, a user attempts to delete an asset without the required privilege).

The following list shows original error codes shipped with the product:

2001 = "The user {0} has no import privileges.";

2002 = "The privileges for the user: {0} do not include creating a new user in the group: {1}.";

2003 = "The privileges for the user: {0} do not include deleting an asset.";

2004 = "The privileges for the user: {0} do not include checking out an asset.";

2005 = "The privileges for the user: {0} do not include sending a message.";

2006 = "The privileges for the user: {0} do not include downloading files.";

2007 = "There are no visible assets for the user: {0}.";

2008 = "The privileges for the user: {0} do not include viewing the record id: {1}.";

2009 = "The privileges for the user: {0} do not include viewing the rendition id: {1}.";

2010 = "The user: {0} does not have administrative privileges.";

2011 = "There are no visible renditions for the user: {0}.";

2012 = "The rendition id {0} is not visible for the user: {1}";

2013 = "The field: {0} is not found or not visible for the user: {1}.";

2014 = "The asset with the record id: {0} is checked out by another user and cannot be deleted.";

2015 = "The asset for record_id: {0} is already checked out.";

2016 = "The asset for record id: {0} is not checked out.";

2017 = "The asset for record id: {0} is checked out by another user.";

2018 = "No asset is found for record_id: {0}.";

2019 = "Failed to execute functional rule with the error message: {0}.";

2020 = "No user record found for name: {0}.";

2021 = "The user name: {0} exceeds 32 characters, maximum allowed for the user name to be created.";

2022 = "The user name: {0} already exists";

2023 = "A group name with the same name as user name: {0} already exists.";

2024 = "No template group name is specified in the SOAPParams.plist file.";

2025 = "The recipient: {0} is not found.";

2026 = "Unable to send email. SMTP host is invalid or not defined.";

2027 = "Unable to send email to: {0}, because recipient does not have a valid email address.";

2028 = "Not all of the requested assets are visible for the user: {0}. Non-visible assets cannot be downloaded.";

2029 = "Could not get data from the doc_renditions table because the rendition id was not provided.";

2030 = "Unable to determine data type for criteria: {0}.The criteria is invalid or the field is not visible for the user.";

2031 = "Challenge Forms are not supported in functional rules using the SOAP API";

2032 = "Some of the assets are not visible for the user: {0} and cannot be downloaded.";

2033 = "The privileges for user: {0} do not include deleting a user: {1}.";

2034 = "The privileges for user: {0} do not include updating the password for user: {1}.";

2035 = "The privileges for user: {0} do not include any valid file migration policy.";

2036 = "The path {0} does not refer to a valid catalog, or the user does not have visibility over the catalog.";

2039 = "The privileges for user: {0} do not include the ability to access the properties of catalog {1}. Only the owning user (or an administrator who has user group visibility over the owning user) can access a catalog's properties.";

2040 = "One or more of the passed in record_id values refer to invalid asset(s), or asset(s) that the logged-in user cannot see.";

2041 = "The permissions for user {0} do not include the ability to edit the catalog {1}.";

2042 = "The new path {0} is a subpath of the intial path {1}.";

2043 = "The user {0} does not have visibility over catalog at the new path {1}, or the catalog at path {1} does not allow nesting.";

2044 = "The catalog cannot be moved to the new nesting level";

2045 = "The catalog name {0} already exists at this nesting level.";

2046 = "One or more of the passed in record_id values refer to invalid asset(s), assets(s) not contained whithin the given catalog, or asset(s) that the logged-in user cannot see.";

2047 = "The supplied password does not match the one associated with the catalog.";

2050 = "The ACL contains entries that refer to invalid user(s), or user(s) or group(s) that the logged-in user cannot see.";

2051 = "The privileges for user {0} do not include the ability to create shared catalogs.";

2052 = "The permissions for user {0} do not include the ability to delete the catalog {1}. Only the owning user (or an administrator who has user group visibility over the owning user) can delete a catalog.";

2053 = "There is no version available for the version_id {0}.";

2054 = "The privileges for the user: {0} do not include checking in an asset.";

2055 = "Some of the record_ids are invalid or not visible for the user: {0}.";

2056 = "Checkouts record doesn't have corresponding doc_renditions record.";

2057 = "Invalid doc_renditions record for record id: {0} and rendition id: {1}";

2058 = "No File Broker with name {0} was found on the Naming Service.";

2059 = "No file was found for current doc_rendition.file_location. Record id: {0}, rendition id: {1}.";

2060 = "Unable to determine a new version file name for original file name: {0}.";

2061 = "File broker failed to rename file.";

2062 = "Failed to copy file to File Broker.";

2063 = "The file: {0} has been already checked in.";

2064 = "The privileges for the user: {0} do not include modifying metadata.";

2065 = "The field: {0} is not editable for the user: {1}.";

2066 = "The privileges for the user: {0} do not include seeing versions.";

2067 = "The settings for your site do not include usage of the {0} conversion type code.";

2068 = "The asset for record id: {0} does not have viewex data type value defined.";

2069 = "The privileges for user: {0} do not include import with no migration policy.";

2070 = "The privileges for user: {0} do not include usage of the migration policy {1}.";

15.2.4 System Errors

System errors occur as a result of a system component being unavailable or not able to fulfill the request (for example, DLManager is not available, or is unable to create a directory).

The following list shows original error codes shipped with the product:

3001 = "Download Manager is not available. Please verify that the Download Manager is running and accessible.";

3002 = "Unable to create a directory: {0}.";

3003 = "Unable to create a unique temporary directory for ingest. The directory: {0} already exists.";

3004 = "Unable to create a unique temporary directory: {0} for ingest.";

3005 = "Unable to get file from: {0}. Error: {1}.";

3006 = "Failed to download file(s). Error: {0}.";

3007 = "Unable to send email to user: {0}. Error: {1}.";

3008 = "Invalid response recieved from the Download Manager. Please contact system administrator.";

3009 = "The administration settings, required to upload files, are not presently set for this database. Please contact your system administrator.";

3010 = "The supplied list of record_id values is too large.";

15.2.5 Broker Errors

Broker errors occur when using the services of file brokers during ingest and download operations (for example, the broker is unable to upload a file, or unable to create a file location for a file).

The following list shows original error codes shipped with the product:

4001 = "Unable to upload file: {0} to the file broker for file location: {1}.";

4002 = "Unable to create file location for file: {0}.";

4003 = "Failed to connect to the Ingest Broker.";

4004 = "An error occurred while contacting the Ingest Broker. Error: {0}";

4005 = "Your current Ingest Broker session is invalid - please disconnect and login again. If the problem persists, please contact your system administrator.";

4006 = "The Ingest Broker has failed to parse the MIMiX data - please retry the operation. If the problem persists, please contact your system administrator.";

4007 = "The File Migration Policy you selected is invalid – please retry the operation. If the problem persists, please contact your system administrator.";

4008 = "The file {0} does not exist or cannot be opened.";

4009 = "The file location specified in the selected File Migration Policy is incorrect – please retry the operation using another policy. If the problem persists, please contact your system administrator.";

4010 = "Your current session is invalid - please disconnect and login again. If the problem persists, please contact your system administrator.";

4011 = "Your current session is invalid - please disconnect and login again. If the problem persists, please contact your system administrator.";

4012 = "Your request cannot be found by the Ingest Broker – please retry the operation. If the problem persists, please contact your system administrator.";

4013 = "The current response has indicated an invalid file ordinal – please retry the operation. If the problem persists, please contact your system administrator.";

4014 = "You don’t have sufficient privileges to execute this operation. Please contact your system administrator.";

4015 = "The rendition you are trying to attach already exists.";

4016 = "The asset is no longer valid in the TeleScope database – please refresh your catalog and retry the operation using a different asset. If the problem persists, please contact your system administrator.";

4017 = "An unexpected call was encountered by the Ingest Broker. Please contact your system administrator.";

4018 = "The Ingest Broker returned invalid request id: {0}";

4019 = "The Ingest Broker returned the following errors: {0}";

4020 = "The Message Broker returned the following errors: {0}";

4021 = "One or more users in the recipients list are not valid TeleScope user_name values from the users table.";

4022 = "The passed-in message ID was not found in the m_messages table.";

4023 = "The action code name passed in in_strActionCode does not refer to a valid message action code name in the m_actions table.";

4024 = "This call was made without the MSGTODO license existing in Session Broker.";

4025 = "The Message Broker returned the following errors: {0}";

4026 = "This recipient has opened the message,and the message is no longer a ‘To Do’ message for the current recipient.";

4027 = "The Message Broker returned the following errors: {0}";

4028 = "The passed-in category ID is not one of the known values.";

4029 = "The Tree Search Broker returned the following errors: {0}";

4030 = "Failed to connect to the File Broker.";

15.2.6 General Errors

The following list shows original error codes shipped with the product:

5001 = "The ingest process has been interrupted. Error: {0}";

5002 = "The checkin process has been interrupted. Error: {0}";

15.2.7 Ingest Process Status Messages

The following list shows original error codes shipped with the product. These error codes apply only to IngestWithStatus calls.

6000 = "Checking files.";

6001 = "Updating files in db.";

6002 = "Checking for functional rules.";

6003 = "Challenge Form is not supported in SOAP calls.";

6004 = "Cleaning Up Derivatives.";

6005 = "Checking For Approval.";

6006 = "Approval Form is not supported in SOAP calls.";

6007 = "Waiting for file {0} transfer.";

6008 = "Generating Graphics for {0}";

6009 = "Executing functional rule for {0}";

6010 = "Waiting in queue";

6011 = "Ingest Broker is done.";

6012 = "Deleting Temporarily files from the File Broker.";

6013 = "Done deleting temporarily files from the File Broker.";

6014 = "Calling Ingest Broker.";

6015 = "Creating File Broker location for file {0}";

6016 = "Deleting temporarily TSWeb files.";

6017 = "Copying file {0} to file broker.";

6018 = "Copying file {0} to temporary directory.";

15.2.8 Unexpected Errors

The following error code appears for unexpected errors:

9999 = "Unexpected error.";