memoQ

memoQ Server Web Service Interface

KILGRAY
Translation Technologies

www.memoQ.com

memoQ Server Web Service Interface

Technical Specification

Versions

	Date	Change
v1.0	Nov 22, 2008	Initial version
v2.0 pre-release	April 13, 2010	Pre-release version for memoQ server 4.2
v2.0	April 18, 2010	Initial version for memoQ server 4.2
v4.2.12	June 24, 2010	Support added for adding language to project
v4.5	Oct 1, 2010	Initial version for memoQ server 4.5 - Support for light resources added - Extra documentation added related to languages and the naming of TMs, TBs and projects
4.5.21	Oct 27, 2010	- Support added for creating projects with Desktop Documents and document workflow status management - New option for statistics controlling whether to take white spaces into consideration in char counts.
4.5.65	Jan 22, 2011	Distribute project and document delivery
4.5.70	Apr 08, 2011	Information about MemoQServicesClient clarified
5.0	Jun 27 2011	Changes in memoQ server 5.0
5.0.22	Sept 12 2011	Support for creating web translation enabled project is added.
5.0.50	Oct 10, 2011	- Sessions, Login and logout functionality. - Error handling changes. - TM Lookup, concordance, add/update entry functionality; reverse TM support.
5.0.53	Nov 01, 2011	- Sessions, Login and logout functionality - LiveDocs support added - Error handling changes - TM Lookup, concordace, add/update entry functionality; reverse TM support - ELM support added
5.0.54		Project license permission can be added to users
	Nov 27, 2011	Documentation has been added for ServerProjectUserInfo.PermForLicense and ServerProjectUserInfoHeader.PermForLicense
6.0	June 16, 2012	Changes in memoQ server 6.0.
6.0 fix	Sept 20, 2012	Lost information has been re-added to this documentation.

6.0	Nov 12, 2012	Documentation for UserInfo.LanguagePairs has been added
6.2.14	Mar 4, 2013	Documentation for Asia Online Web Service API has been added ServerProjectService.SetProjectTBs behavior has been made consistent with the UI. TBService.ListTBs2 has been introduced.
6.5	Mar 25, 2013	New light resource types supported ServerProjectInfo extended with properties of projects.
6.5	Mar 25, 2013	Documentation for X-translate operation has been added This documentation has been extended to describe how to set/get permissions on light resources.
6.5	Apr 24, 2013	AddLanguageToProject functions has been extended with TB language check
6.7	Jul 11, 2013	New server project properties (delivery with QA errors, package creation) Package management and delivery operations User info extended with package-only users IServerProjectService interface has been extended whit the GetDocumentHistory function Server project TB handling changes for TB prioritization New functionality: alignment and export as TMX IceSpice support
6.7.2	Sep 17, 2013	Run QA and get QA report
6.8	Sept 10, 2013	Modifications in document history entity classes related to the group assignment changes
6.8.50	Nov 14, 2013	CAL related modifications New functionality: Post translation analysis reports
6.8.55	Feb 11, 2014	New option in pre-translation: copy source to target in certain rows
6.8.58	May 7, 2014	Alignment with segmentation and not only based on structure. Reimport of documents with resource guid.
7.0	June 1, 2014	New status has been introduced in ConfirmAndUpdateSegmentStatuses Master TM handling Creating server project changes.

		Image handling and localization in projects. New history item types. IServerProjectService.GetProject has been introduced Advanced document-user assignment modes (FirstAccept, GroupSourcing and subvendor group assignments) are supported.
7.0.50	June 16, 2014	New statistics and post-translation analysis options (cross file repeptition, repetition preference over 100%). memoQWebTrans URL added to document list. Callback to signal delivery. New function to return the list of the subvendor groups.
7.0.65	October 15, 2014	New ability to specify path to store as import path for documents.
7.0.67	November 18, 2014	ServerProjectInfo contains information about the status of the project (live or wrapped up)
7.5	September 01, 2014	New Confirm and Update options (RoleForConfirmation)
7.5.50	December 15, 2014	Allow same user in multiple roles and allow separate subvendor assignments in roles Subvendor group managment
7.5.61	June 12, 2015	Custom pre-translate parameters has been introduced in class PretranslateOptions
7.8.50	June 15, 2015	Template-based project creation and project template resource management Export of multilingual documents
7.8.53	August 5, 2015	Update heavy resource properties (TM, TB, Corpus) Get heavy resource info based on the guid of the resource
7.8.54	September 11, 2015	Wrap up project support
7.8.100	July 23, 2015	Support for new type of license pool in ELM licensing. TM API operations for lookup, concordance search, adding and updating entries are deprecated, the new memoQ Server HTTP API is to be used instead.

1. Introduction

This document is a technical specification for the memoQ Server web service interface (also known as web service API, or shortly WS API). This API provides access to memoQ Server functionality via standard web services technology, therefore enable integration with any platform supporting standard web services, including applications developed in Microsoft .NET or Java.

If you already have used a previous version of the API we recommend reading the following chapter: “6 Migration from previous versions”.

1.1 Different APIs for memoQ Server

memoQ Server has a new HTTP/REST based API for certain translator related functions starting version 7.8.100 besides the SOAP protocol based API described in this document. The two APIs have different application roles:

- WS API
  - Offers operations for management functionality for TMs/TBs/Security/Projects/etc.
  - Is based on classic Web Service SOAP messages that is best suited for enterprise level integration.
  - The caller of the WS API is a super user with unlimited privileges: no user level authentication or authorization is in place.
  - Although this API currently supports a few TM related operations that require user level authentication (lookup, concordance search, adding and updating entries), these are considered to be deprecated, and will be eliminated in memoQ Server version 8.0. The reason is that the new HTTI API “takes over” these operations.
  - Requires a server level license.
- HTTP API (sometimes referred to as REST API, though the term “REST” is not fully accurate)
  - This API offers translator related functionality: currently TM/TB operations, such as lookup, concordance, adding TM entries and similar operations are supported.
  - Is based on HTTP (simple http GET/POST, JSON objects) that is easily accessible for all kinds of different platforms, including Javascript and mobile phones.
  - The caller of the API acts as a specific user: the caller has to log on as a specific user, most user level permission checks are effective.
  - Requires license from a web or tpro license pool for each active connection instead of a server level license.

This document is about the WS API from now on, the HTTP API is described in another document.

1.2 Functionality provided by the WS API

A brief overview of functionality provided by the WS API:

Translation memory management

- Creating, publishing, deleting and listing translation memories
- Importing from and exporting to TMX files
- Lookup, concordance search, adding and updating entries operations are still present, but are deprecated, will be eliminated in 8.0. From now on the new memoQ Server HTTP API is to be used instead.

Term base management

- Creating, publishing, deleting and listing term bases
- Exporting to memoQ CVS format and MultiTerm format
- Importing from memoQ CVS format

Live documents

- Creating, publishing, deleting and listing corpora

Light resource management

TM and TB are called heavy resources. Opposed to them, the following entities belong to light resources: segmentation rules, auto translatables, non translatables, ignore lists, autocorrect lists, TM settings, filter configurations, keyboard shortcuts, export path rules and QA settings.

- Creating, publishing, cloning, deleting and listing light resources
- Importing from and exporting to memoQ resource format (different for each resource type)

Security

- User management
- Group management
- Permission management for translation memories and term bases
- Login and logout, user sessions. Deprecated, will be eliminated in 8.0. From now on the new memoQ Server HTTP API is to be used instead.
-

Server projects

- Server project management (creating, updating, listing, deleting, assigning users, etc.)
- Assigning resources (TM, TB, Corpora, light resources) to server projects
- Import and export of translation documents in primary (such as plain text, RTF, XML, etc.) and in bilingual formats for online server projects.
- Statistics
- Pre-translation
- X-translation
- Get document history
- Run QA and get a QA report
- Run, and retrieve the result of post-translation analysis

File management

- Upload of files in chunks for later processing (such as for document import)
- Download of file in chunks created by server operations (such as document export)

ELM support

- Managing license assignments (assigning licenses to users and listing assignments, etc.)
- Managing license permissions (adding license permissions to users/groups, listing license permissions, etc.)
- Listing project license permissions
- Listing ELM license pools

1.3 Requirements

Accessing the memoQ Server web service API requires the following:

- An installed an licensed copy of memoQ Server
- A special, "Web Service API" license

2. Technical aspects

2.1 Web service technology

The memoQ Server WS API is exposed based on the WS-I Basic Profile Specification 1.1 web service standard. This means supporting SOAP 1.1 for web service calls and WSDL 1.1 for schema definition.

The WS API is implemented using Windows Communication Foundation (WCF), which is part of .NET Framework from version 3.0. This offers the latest communication technology provided by Microsoft. Using WCF, services can be exposed in an interoperable way, such as web services based on the WS-I Basic Profile Specification 1.1. WCF is wire level compatible with ASMX web services. WCF also supports several advanced web service standards (most of the WS-*, including WS-Security). It is important to emphasize, that even though the web service interface has been implemented based on the Microsoft .NET Framework, it can be accessed from any programming environment supporting standard web services, such as Java.

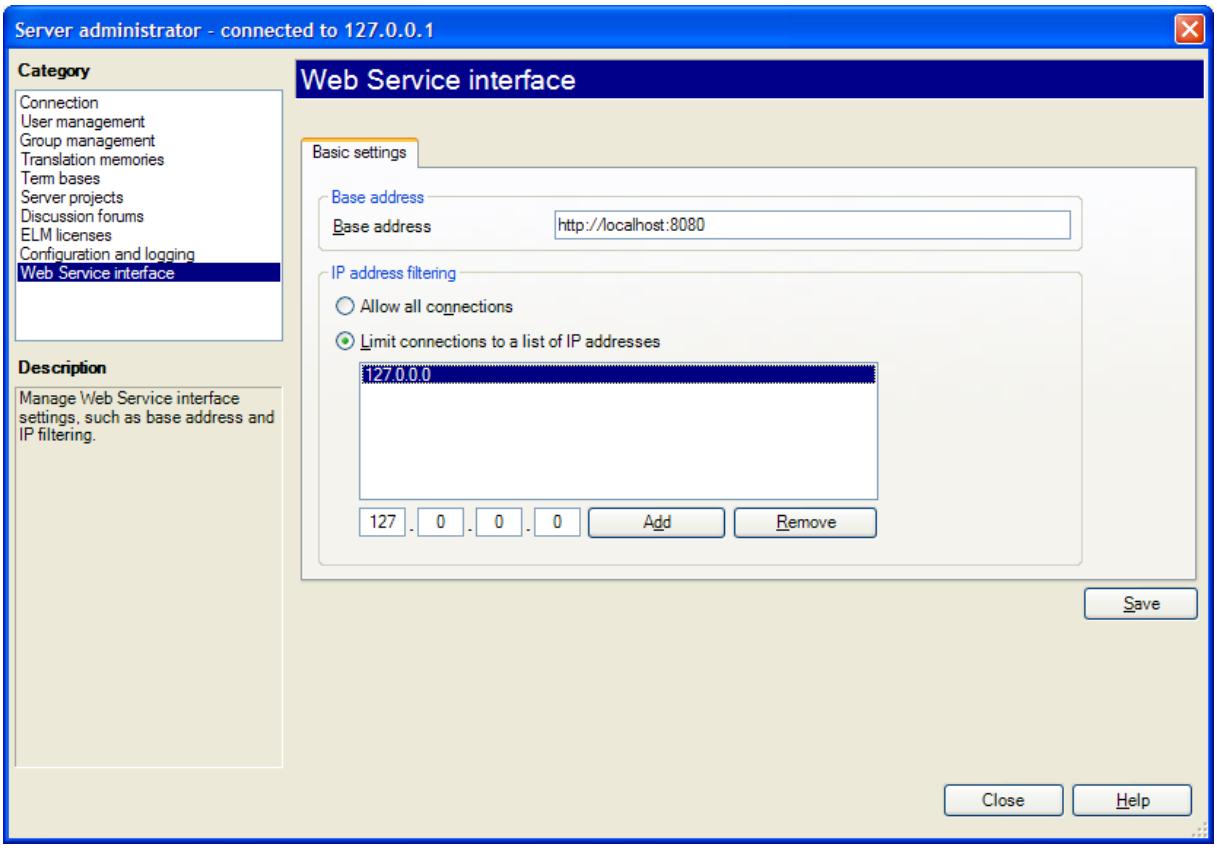
The services of the WS API are exposed by the existing memoQ Server component directly, which runs as a Windows Service. No additional components (such as an Internet Information Server web service application) are required. However, the web service interface is only available if your memoQ Server has a "Web Service API" license, and the Web Service add-in has been installed.

2.2 Security

There are several security scenarios for accessing memoQ Server. Most existing web service security standards (such as WS-I Basic Profile security, WS-Security, etc.) are supported. If memoQ Server is accessed from a .NET environment, other, non-standard options are also possible.

2.2.1 One simple scenario

The most likely scenario for the WS API is that the consumer is another server-based system that accesses memoQ Server from a protected intranet zone. In this case the default web service configuration options for memoQ Server are suitable: the security is basically turned off, and the "Web Service interface" page of the Server Administrator of memoQ Client can be used to enable web service access only for the other server-based system by enabling IP address filtering:



1. Figure – Configuring the Web service interface

The IP address filtering can be used for any security scenario.

2.2.2 Other security scenarios

It is possible to configure memoQ Server to use SSL for web service communication.

It is also possible to limit access to the web service based on the Windows identity of the consuming system if it supports Windows Integrated Security.

2.2.3 User level authorization

DEPRECATED, User level authorization and operations requiring user level authentication will be eliminated in 8.0

The memoQ Server's user permissions do not apply to **most** of the WS API "user." API access assumes unlimited trust, the WS API is a super user with unlimited privileges, **except** for the following functions:

- TMService/LookupSegment
- TMService/Concordance
- TMService/AddOrUpdateEntry

The functions require a memoQ user session and validate the user's permissions.

2.3 Accessing the services

The web service API is implemented by five different service endpoints, each accessible via a different URL:

- Translation memory management:
  Service URL: http://<base address>/memoqservices/tm/tmservice
  Metadata URL: http://<base address>/memoqservices/tm
- Term base management:
  Service URL: http://<base address>/memoqservices/tb/tbservice
  http://<base address>/memoqservices/tb
- Light resource management:
  Service URL: http://<base address>/memoqservices/resource/resourceservice
  http://<base address>/memoqservices/resource
- Security:
  Service URL: http://<base address>/memoqservices/security/securityservice
  Metadata URL: http://<base address>/memoqservices/security
- Server projects:
  Service URL:
  http://<base address>/memoqservices/serverproject/serverprojectservice
  Metadata URL: http://<base address>/memoqservices/serverproject
- File management:
  Service URL:
  http://<base address>/memoqservices/filemanager/filemanagerservice
  Metadata URL: http://<base address>/memoqservices/filemanager
- ELM management:
  Service URL:
  http://<base address>/memoqservices/elm/elmService
  Metadata URL: http://<base address>/memoqservices/elm

Each service endpoint has to ULRs: the *Service URL* is the actual address of the service, the *Metadata URL* is where its metadata can be accessed (see next chapter for details).

The base address that is part of the URLs can be configured using the "Web Service interface" page of the Server Administrator of memoQ Client, as can be seen in figure " *1. Figure – Configuring the Web service* interface". The default base address is http://localhost:8080.

2.4 Accessing metadata, code generation

To see the WSDL description of a service run *MemoQ Server (GUI).exe,* start a browser and type the Metadata URL of the specific service followed by "?wsdl" in the address bar.

In case of the security service type http://<base address>/memoqservices/security?wsdl. The XSD documents describing the message structures and data types participate as "import" in the WSDL document (try http://<base address>/memoqservices/security?xsd=xsd0 in the address bar). For the interpretation of the "<base address>" part of the URLs see chapter 2.3 Accessing the services.

The metadata is also exposed according to the WS-Metadata Exchange standard at the Metadata URL of the specific service (e.g. http://<base address>/memoqservices/security in case of the security service).

Techniques to generate the proxy class, the interfaces and the entity classes for the memoQ service API are the following:

- Any tool that can generate the interfaces/classes of the service based on WSDL/XSD or WS Metadata Exchange can be used. Please check the documentation of your development environment for information.
- For a WCF client, you can use svcutil.exe, or the Add Service Reference function of Visual Studio (the *MemoQ Server (GUI).exe* application has to be running!). The address to be provided is the Metadata URL of the specific service (e.g. http://<base address>/memoqservices/security in case of the security service).
- For an ASMX web client The *Add Web Reference …* function of Visual Studio (the *MemoQ Server (GUI).exe* application has to be running!), or the xsd.exe tool can be used. The address to be provided is the Metadata URL of the specific service (e.g. http://<base address>/memoqservices/security in case of the security service).

Important note: To generate the source code as described above, you have to use the GUI version of the memoQ Server. It works exactly the same way as the Windows Service version, the only difference is that it has a user interface with a log window. To start it, first stop the MemoQ Server Service, and then run *MemoQ Server (GUI).exe* from the memoQ application directory.

2.5 Error reporting

The way WS API services report errors has changed in version 5.0.23. From this point memoQ Server now supports two different modes:

- **Legacy mode**. This is the error handling mode memoQ Server supported in versions less than 5.0.23. Using this error mode there is no way to specifically handle different errors: all errors are grouped into expected and unexpected category, and only this category is returned to the caller. This error mode is kept only to preserve compatibility with previous versions, and may be eliminated in a future version. This error mode is described in details in chapter *2.5.4 Legacy mode*.
- **Advanced fault handing mode**. This new error handling mode is supported by memoQ Server version 5.0.23 and above. For most part of the API works the same

way as legacy mode (errors are grouped into expected and unexpected categories, and only the category is returned to the caller). But a subset of the API (TM lookup, ELM) enables the caller to differentiate between individual errors (and not only their category). This is the recommended mode for new development. This error mode is described in details in chapter *2.5.5. Advanced error handling mode*.

2.5.1 Error handling configuration

The error handling mode is set in the ProgramData\MemoQ Server\WSIFConfig.xml configuration file. To enable advanced error handling set the value of the AdvancedFaultHandling element to true. If this value is false (or the element is missing), the error handling mode is Legacy mode.

```
<?xml version="1.0" encoding="utf-8"?>
<WsIfConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  …
  <AdvancedFaultHandling>true</AdvancedFaultHandling>
</WsIfConfig>
```

There is an easier way to set the error handling mode using memoQ Client. Open Server Administrator, connect to memoQ Server, select the "Web Service interface" category, and check the "Use advanced fault handling" checkbox to enable advanced error handling (opposed to legacy mode).

The default error handling mode for clean installations is "Advanced fault handing mode". However, the error handling mode is "Legacy mode" when upgrading from previous versions.

2.5.2 Exception categories

Errors can be grouped into the following two categories.

1. **Expected errors/warnings:** These can happen under normal conditions. A few examples: trying to import into a read only TM, trying to modify a deleted project, etc. The text of the fault is a message that can be displayed to the user.
2. **Unexpected errors:** The origin of these errors can be:
    a. **The incorrect use of the API** (e.g., missing or incorrect parameters). Such errors are only expected during the development and testing phase of the module's deployment.
    b. **Unexpected errors during normal operation**. Such errors are unlikely; if they do occur, the project manager/system administrator needs to be notified. The cause of these problems can be network errors, inaccessible database, an unexpected failure in memoQ server, etc. If the problem cannot be solved locally, detailed report should be sent to Kilgray support.

   An error message something like "Internal memoQ Server error" should be displayed to the user. The information contained in the exception (fault) should not

be displayed to the user. Instead, the log of the memoQ Server should be used to determine the cause of the error.

2.5.3 Technology

Errors are reported as SOAP exceptions/faults using most client technologies. The SOAP standard defines the structure of SOAP fault messages. There are two SOAP standards (1.1 and 1.2), the structure of fault messages is slightly different for these. Luckily, most parts are the same, and memoQ Server utilizes only these parts:

- Fault code: A string code identifying the cause of the error.
- Message: A textual description of the problem
- Detail: Optional. Place to send extra structured information with the fault. This can be accesses using XML parser using old WS technologies, but modern tools generate classes/object (in the languages used) based on the fault contracts defined in WSDL, so it is possible to access this information in a typed manner. Certain technologies also enable to catch faults based on their fault type (which is defined in fault contracts).

The error handling - both in case of legacy and advanced mode - has been elaborated to support different platforms (ASMX web services, WCF web services, Java, etc.)

2.5.4 Legacy mode

Using this error mode there is no way to specifically handle different errors: all errors are grouped into expected and unexpected category, and only this category is returned to the caller.

Different web service platforms handle SOAP faults slightly differently (but typically an exception is thrown). This exception holds the fault code. The value of this fault code can be used to differentiate between expected/unexpected errors: for expected errors, the fault code name is "Server.Expected", for unexpected error it is a different string.

An example showing how to differentiate between expected/unexpected errors in WCF:

```
try
{
       // service call

}
catch (FaultException ex)
{
    if (ex.Code.Name == "Server.Expected")
       MessageBox.Show("Expected exception. The following message can be
              displayed to the user: \n\n" + ex.Message);
    else
        MessageBox.Show("Unexpected exception. See server log for
       details.");
}
```

An example showing how to differentiate between expected/unexpected errors in an ASMX client:

```
try
{
       // service call
```

```
}
catch (SoapException ex)
{
    if (ex.Code.Name == "Server.Expected")
        MessageBox.Show("Expected exception. The following message can be
       displayed to the user: \n\n" +
            ex.Message);
    else
        MessageBox.Show("Unexpected exception. See memoQ server log for
       details.\nMessage: " + ex.Message);
}
```

It should be noted, that memoQ Server logs all exceptions, and this log can be used to check for the internal details of the exceptions.

2.5.5 Advanced error handling mode

Note: Although the LookupSegment operation is used in the following chapter for demonstration, this operation is considered to be deprecated, and will be eliminated in 8.0. But the error handling concept/technique is general, also stands in other contexts.

This error handling mode also enables the caller to differentiate between expected and unexpected errors (though in a way different from the way legacy mode works). However, for a subset of the API, this mode also enables to differentiate between individual errors (not only their category), and this way enables the caller to handle them differently. For an example, an excerpt from the TM API documentation (the LookupSegment operation) is shown below:

```
[FaultContract(typeof(InvalidSessionIdFault))]
[FaultContract(typeof(UnauthorizedAccessFault))]
[FaultContract(typeof(NoLicenseFault))]
[FaultContract(typeof(TMFault))]
[FaultContract(typeof(RequestXmlFormatFault))]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
string LookupSegment( string sessionId, string lookupDataXml, Guid tmGuid,
        LookupSegmentRequest options );
```

The function above can return various types of faults. Before going through each of them, it is important to emphasize, that all of these faults have hold information defined in the SOAP standard (fault code, message, and the optional detail). Besides these, extra data is be defined for some of the faults: e.g. TMFault has a member, holding the TM specific error code. Modern WS technologies (e.g. WCF) make these data available, older ones (e.g. ASMX) does not. To enable at least to differentiate between the types of the faults for these old technologies, different fault code is used for different fault types. The fault code is similar to the name of the fault contract type for most cases. E.g. if the server throws InvalidSessionIdFault, the fault code is “InvalidSessionIdFault”. Some fault types hold an extra error code in the fault Detail. E.g. the TMFault fault type is used for different types of TM related faults: the TM does not exit, the TM is corrupt, etc. The ErrorCode member of the TMFault can be used to differentiate between them. However, older WS technologies do not make this extra data (ErrorCode in this case) available in an easy way: use the fault code in this case instead, which is different for each kind of TM related error, e.g. “TM+DoesNotExists” if the TM does not exist, “TM+Corrupt” if the TM is corrupt, and so on.

Now let's categorize the faults possibly returned by LookupSegment (and all other API operations).

2.5.5.1 Error categories

Unexpected server errors

For unexpected server errors **UnexpectedFault** is returned. All API functions return UnexpectedFault in this case in advanced error handling mode, even if the operation is not decorated by the FaultContract(typeof(UnexpectedFault))] attribute in this document! As you can see from the following declaration it has one member, holding the stack trace:

```
/// <summary>
/// Throw in case of unexpected errors
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class UnexpectedFault
{
    /// <summary>
    /// The stack trace on the memoQ server for the operation throwing
    /// the fault.
    /// </summary>
    [DataMember]
    public string StackTrace;
}
```

The fault code for these errors is "Unexpected". The message of the fault is the message of the original exception thrown inside memoQ Server. Do not display it to the user (display something like "Internal server error" instead"). To find the source of the problem, check memoQ Server error log, and the message of the fault.

Specific errors supported by fault contracts

These are different kinds of expected errors the operation is annotated with as fault contracts. For the above example these are **InvalidSessionIdFault**, **UnauthorizedAccessFault**, **NoLicenseFault**, **TMFault** and **RequestXmlFormatFault**. Please note that even though the operation is annotated with UnexpectedFault and GenericFault, we do not consider them as belonging into this category, as all API operations are annotated by them, and they are generic in this manner.

Uncategorized expected server errors (generic faults)

These are expected errors that are not distinguished based on their fault type. For these types of errors **GenericFault** is returned. This fault is used when an operation can run into different expected error scenarios, but not different fault contracts have been introduced for them. A few scenarios you can receive this fault: trying to modify a project that no longer exists, trying to import into a read only TM, etc.. The fault code for these errors is currently "Generic" for all operations. However, different fault codes may be introduced in the future (e.g. "Generic.UpdateError or "Generic+ProjectNotFound"): you can assume that the all start with the word "Generic". So if you want to catch generic faults based on their fault code, the recommended way to do it is to check that the fault code starts with "Generic".

The message of the fault is a message that can be displayed to the user. The message is localized based on the language of your memoQ Server.

You should be prepared to receive this kind of fault for all API operations (even if the operation is not decorated by the FaultContract(typeof(UnexpectedFault))] attribute in this document).

2.5.5.2 Recommendation

Advanced error handling mode enables you to catch and handle errors based on their specific type. However this is not necessary in most cases. One easy to implement solution is the following:

- For unexpected server errors you can show the user an “internal server error” screen
- For generic expected faults (GenericFault), you can display the message of the fault to the user in a user friendly error screen. This is not possible however if your application is localized to multiple languages.
- For specific errors supported by fault contracts: some may require special handling, but for most of them.
    - You can display the message of the fault to the user in a user friendly error screen
    - Or if you application is localized to multiple languages, you can display a message defined in your system based on the type of the fault or the value of the fault code.

You probably can use the same error handling logic for most of the API calls: implement it once in a central error handler function, call this same function whichever API method throws.

2.5.5.3 Code samples

WCF enables catching the faults based on their type:

```
try
{
    string res = tmService.LookupSegment(...);
}
catch (FaultException<InvalidSessionIdFault> e)
{
    // E.g. re-login user
}
catch (FaultException<RequestXmlFormatFault> e)
{
    // This is a kind of programmatic error, handle it specially
}
catch (FaultException<GenericFault> e)
{
    // Uncategorized expected server errors (generic faults)
    // E.g. display the message of the fault to the user
    // in a user friendly error screen.
}
catch (FaultException<UnexpectedFault> e)
{
    // Unexpected server errors
    // E.g. show the user an “internal server error” screen
    // Also log stack trace:
    Log.WriteError(e.Detail.StackTrace);
}
catch (FaultException e)
{
    // All other faults (UnauthorizedAccessFault, NoLicenseFault, TMFault),
    // handle them the same way
}
```

The last catch block catches all SOAP faults. This may include connectivity, and other errors, depending on the WS technology you use.

An alternative way to handle faults is based on their fault code (with older WS technologies this is the only alternative). The possible values of the fault codes are listed in the chapters describing the details of the API (e.g. “3.9.4 Reported errors for ELM”).

Using WCF the fault code is the Code.Name member of the caught FaultException object:

```
try
{
    // string res = tmService.LookupSegment(...);
}
catch (FaultException e)
{
    if (ex.Code.Name == "Unxecpected")
        Log.WriteError("Unexpected error occoured");
}
```

Now, the above complete error handling mechanism is implemented based on fault codes using ASMX technology:

```
try
{
    // string res = tmService.LookupSegment(...);
}
catch (SoapException e)
{
    // All other faults (UnauthorizedAccessFault, NoLicenseFault, TMFault),
    // handle them the same way
    switch (e.Code.Name)
    {
        case "InvalidSessionIdFault":
        // E.g. re-login user
            break;
        case "RequestXmlFormatFault":
            // This is a kind of programmatic error, handle it specially
            break;
        case "Unexpected":
            // Unexpected server errors
            // E.g. show the user an “internal server error” screen
            // The stack trace is in Detail in XML format, if you want it.
            break;
        default:
            if (e.Code.Name.StartsWith("Generic"))
            {
                // Uncategorized expected server errors (generic faults)
                // E.g. display the message of the fault to the user
                // in a user friendly error screen.
            }
            else
            {
                // All other faults (UnauthorizedAccessFault, NoLicenseFault,
                //  TMFault, etc.),
                // Handle them the same way
            }
            break;
    }
}
```

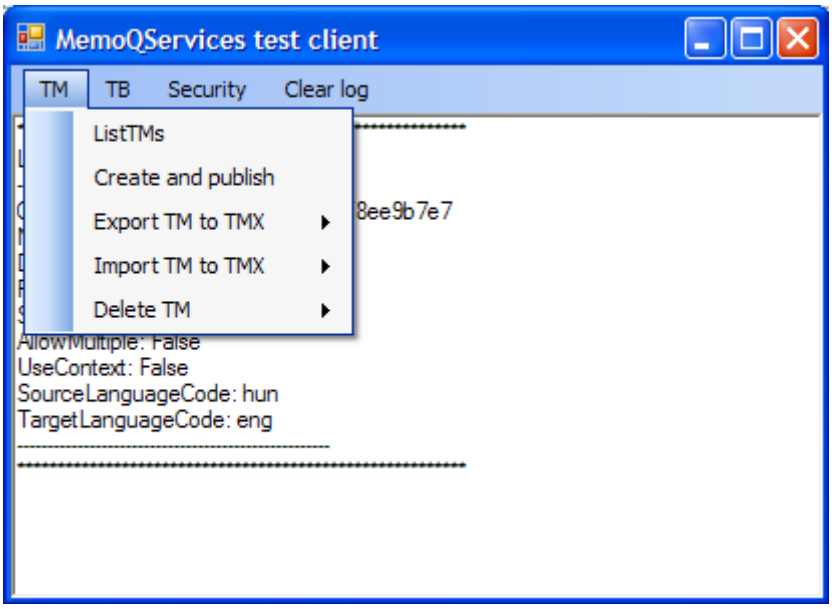
The default block catches all SOAP faults. This may include connectivity, and other errors, depending on the WS technology you use.

2.6 Testing the API configuration

MemoQServicesClient is a Windows Forms application that is bundled with this documentation in source code form. This tool can be used to test connectivity and certain functions of the WS API, and the source code should provide examples and directions for development. The main window of the application has the following main menu items:

- TM: It has submenus to test TM functionality (such as listing TMs, creating a TM, importing into a TM and so on);
- TB: It has submenus to test TB functionality (such as listing TBs, creating a TB, importing into a TB and so on);
- Security: It has submenus to test Security functionality (such as creating/listing users/groups, assigning users to groups, managing permissions, and so on)

The results of the operations are logged in a readable format in the main window:



2. Figure MemoQServices test client

It's important to remember that if you change the base address of the API services, then you have to change the endpoint addresses in MemoQServicesClient.exe.config for the test client tool.

The MemoQServicesClient tool demonstrates only the usage of the Translation Memory, Term base and Security API, the File Management and Server Project API is not covered. However, in chapter 5 *Sample code* the usage of these APIs is also demonstrated.

3. memoQ Server API services

3.1 Common information

3.1.1 Language codes

Several API functions expect language code(s) as parameter. Also, entity classes may have language code members. In all cases (unless indicated otherwise) three+(two) letter ISO language codes are to be used, such as fre, eng, eng-US. To find out which are the supported language codes see file LangInfo.xml under Documents and Settings\All Users\MemoQ. The three-letter language code is from ISO 639, and the two-letter country code is from ISO 3166. Both standards have different versions (e.g.: ISO 639-1, ISO 639-2, etc.). memoQ may use a mixture of these, therefore refer LangInfo.xml for the exact languages codes to be used.

3.2 Translation memory API

3.2.1 Overview

The translation memory API provides the following functionality:

- Provide the list of translation memories published by the memoQ Server
- Provide information about a specific TM
- Creating and publishing a TM
- Updating the properties of a TM
- Export the content of a specified translation memory in TMX format
- Import TMX documents into the specified already existing translation memory
- Perform lookup, concordance search, add/update entry in a TM. Deprecated. These will be eliminated in 8.0. The new memoQ Server HTTP API is to be used instead.

3.2.2 Brief description of the interface

The translation memory related interfaces and entity classes of the memoQ Server WS API are the following:

ResourceInfo: Base class for the `TMInfo` and `TBInfo` classes encapsulating common information for a TM/TB, such as its guid, name and description.

TMInfo: Encapsulates the description of a translation memory. This class is derived from `ResourceInfo`.

ITMService (possibly exposed as TMService proxy class): This interface has operations for listing TMs published by the server, importing TMX and exporting TMX. It also supports the creation of a TM.

3.2.3 Detailed description of the interface

Below is the C# code for the interfaces/classes with their description. The source is annotated by .NET Framework WCF attributes (ServiceContract, OperationContract, DataContract, DataMember, FaultContract), which can be ignored.

ITMService interface

```
/// <summary>
/// This interface has operations for listing TMs published by the server, importing
/// TMX and exporting TMX.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface ITMService
{
    /// <summary>
    /// Creates a new TM with the parameters specified by the info operation parameter.
    /// </summary>
    /// <param name="info">The parameters of the TM to be created. The Guid member
    /// of the info parameter is ignored.</param>
    /// <returns>The Guid of the newly created TM.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    Guid CreateAndPublish(TMInfo info);
    /// <summary>
    /// Returns the list of translation memories published by the memoQ Server.
    /// </summary>
    /// <param name="srcLang">Each memoQ TM can store segments for one language pair.
    /// This parameter can be used to filter the list based on the SOURCE language
    /// of the TMs. The three+(two) letter langusage code of the language
    /// is to be provided, such as fre, eng, eng-US. If a three letter language
    /// code is provided all three+two letter matches are returned (e.g. if eng is
    /// specified then TMs with source language eng- US, eng-GB, ... and eng are
    /// returned. If this parameter is null, no source language filtering is applied.
    /// Note: if a TM supports reverse lookup, it is returned if its source language
    /// matches the target language of the filter, and its target language matches
    /// the source language of the filter.
    /// </param>
    /// <param name="targetLang">Can be used to filter the returned list of TMs
    /// based on the TARGET language of the TM. The same rules are applied as for
    /// the srcLang parameter. It is allowed to provide null for both parameters
    /// or to provide not null for both. In the latter case only TMs satisfying
    /// both filter criteria are returned.</param>
    /// <returns>An array of TMInfo objects each describing one TM.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    TMInfo[] ListTMs(string srcLang, string targetLang);
    /// <summary>
    /// Gets information about the specified translation memory.
    /// </summary>
    /// <param name="tmGuid">The guid of the translation memory.</param>
    /// <returns>
    /// The TMInfo object describing the translation memory.
    /// </returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    TMInfo GetTMInfo(Guid tmGuid);
    /// <summary>
    /// Updates the properties of a translation memory.
    /// </summary>
    /// <param name="updateInfo">The new properties of the translation memory.
    /// The Guid member identifies the translation memory to be updated.</param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void UpdateProperties(TMUpdateInfo updateInfo);
    /// <summary>
    /// Deletes a TM.
    /// </summary>
    /// <param name="tmGuid">The guid of the TM to be deleted.
    /// </param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void DeleteTM(Guid tmGuid);
```

```
/// <summary>
/// Starts a new chunked TMX import session.
/// </summary>
/// <param name="tmGuid">The guid of an already existing TM into which the
/// TMX data is to be imported.
/// </param>
/// <returns>The session id (guid) of the newly started chunked TMX import session.
/// It should be provided as first parameter for the AddNextTMXChunk and
/// EndChunkedTMXImport operations.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
Guid BeginChunkedTMXImport(Guid tmGuid);
/// <summary>
/// Performs the import of a TMX file chunk. The operation
/// should be called in turns to import the next chunk of the TMX document.
/// It is not required that the chunk ends/begins at a valid TMX XML part.
/// The operation does not return until the segments included in the TMX chunk
/// are imported (except from the last partially provided segment of the chunk).
/// It is important that the interval between two AddNextTMXChunk calls
/// (and the interval between the call of BeginChunkedTMXImport and the first
/// call of AddNextTMXChunk) is less than a minute or two. If the interval
/// is larger, then the TMX import session times out on the server and reserved
/// resources (such as TM locks) are released, the import can not continue.
/// </summary>
/// <param name="sessionId">The session id (guid) of the chunked TMX import
/// session created by BeginChunkedTMXImport.</param>
/// <param name="tmxData">The next chunk of data to be imported into the TM.
/// The data size should be approximately 500 Kbytes (better estimation may
/// be required, approximately 10000 segments should be included). The encoding
/// of the chunk must be UTF-16 LE.</param>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
void AddNextTMXChunk(Guid sessionId, byte[] tmxData);
/// <summary>
/// Call to indicate to the MemoQ Server that all chunks have been sent
/// by calling AddNextTMXChunk. Closes the chunked TMX import session.
/// It is very important to call this method as soon as possible after importing the
/// last chunk of data to release server resources (such as TM locks).
/// </summary>
/// <param name="sessionId">The session id (guid) of the chunked TMX import session
/// to be closed. Always provide an id cerated by the BeginChunkedTMXImport operation.
/// </param>
/// <returns>Returns summary information about the import process.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
TmxImportResult EndChunkedTMXImport(Guid sessionId);
/// <summary>
/// Starts a new chunked TMX export session.
/// </summary>
/// <param name="tmGuid">The guid of the TM whose content is to be exported as TMX.
/// </param>
/// <returns>The session id (guid) of the newly started chunked TMX export session.
/// It should be provided as first parameter for the GetNextTMXChunk and
/// EndChunkedTMXExport operations.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
Guid BeginChunkedTMXExport(Guid tmGuid);
/// <summary>
/// Performs the export of a TMX document. The operation should be called in turns
/// to get the next chunk of the TMX document. It is important that the time interval
/// between two GetNextTMXChunk calls (and the interval between the
/// call of BeginChunkedTMXExport and the first call of GetNextTMXChunk) is less
/// than a minute or two. If the interval is larger, the TMX export session
/// times out on the server and reserved resourced (such as TM locks) are released,
/// the export can not continue.
/// It is not guaranteed that the chunk ends/begins at valid TMX XML part.
/// </summary>
/// <param name="sessionId">The session id (guid) of the chunked TMX import session
/// created by BeginChunkedTMXExport.</param>
/// <returns>The next chunk of TMX data exported from the TM. If no more data is
/// available, the operation returns null, or an empty array of bytes.</returns>
[OperationContract]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
byte[] GetNextTMXChunk(Guid sessionId);
/// <summary>
/// Call to close the chunked TMX export session.
/// It is very important to call this method as soon as possible after exporting
```

```
/// the last chunk of data to release server resources (such as TM locks).
/// </summary>
/// <param name="sessionId">The session id (guid) of the chunked TMX import session
/// to be closed. Alway provide an id cerated by the BeginChunkedTMXExport operation.
/// </param>
[OperationContract]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
void EndChunkedTMXExport(Guid sessionId);
/// <summary>
/// !!!! DEPRECATED !!!
/// Performs a fuzzy lookup for a single segment in a single TM published by the
/// server. This operation requires a valid user session. The operation may throw
/// various types of exceptions in case of error. The recognized errors are:
/// invalid session or session id, no valid license for this function, error
/// reported by the TM (such as reverse lookup requested but not supported),
/// input xml format error, and other uncategorized errors.
/// </summary>
/// <param name="sessionId">The session id belonging to the user who performs
/// the lookup. Permissions are checked against the user of this session.</param>
/// <param name="lookupDataXml">The xml document as a string, which contains
/// the single segment to lookup. For requirements and schema please consult the
/// documentation.</param>
/// <param name="tmGuid">The Guid of the single TM to perform the lookup
/// against.</param>
/// <param name="options">Other configuration options that specify the details
/// of the lookup.</param>
/// <returns>A valid Xml document as a string which containt the lookup results.
/// The string matched the schema in the documentation.</returns>
[FaultContract(typeof(InvalidSessionIdFault))]
[FaultContract(typeof(UnauthorizedAccessFault))]
[FaultContract(typeof(NoLicenseFault))]
[FaultContract(typeof(TMFault))]
[FaultContract(typeof(RequestXmlFormatFault))]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
string LookupSegment( string sessionId, string lookupDataXml, Guid tmGuid,
    LookupSegmentRequest options );
/// <summary>
/// !!!! DEPRECATED !!!
/// Performs a concordance search for a string expression in a single TM published
/// by the server. This operation requires a valid user session. The operation may
/// throw various types of exceptions in case of error. The recognized errors are:
/// invalid session or session id, no valid license for this function, error
/// reported by the TM (such as reverse lookup requested but not supported),
/// and other uncategorized errors.
/// </summary>
/// <param name="sessionId">The session id belonging to the user who performs
/// the lookup. Permissions are checked against the user of this session.</param>
/// <param name="searchExpression">A string search expression, including wildcard
/// characters.</param>
/// <param name="tmGuid">The Guid of the single TM to perform the concordance
/// against.</param>
/// <param name="options">Other configuration options that specify the details
/// of the concordance search.</param>
/// <returns>A valid Xml document as a string which containt the lookup results.
/// The string matched the schema in the documentation.</returns>
[FaultContract( typeof( InvalidSessionIdFault ) )]
[FaultContract( typeof( UnauthorizedAccessFault ) )]
[FaultContract( typeof( NoLicenseFault ) )]
[FaultContract( typeof( TMFault ) )]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
string Concordance( string sessionId, string searchExpression, Guid tmGuid,
    ConcordanceRequest options );
/// <summary>
/// !!!! DEPRECATED !!!
/// Adds or updates an existing entry to a TM published by the server. This
/// operation requires a valid user session. The operation may throw various
/// types of exceptions in case of error. The recognized errors are: invalid
/// session or session id, no valid license for this function, error reported
/// by the TM (such as reverse lookup requested but not supported), input xml
/// format error, and other uncategorized errors.
/// </summary>
/// <param name="sessionId">The session id belonging to the user who performs
/// the lookup. Permissions are checked against the user of this session.</param>
/// <param name="segmentDataXml">An xml document as a string wich contains a
/// single segment data to store/update in the TM. For requirements and schema
```

```
    /// please consult the documentation.</param>
    /// <param name="tmGuid">The Guid of the single TM to perform the concordance
    /// against.</param>
    [FaultContract( typeof( InvalidSessionIdFault ) )]
    [FaultContract( typeof( UnauthorizedAccessFault ) )]
    [FaultContract( typeof( NoLicenseFault ) )]
    [FaultContract( typeof( TMFault ) )]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
     void AddOrUpdateEntry( string sessionId, string segmentDataXml, Guid tmGuid );
}
```

ResourceInfo class

```
/// <summary>
/// Base class for resource information objects (such as the TMInfo
/// and TBInfo classes encapsulating common information for a TM/TB, such
/// as Guid, name and description).
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ResourceInfo
{
    /// <summary>
    /// The Guid of the resource (TM/TB, etc).
    /// </summary>
    [DataMember]
    public Guid Guid;
    /// <summary>
    /// The name of the resource (TM/TB, etc).
    /// The name can not contain characters that are invalid in file names.
    /// </summary>
    [DataMember]
    public string Name;
    /// <summary>
    /// The description of the resource (TM/TB, etc).
    /// </summary>
    [DataMember]
    public string Description;
    /// <summary>
    /// Indicates the readonly state of the resource (TM/TB, etc).
    /// </summary>
    [DataMember]
    public bool Readonly;

}.
```

HeavyResourceInfo class

```
/// <summary>
/// Base class for heavy resources (TMInfo and TBInfo classes) including
/// some meta data (Domain, Subject, etc.).
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class HeavyResourceInfo: ResourceInfo
{
    /// <summary>
    /// The domain attribute of the TM/TB.
    /// </summary>
    [DataMember]
    public string Domain;
    /// <summary>
    /// The subject attribute of the TM/TB.
    /// </summary>
    [DataMember]
    public string Subject;
    /// <summary>
    /// The client attribute of the TM/TB.
    /// </summary>
    [DataMember]
```

```
    public string Client;
    /// <summary>
    /// The project attribute of the TM/TB.
    /// </summary>
    [DataMember]
    public string Project;

}
```

Setting these fields is not required for most derived classes.

TMInfo class

```
/// <summary>
/// Encapsulates the description of a TM. This class is derived from ResourceInfo.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class TMInfo : HeavyResourceInfo
{
    /// <summary>
    /// Indicates if the TM can store formating information related to the segments.
    /// </summary>
    [DataMember]
    public bool StoreFormatting;
    /// <summary>
    /// Indicates if the TM can store more than one translation for a specific
    /// translation unit.
    /// </summary>
    [DataMember]
    public bool AllowMultiple;
    /// <summary>
    /// Indicates if the TM supports context information attached to the segments.
    /// If true then the previous segment and the following segment is attached
    /// to the source segment in the TM as context information.
    /// This context can be provided during lookup, and in case of matching
    /// context 101% hits are provided by the TM.
    /// </summary>
    [DataMember]
    public bool UseContext;
    /// <summary>
    /// Indicates whether the TM supports IceSpice context or not. If IceSpice
    /// context is supported the API retrieves 102% hits if the ID-based context
    /// and the text flow context are both identical.
    /// </summary>
    [DataMember]
    public bool UseIceSpiceContext;
    /// <summary>
    /// The three+(two) letter code of the source language of the TM
    /// (such as fre, eng, eng-US).
    /// </summary>
    [DataMember]
    public string SourceLanguageCode;
    /// <summary>
    /// The three+(two) letter code of the target language of the TM.
    /// (such as fre, eng, eng-US).
    /// </summary>
    [DataMember]
    public string TargetLanguageCode;
    /// <summary>
    /// Indicates if the TM allows reverse lookup (lookup in target language to
    /// provide hit for source language).
    /// </summary>
    [DataMember]
    public bool AllowReverseLookup;
}
```

ResourceUpdateInfo class

```
/// <summary>
/// Base class for resource property update objects (such as the TMInfo,
/// TBInfo and CorpusInfo classes encapsulating common information for a
/// TM/TB/Corpus, such as Guid, name and description).
/// </summary>
```

```
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ResourceUpdateInfo
{
    /// <summary>
    /// The Guid of the resource (TM/TB/Corpus, etc).
    /// </summary>
    [DataMember]
    public Guid Guid;
    /// <summary>
    /// The name of the resource (TM/TB/Corpus, etc).
    /// The name can not contain characters that are invalid in file
    /// names (e.g.: ",:,/,\,tab, ...).
    /// </summary>
    [DataMember]
    public string Name;
    /// <summary>
    /// The description of the resource (TM/TB/Corpus, etc).
    /// </summary>
    [DataMember]
    public string Description;
    /// <summary>
    /// Indicates the readonly state of the resource (TM/TB/Corpus, etc).
    /// </summary>
    [DataMember]
    public bool Readonly;
}
```

HeavyResourceUpdateInfo class

```
/// <summary>
/// Base class for heavy resource property update objects (TMInfo,
/// TBInfo and CorpusInfo classes) including some meta data (Domain,
/// Subject, etc.).
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class HeavyResourceUpdateInfo : ResourceUpdateInfo
{
    /// <summary>
    /// The domain attribute of the TM/TB/Corpus.
    /// </summary>
    [DataMember]
    public string Domain;
    /// <summary>
    /// The subject attribute of the TM/TB/Corpus.
    /// </summary>
    [DataMember]
    public string Subject;
    /// <summary>
    /// The client attribute of the TM/TB/Corpus.
    /// </summary>
    [DataMember]
    public string Client;
    /// <summary>
    /// The project attribute of the TM/TB/Corpus.
    /// </summary>
    [DataMember]
    public string Project;
}
```

TMUpdateInfo class

```
/// <summary>
/// Encapsulates the description of a TM. This class is derived
/// from ResourceUpdateInfo.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TMUpdateInfo : HeavyResourceUpdateInfo
{ }
```

TmxImportResult class

```
/// <summary>
/// Represents summary information about the TMX import process.
/// </summary>
public class TmxImportResult
```

```
{
    /// <summary>
    /// Number of segments (TU-s) contained in the TMX file that were successfully imported.
    /// </summary>
    int ImportedSegmentCount;
    /// <summary>
    /// Number of all segments (TU-s) contained in the TMX file (including those that could not
    /// be imported).
    /// </summary>
    int AllSegmentCount;
}
```

To get the list of TMs published by the server call the `ListTMs` operation of the `TMService` service.

To get information about a specific TM call the `GetTMInfo` operation of the `TMService` service.

To update the properties of an existing TM call the `UpdateProperties` operation of the `TMService` service.

TMX import and TMX export is performed in multiple chunks because TMX files can be of very large size. Sending/receiving of a TMX document as one web service call would require the entire document to be kept in memory by most web service platforms.

To perform TMX import the required steps are the following:

- Start a new TMX import session by calling `BeginChunkedTMXImport`.
- Import the chunks of a TMX document by calling `AddNextTMXChunk` in turns.
- Call `EndChunkedTMXImport` at the end to close the TMX import session.

It is crucial that the chunk boundaries are on whole characters. It should never happen that the end of a chunk contains the beginning of a unicode character and the beginning of the next chunk contains the remaining part! This can be easily implemented in .NET by using `StreamReader` to process the TMX file on the client.

The steps of TMX export are very similar:

- Start a new TMX export session by calling BeginChunkedTMXExport.
- Import the chunks of a TMX document by calling `GetNextTMXChunk` in turns.
- Call `EndChunkedTMXExport` at the end to close the TMX import session.

3.2.4 Lookup, concordance add/update entry - DEPRECATED

IMPORTANT: All operations described in this chapter are deprecetaed, and will be eliminated in 8.0. From now on the new memoQ Server HTTP API is to be used instead.

~~The lookup related services provide the following operations:~~

- ~~Lookup in a TM: performs a lookup in a single TM searching for similar segments and returns the matches with the stored translation.~~
- ~~Concordance: performs a free-text search in the TM and returns the matches.~~
- ~~Add/update entry in the TM: add a new entry (segment pair) to the TM or update an existing entry.~~

These functions use Xml documents for describing the lookup data, the returned match results and the segment content. These Xml documents are transported as strings through the API functions. The Xml documents are described by an XSD scheme detailed below. Every Xml document input is validated according to this scheme, and every result returned by the API in Xml form is guaranteed to conform to this scheme.

These functions require an existing and valid user session to the API in the name of a user who exists in the memoQ server. The permissions associated with the TM must allow the user access. For permission management please see Section 3.6 Security API**¡Error! No se encuentra el origen de la referencia.** and Section 3.6.4 for details about the user sessions.

A special license "MemoQServerWS-TM" is required in the memoQ server to access these functions. If the license is not present or not valid, the service returns a specific error.

3.2.4.1 Detailed description of the entity classes

Below is the C# code of the entity classes that are used in the lookup, concordance and add/update entry functions. These classes do not transport segment content but describe and specify the behavior of the operations.

LookupSegmentRequest class

```
///<summary>
/// Specifies TM lookup behavior details.
///</summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class LookupSegmentRequest
{
    ///<summary>
    /// Perform reverse lookup in the TM. The TM must support reverse lookup.
    ///</summary>
    [DataMember] public bool ReverseLookup;
    ///<summary>
    /// The match threashold for the lookup result.
    /// A number between 50 and 102 is expected (an exception is thrown otherwise).
    ///</summary>
    [DataMember] public int MatchThreshold;
    ///<summary>
    /// If true, only unambiguous mathes are returned.
    /// Only relevant if MatchThreshold is 100, 101 or 102.
    ///</summary>
    [DataMember] public bool OnlyUnambiguous;
    ///<summary>
    /// If true, an adjusment of fuzzy hits (punctuation, formatting etc.) is
    /// performed.
    ///</summary>
    [DataMember] public bool AdjustFuzzyMatches;
    ///<summary>
    /// The strieness of matching inline tags.
    ///</summary>
    [DataMember] public InlineTagStrictness InlineTagStrictness;
    ///<summary>
    /// Indicates whether or not only the TUs with the best match rate should be re
turned.
    /// Note that this may still mean several TUs with the same match rate.
    ///</summary>
    [DataMember] public bool OnlyBest;
}
```

ConcordanceRequest class

```
///<summary>
```

```
/// Specifies TM concordance search behavior details.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class ConcordanceRequest
{
    /// <summary>
    /// Perform reverse lookup in the TM. The TM must support reverse lookup.
    /// </summary>
    [DataMember] public bool ReverseLookup;
    /// <summary>
    /// Perform case sensitive search.
    /// </summary>
    [DataMember] public bool CaseSensitive;
    /// <summary>
    /// Check for numeric equivalence.
    /// </summary>
    [DataMember] public bool NumericEquivalence;
    /// <summary>
    /// The maximum number of results to return.
    /// </summary>
    [DataMember] public int ResultsLimit;
}
```

3.2.4.2 Xml format

The XSD scheme below describes the expected input format of the Xml documents and the format of the returned results. The segment content scheme is partially derived from the TMX 1.4b specification, but this schema allows a subset of what TMX allows in the TU elements.

Please note that a single scheme describes the following types of documents, but at all times only a single document is contained in the Xml strings (the XSD scheme verifies this too):

- Lookup request (root element: mq:lookup, input of function LookupSegment);
- Lookup results (root element: mq:fuzzy-results, output of function LookupSegment);
- Concordance results (root element: mq:concordance-results, output of function Concordance);
- Add/update entry (root element: mq:item-to-store, input of function AddOrUpdateEntry).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://kilgray.com/memoqservices/2007"
    xmlns:mq="http://kilgray.com/memoqservices/2007"
    targetNamespace="http://kilgray.com/memoqservices/2007"
    elementFormDefault="qualified">

  <!-- Lookup request -->
  <xs:complexType name="CT_LookupRequestItem">
    <xs:sequence>
      <xs:element name="context" minOccurs="0" maxOccurs="1" type="xs:string" />
      <xs:element name="context-pre" type="CT_Seg" minOccurs="0" maxOccurs="1" />
      <xs:element name="context-post" type="CT_Seg" minOccurs="0" maxOccurs="1" />
      <xs:element name="seg" type="CT_Seg" minOccurs="1" maxOccurs="1"
                  form="unqualified" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CT_LookupRequestRoot">
    <xs:sequence>
      <xs:element name="lookup-item" type="CT_LookupRequestItem"
                  minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
```

```
  </xs:complexType>
  <xs:element name="lookup" type="CT_LookupRequestRoot" />

  <!-- Lookup results -->
  <xs:complexType name="CT_LookupResultItem">
    <xs:sequence>
      <xs:element name="tu" type="CT_Tu" minOccurs="0" maxOccurs="unbounded"
                  form="unqualified" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CT_LookupResultRoot">
    <xs:sequence>
      <xs:element name="fuzzy-result-item" type="CT_LookupResultItem"
                  minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="fuzzy-results" type="CT_LookupResultRoot" />

  <!-- Concordance results -->
  <xs:complexType name="CT_ConcordanceItemRange">
    <xs:attribute name="position" type="xs:int" form="unqualified" use="required" />
    <xs:attribute name="length" type="xs:int" form="unqualified" use="required" />
  </xs:complexType>
  <xs:complexType name="CT_ConcordanceItem">
    <xs:sequence>
      <xs:element name="concordance-range" type="CT_ConcordanceItemRange"
                  minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="tu" type="CT_Tu" minOccurs="1" maxOccurs="1"
                  form="unqualified" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CT_ConcordanceResultRoot">
    <xs:sequence>
      <xs:element name="concordance-item" type="CT_ConcordanceItem"
                  minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="totalHitCount" type="xs:int" form="unqualified"
                  use="required" />
    <xs:attribute name="searchExpressionProblem" type="xs:string"
                  form="unqualified" use="optional" />
  </xs:complexType>
  <xs:element name="concordance-results" type="CT_ConcordanceResultRoot" />

  <!-- Item to store in TM -->
  <xs:complexType name="CT_ItemToStoreRoot">
    <xs:sequence>
      <xs:element name="context" minOccurs="0" maxOccurs="1" type="xs:string" />
      <xs:element name="context-pre" type="CT_Seg" minOccurs="0" maxOccurs="1" />
      <xs:element name="context-post" type="CT_Seg" minOccurs="0" maxOccurs="1" />
      <xs:element name="seg-source" type="CT_Seg" minOccurs="1" maxOccurs="1" />
      <xs:element name="seg-target" type="CT_Seg" minOccurs="1" maxOccurs="1" />
      <xs:element name="prop" type="CT_TuProp" minOccurs="0" maxOccurs="unbounded"
                  form="unqualified" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="item-to-store" type="CT_ItemToStoreRoot" />

  <!-- TU and TUV -->
  <xs:complexType name="CT_Tuv">
    <xs:sequence>
      <xs:element name="seg" type="CT_Seg" form="unqualified" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="CT_TuProp" mixed="true">
    <xs:attribute name="type" use="required" form="unqualified" />
  </xs:complexType>
  <xs:complexType name="CT_Tu">
    <xs:sequence>
      <xs:element name="prop" type="CT_TuProp" minOccurs="0" maxOccurs="unbounded"
                  form="unqualified" />
      <xs:element name="tuv" type="CT_Tuv" minOccurs="2" maxOccurs="2"
                  form="unqualified" />
    </xs:sequence>
    <xs:attribute name="tuid" form="unqualified" />
    <xs:attribute name="creationdate" form="unqualified" />
    <xs:attribute name="creationid" form="unqualified" />
```

```
    <xs:attribute name="changedate" form="unqualified" />
    <xs:attribute name="changeid" form="unqualified" />
    <xs:anyAttribute></xs:anyAttribute>
  </xs:complexType>

  <!-- Segment content -->
  <xs:complexType mixed="true" name="CT_Seg">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="G_Seg_Bpt"></xs:group>
      <xs:group ref="G_Seg_Ept"></xs:group>
      <xs:group ref="G_Seg_It"></xs:group>
      <xs:group ref="G_Seg_Ph"></xs:group>
    </xs:choice>
  </xs:complexType>
  <xs:group name="G_Seg_Bpt">
    <xs:sequence>
      <xs:element name="bpt" form="unqualified">
        <xs:complexType mixed="true">
          <xs:attribute name="i" use="required" type="xs:int" form="unqualified" />
          <xs:attribute name="type" use="required" form="unqualified">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="bold"></xs:enumeration>
                <xs:enumeration value="italic"></xs:enumeration>
                <xs:enumeration value="ulined"></xs:enumeration>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>
  <xs:group name="G_Seg_Ept">
    <xs:sequence>
      <xs:element name="ept" form="unqualified">
        <xs:complexType mixed="true">
          <xs:attribute name="i" use="required" type="xs:int" form="unqualified" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>
  <xs:group name="G_Seg_It">
    <xs:sequence>
      <xs:element name="it" form="unqualified">
        <xs:complexType mixed="true">
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>
  <xs:group name="G_Seg_Ph">
    <xs:sequence>
      <xs:element name="ph" form="unqualified">
        <xs:complexType mixed="true">
          <xs:attribute name="type" form="unqualified" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

</xs:schema>
```

3.2.4.3 Reported errors

Important note: When using Lookup, concordance add/update entry make sure to enable advanced error handling mode (see 2.5.1 Error handling configuration).

The functions above report various types of errors. Besides the generally used UnexpectedFault and GenericFault (see 2.5.2 Exception categoriesfor the details) several specific faults are used. There is one special: TMFault. It has an ErrorCode member so that the caller can differentiate between different types of TM related errors. Using older WS technologies it is difficult to access the ErrorCode: you can check the fault code instead,

which has the same value as the TMFault.ErrorCode. Possible fault types, fault code values with their fault message are the following:

Fault type	Fault code (same as ErrorCode in case of TMFault)	Fault message
NoLicenseFault	NoLicense	No valid license is available. (remark: using this API requires a special license)
UnauthorizedAccessFault	UnauthorizedAccess	Access denied. (remark: the specified user (memoQ user) has no rights to perform the request operation.)
InvalidSessionIdFault	InvalidSessionId	The session ID is invalid (not logged in or expired).
RequestXmlFormatFault	RequestXmlFormatError	The provided XML is not valid against schema.
TMFault	TM+DoesNotExists	The TM does not exist or is not published on the server.
	TM+NoReverseLookupIsSupported	Reverse lookup is not supported by the TM.
	TM+Busy	The TM is currently busy (exclusively used by an operation).
	TM+Readonly	The TM is read only.
	TM+Corrupt	The TM is corrupt, it has to be repaired.
GenericFault	Generic	<The message of the original exception is used as fault message.>
UnexpectedFault	Unexpected	<The message of the original exception is used as fault message.>

Please note that using C# and WCF these faults can be caught based on their type using regular exceptions:

```
try
{
    // call operation
}
// This catches TMFault errors only
catch (System.ServiceModel.FaultException<TMFault> e)
{

    // Check e.Code.Name or e.ErrorCode for the fault code
}
// This catches all other API errors
catch (System.ServiceModel.FaultException e)
{
```

```
    // Check e.Code.Name or e.ErrorCode for the fault code if you want to
    // differentiate between errors
}

Using older WS technologies, such as ASMX, you need to check the fault code:

try
{
    // call operation
}
// Important: this catches all API errors, not only TMFault errors!!!
catch (SoapException e)
{
    // Check e.Code.Name or e.ErrorCode for the fault code if you want to
    // differentiate between errors
}
```

3.2.4.4 Example code and Xml

The "memoQ services test client" bundled with this documentation contains example code on how to use these services. The code also contains simple examples of the Xml documents expected as input. These documents are assembled as a string in the example, however there are various other ways of creating Xml documents and exporting them as a single string. In .NET for example one could use the `XmlWriter`-class to create the document.

3.3 Term base API

3.3.1 Overview

The term base API provides the following functionality:

- Provide the list of term bases published by the memoQ Server
- Provide information about a specific TB
- Creating and publishing a TB
- Updating the properties of a TB
- Export the content of a specified term base in memoQ CVS format and MultiTerm format
- Import memoQ CSV documents into the specified existing term base

3.3.2 Brief description of the interface

The term base related interfaces and entity classes of the memoQ Server WS API are the following:

ResourceInfo: Base class for the `TMInfo` and `TBInfo` classes encapsulating common information for a TM/TB, such as its guid, name and description.

TBInfo: Encapsulates the description of a TB. This class is derived from `ResourceInfo`.

TBFilter: Represents a TB listing filter that can be used with ListTBs2 to filter the list of returned TBs.

TBFilterLangMode: Defines how the provided languages are used when filtering TB lists.

ITBService This interface has operations for listing TBs published by the server, importing CSV and exporting CSV. It also supports the creation of a TB.

CSVImportResult: Provides summary information of a CSV import session.

3.3.3 Detailed description of the interface

Below is the C# code for the interfaces/classes with their description. The source is annotated by .NET Framework WCF attributes (ServiceContract, OperationContract, DataContract, DataMember), which can be ignored.

ITBService interface

```
/// <summary>
/// This interface has operations for listing TBs published by the server, importing
/// CSV and exporting CSV.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface ITBService
{
    /// <summary>
    /// Creates a new TB with the parameters specified by the info operation parameter.
    /// </summary>
    /// <param name="info">The parameters of the TB to be created. The Guid member
    /// of the info parameter is ignored.</param>
    /// <returns>The Guid of the newly created TB.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    Guid CreateAndPublish(TBInfo info);
    /// <summary>
    /// Returns the list of termbases published by the MemoQ Server.
    /// </summary>
    /// <param name="languages">Each MemoQ termbase can store terms for multiple
    /// languages. This parameter can be used to filter the returned termbase
    /// list based based on the language of the terms. Only termbases that
    /// have at least one term of the specified languages is included in the
    /// result list. The three+(two) letter langusage code of the language
    /// is to be provided, such as fre, eng, eng-US. If a three letter language
    /// code is provided all three+two letter matches are returned (e.g. if eng is
    /// specified then TBs with language eng-US, eng-GB, ... and eng are
    /// returned. If this parameter is null, no language filtering is applied.
    /// </param>
    /// <returns>An array of TBInfo objects each describing one TB.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    TBInfo[] ListTBs(string[] languages);
    /// <summary>
    /// Returns a filtered list of TBs published by the memoQ Server.
    /// </summary>
    /// <param name="tbFilter">The filter used to filter the TB list. See class
    /// TBFilter for details. If null, no filtering is performed.
    /// </param>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    TBInfo[] ListTBs2(TBFilter tbFilter);
    /// Gets information about the specified term base.
    /// </summary>
    /// <param name="tmGuid">The guid of the term base.</param>
    /// <returns>
    /// The TBInfo object describing the term base.
    /// </returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    TBInfo GetTBInfo(Guid tbGuid);
    /// <summary>
    /// Updates the properties of a term base.
    /// </summary>
    /// <param name="updateInfo">The new properties of the term base.
    /// The Guid member identifies the term base to be updated.</param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
```

```
void UpdateProperties(TBUpdateInfo updateInfo);
/// <summary>
/// Deletes a TB.
/// </summary>
/// <param name="tbGuid">The guid of the TB to be deleted.
/// </param>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
void DeleteTB(Guid tbGuid);
/// <summary>
/// Starts a new chunked CSV import session.
/// </summary>
/// <param name="tbGuid">The guid of an already existing TB into which the
/// CSV data is to be imported.
/// </param>
/// <returns>The session id (guid) of the newly started chunked CSV import session.
/// It should be provided as first parameter for the AddNextCSVChunk and
/// EndChunkedCSVImport operations.</returns>
/// <summary>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
Guid BeginChunkedCSVImport(Guid tbGuid);
/// <summary>
/// OBSOLETE, PLEASE USE FUNCTION GetNextExportChunk INSTEAD
/// Performs the import of a CSV file chunk. The operation
/// should be called in turns to import the next chunk of the CSV document.
/// It is not required that the chunk ends/begins at a valid CSV entry part.
/// The operation does not return until the rows included in the CSV chunk
/// are imported (except from the last partially provided row of the chunk).
/// It is important that the interval between two AddNextCSVChunk calls
/// (and the interval between the call of BeginChunkedCSVImport and the first
/// call of AddNextCSVChunk) is less than a minute or two. If the interval
/// is larger, then the CSV import session times out on the server and reserved
/// resources (such as TB locks) are released, the import can not continue.
/// </summary>
/// <param name="sessionId">The session id (guid) of the chunked CSV import session
/// created by BeginChunkedCSVImport.</param>
/// <param name="data">The next chunk of data to be imported into the TB.</param>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
void AddNextCSVChunk(Guid sessionId, byte[] data);
/// <summary>
/// OBSOLETE, PLEASE USE FUNCTION EndChunkedExport INSTEAD
/// Call to indicate to the MemoQ Server that all chunks have been sent
/// by calling AddNextCSVChunk. Closes the chunked CSV import session.
/// It is very important to call this method as soon as possible after importing the
/// last chunk of data to release server resources (such as TB locks).
/// </summary>
/// <param name="sessionId">The session id (guid) of the chunked CSV import session
/// to be closed. Always provide an id created by the BeginChunkedCSVImport operation.
/// </param>
/// <returns>Returns summary information about the import process.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
CSVImportResult EndChunkedCSVImport(Guid sessionId);
/// <summary>
/// Starts a new chunked CSV export session.
/// </summary>
/// <param name="tbGuid">The guid of the TB whose content is to be exported as CSV.
/// </param>
/// <returns>The session id (guid) of the newly started chunked CSV export session.
/// It should be provided as first parameter for the GetNextCSVChunk and
/// EndChunkedCSVExport operations.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
Guid BeginChunkedCSVExport(Guid tbGuid);
/// <summary>
/// Starts a new chunked MultiTerm export session.
/// </summary>
/// <param name="tbGuid">The guid of the TB whose content is to be exported in
/// MultiTerm format.
/// </param>
/// <param name="xdl">The XDL descriptor of the MultiTerm export, as a string
/// out parameter.</param>
/// <param name="xdt">The XDT descriptor of the MultiTerm export, as a string
/// out parameter.</param>
/// <returns>The session id (guid) of the newly started chunked export session.
```

```
    /// It should be provided as first parameter for the GetNextExportChunk and
    /// EndChunkedExport operations.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    Guid BeginChunkedMultiTermExport( Guid tbGuid, out string xdt, out string xdl );
    /// <summary>
    /// Performs the export of a Term Base. The operation should be called in turns
    /// to get the next chunk of the CSV/MultiTerm document. It is important that the
    /// time interval between two GetNextExportChunk calls (and the interval between the
    /// call of BeginChunked*Export and the first call of GetNextExportChunk) is less
    /// than a minute or two. If the interval is larger, the export session times out
    /// on the server and reserved resourced (such as TB locks) are released,
    /// the export can not continue.
    /// It is not guaranteed that the chunk ends/begins at valid entry part specific
    /// to the document type.
    /// </summary>
    /// <param name="sessionId">The session id (guid) of the chunked export session
    /// created by BeginChunked*Export. In can either be CSV or MultiTerm export
    /// session.</param>
    /// <returns>The next chunk of data exported from the TB, in the specific format
    /// as defined by the initial call to the BeginChucnked*Export function. If no
    /// more data is available, the operation returns null, or an empty array of
    /// bytes.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    byte[] GetNextExportChunk( Guid sessionId );
    /// <summary>
    /// Call to close the chunked TB export session.
    /// It is very important to call this method as soon as possible after exporting
    /// the last chunk of data to release server resources (such as TB locks).
    /// </summary>
    /// <param name="sessionId">The session id (guid) of the chunked import session
    /// to be closed (may it be CSV or MultiTerm session). Always provide an id
    /// created by the BeginChunked*Export operation.
    /// </param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void EndChunkedExport( Guid sessionId );
    /// <summary>
    /// Performs the export of a CSV document. The operation should be called in turns
    /// to get the next chunk of the CSV document. It is important that the time interval
    /// between two GetNextCSVChunk calls (and the interval between the
    /// call of BeginChunkedCSVExport and the first call of GetNextCSVChunk) is less
    /// than a minute or two. If the interval is larger, the CSV export session
    /// times out on the server and reserved resourced (such as TB locks) are released,
    /// the export can not continue.
    /// It is not guaranteed that the chunk ends/begins at valid CSV entry part.
    /// </summary>
    /// <param name="sessionId">The session id (guid) of the chunked CSV import session
    /// created by BeginChunkedCSVExport.</param>
    /// <returns>The next chunk of CSV data exported from the TB. If no more data is
    /// available, the operation returns null, or an empty array of bytes.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    byte[] GetNextCSVChunk(Guid sessionId);
    /// <summary>
    /// Call to close the chunked CSV export session.
    /// It is very important to call this method as soon as possible after exporting
    /// the last chunk of data to release server resources (such as TB locks).
    /// </summary>
    /// <param name="sessionId">The session id (guid) of the chunked CSV import session
    /// to be closed. Alway provide an id cerated by the BeginChunkedCSVExport operation.
    /// </param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void EndChunkedCSVExport(Guid sessionId);
}
```

ResourceInfo class

See chapter *3.2.3 Detailed description of the interface*.

HeavyResourceInfo class

See chapter *3.2.3 Detailed description of the interface*.

TBInfo class

```
/// <summary>
/// Encapsulates the description of a TB. This class is derived from ResourceInfo.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TBInfo : HeavyResourceInfo
{
    /// <summary>
    /// Contains the three+(two) letter languages codes for the termbase.
    /// A termbase can support more than two languages but should support at least two.
    /// It is possible to add new languages after creating using the MemoQ client.
    /// </summary>
    [DataMember]
    public string[] LanguageCodes;
    /// <summary>
    /// Indicates if the TB is moderated or not. Should be false for a normal TB.
    /// </summary>
    [DataMember]
    public bool IsModerated;
    /// <summary>
    /// Only applicable if TB is moderated; late disclosure means others only see changes
    /// after moderation. The default value should be true.
    /// </summary>
    public bool ModLateDisclosure;
}
```

ResourceUpdateInfo class

See chapter *3.2.3 Detailed description of the interface*.

HeavyResourceUpdateInfo class

See chapter *3.2.3 Detailed description of the interface*.

TBUpdateInfo class

```
/// <summary>
/// Encapsulates the description of a TB. This class is derived
/// from ResourceUpdateInfo.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TBUpdateInfo : HeavyResourceUpdateInfo
{ }
```

TBFilter class

```
/// <summary>
/// Represents a TB listing filter that can be used with ListTBs2 to filter the list
/// of returned TBs.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TBFilter
{
    /// <summary>
    /// The three+(two) letter languages codes of the languages used to filter
    /// the list of TBs. A lazy match is performed when matching filter language
    /// codes with TB language codes: e.g. filter "eng" is a match for TB language
    /// "eng-US" and filter "eng-US" is also a match for TB language "eng".
    /// If null, no language filtering is performed.
    /// </summary>
    [DataMember]
    public string[] LanguageCodes;
    /// <summary>
    /// Describes how the languages in the LanguageCodes list are used to
    /// filter the TBs. See TBFilterLangMode for details.
    /// </summary>
```

```
    [DataMember]
    public TBFilterLangMode LangMode;
}
```

TBFilterLangMode

```
/// <summary>
/// Defines how the provided languages are used when filtering TB lists.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public enum TBFilterLangMode
{
    /// <summary>
    /// A TB is returned if it has any of the languages of the filter's
    /// language list.
    /// </summary>
    [EnumMember]
    AnyMatch,
    /// <summary>
    /// A TB is returned if it has all of the languages of the filter's
    /// language list.
    /// </summary>
    [EnumMember]
    AllMatch
}
```

CSVImportResult class

```
/// <summary>
/// Represents summary information about a TB import process.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class CSVImportResult
{
    /// <summary>
    /// Number of entries contained in the CSV file that could be successfully imported.
    /// </summary>
    [DataMember]
    public int SuccessCount;
    /// <summary>
    /// Number of all entries contained in the CSV file (including those that could not
    /// be imported).
    /// </summary>
    [DataMember]
    public int AllCount;

}
```

To get the list of TBs published by the server call the `ListTBs` operation of the `TBService` service. To get an unfiltered list of the Term Bases, submit `null` parameter to the function.

To get information about a specific TB call the `GetTBInfo` operation of the `TBService` service.

To update the properties of an existing TB call the `UpdateProperties` operation of the `TBService` service.

CSV import and CSV export is performed in multiple chunks because CSV files can be of very large size. Sending/receiving of a CSV document as one web service call would require the entire document to be kept in memory by most web service platforms.

To perform CSV import the required steps are the following:

- Start a new CSV import session by calling BeginChunkedCSVImport.
- Import the chunks of a CSV document by calling `AddNextCSVChunk` in turns.
- Call `EndChunkedCSVImport` at the end to close the CSV import session.

It is crucial that the chunk boundaries are on whole characters. It should never happen that the end of a chunk contains the beginning of a unicode character and the beginning of the next chunk contains the remaining part! This can be easily implemented in .NET by using `StreamReader` to process the CSV file on the client.

The speed if the import is highly affected by the size of the chunks. The choice of the chuck size for import depends only on the invoking system, thus fine tuning can be performed by the consumer system.

The steps of CSV export are very similar:

- Start a new CSV export session by calling BeginChunkedCSVExport.
- Export the chunks of a CSV document by calling `GetNextExportChunk` in turns.
- Call `EndChunkedExport` at the end to close the CSV import session.

The memoQ CVS format is supported both for CSV import and export. The import file must have header information, otherwise the content cannot be imported. (The CSV files exported using TB services contain this header.)

MultiTerm export is also performed in multiple chunks for the same reason. However, only the XML content file is large in size, thus the two other descriptor files, the XDL and XDT files are returned as string parameters. The XML file is returned in chunks, like the CSV file, but both the XDL and the XDT files are returned with the first call to the `BeginChunkedMultiTermExport` function, as out string parameters.

The steps of MultiTerm export are:

- Start a new MultiTerm export session by calling `BeginChunkedMultiTermExport`.
- Write the two out string parameters returned by the previous function to XDL and XDT files.
- Export the chunks of the XML document by calling `GetNextExportChunk` in turns.
- Call `EndChunkedExport` at the end to close the import session.

3.4 LiveDocs API

3.4.1 Overview

The LiveDocs API provides the following functionality:

- Provide the list of corpora published by the memoQ Server
- Provide information about a specific corpus
- Supports the creation and publishing of a corpus
- Allows updating the properties of a corpus
- Supports the deletion of a corpus

- Allows aligning two documents structurally and based on segmentation, and exporting segment pairs as TMX

3.4.2 Brief description of the interface

The LiveDocs related interfaces and entity classes of the memoQ Server WS API are the following:

ResourceInfo: Base class for the `TMInfo` and `TBInfo` and `CorpusInfo` classes encapsulating common information for a TM/TB/Corpus, such as its guid, name and description.

CorpusInfo: Encapsulates the description of a corpus. This class is derived from `ResourceInfo`.

DocumentForAlignment: Encapsulates one of the documents used in the structural alignment. It contains the language of the document and its import settings.

AlignmentOptions: Encapsulates the options of the structural alignment.

AlignmentResultInfo: Contains the description of the result of the alignment process. This class is derived from `ResultInfo`.

ILiveDocsService This interface has operations for listing corpora published by the server. It supports the creation and the deletion of a corpora.

3.4.3 Detailed description of the interface

Below is the C# code for the interfaces/classes with their description. The source is annotated by .NET Framework WCF attributes (ServiceContract, OperationContract, DataContract, DataMember), which can be ignored.

ILiveDocsService interface

```
/// <summary>
/// This interface has operations for listing managing corpora on
/// the memoQ Server.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface ILiveDocsService
{
    /// <summary>
    /// Creates a new corpus with the parameters specified by the info operation parameter.
    /// </summary>
    /// <param name="info">The parameters of the corpus to be created. The Guid and the
    /// Languages members of the info parameter are ignored.</param>
    /// <returns>The Guid of the newly created corpora.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    Guid CreateAndPublish(CorpusInfo info);
    /// <summary>
    /// Returns the list of corpora published by the memoQ Server.
    /// </summary>
    /// <param name="targetLang">Can be used to filter the returned list of corpora
    /// based on the languages of the corpora. It is allowed to provide null.</param>
    /// <returns>An array of CorporaInfo objects each describing one corpora.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    CorpusInfo[] ListCorpora(string[] languages);
    /// <summary>
    /// Gets information about the specified corpus.
    /// </summary>
    /// <param name="corpusGuid">The guid of the corpus.</param>
    /// <returns>
```

```
    /// The CorpusInfo object describing the corpus.
    /// </returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    CorpusInfo GetCorpusInfo(Guid corpusGuid);
    /// <summary>
    /// Updates the properties of a corpus.
    /// </summary>
    /// <param name="updateInfo">The new properties of the corpus.
    /// The Guid member identifies the corpus to be updated.</param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void UpdateProperties(CorpusUpdateInfo updateInfo);
    /// <summary>
    /// Deletes a corpus.
    /// </summary>
    /// <param name="tmGuid">The guid of the corpus to be deleted.
    /// </param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void DeleteCorpus(Guid corpusGuid);
    /// <summary>
    /// Performs structural alignment of two documents and exports the aligned
    /// segment pairs as a TMX.
    /// First the documents are imported using the specified configurations,
    /// then the alignment is performed, and the resulting matched segment pairs
    /// are exported into a TMX (optionally zipped).
    /// The two files must be uploaded using IFileManagerServive before invoking
    /// this operation. If the result is success, the resulting file can also be
    /// downloaded using IFileManagerService.
    /// </summary>
    /// <param name="source">The source document and its import configuration.
    /// </param>
    /// <param name="target">The target document and its import configuration.</param>
    /// <param name="options">Options of the alignment operation.</param>
    /// <returns></returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    AlignmentResultInfo AlignDocumentsGetTmx(DocumentForAlignment source,
        DocumentForAlignment target, AlignmentOptions options);
}
```

ResourceInfo class

See chapter *3.2.3 Detailed description of the interface*.

HeavyResourceInfo class

See chapter *3.2.3 Detailed description of the interface*.

CorpusInfo class

```
/// <summary>
/// Encapsulates the description of a corpus. This class is derived from ResourceInfo.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class CorpusInfo : HeavyResourceInfo
{
    /// <summary>
    /// The three+(two) letter codes of the languages of the corpus.
    /// (such as fre, eng, eng-US).
    /// </summary>
    [DataMember]
    public string[] LanguageCodes;
}
```

ResourceUpdateInfo class

See chapter *3.2.3 Detailed description of the interface*.

HeavyResourceUpdateInfo class

See chapter *3.2.3 Detailed description of the interface*.

CorpusUpdateInfo class

```
/// <summary>
/// Encapsulates the description of a corpus. This class is derived
/// from ResourceUpdateInfo.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class CorpusUpdateInfo : HeavyResourceUpdateInfo
{ }
```

To get the list of corpora published by the server call the `ListCorpora` operation of the `LiveDocsService` service. To get an unfiltered list of the corpora, submit `null` parameter to the function.

To create and publish a corpus call the `CreateAndPublish` operation of the `LiveDocsService` service. The `Guid` and the `Languages` parameters of the info parameter are ignored.

To get information about a specific corpus call the `GetCorpusInfo` operation of the `LiveDocsService` service.

To update the properties of an existing corpus call the `UpdateProperties` operation of the `CorpusService` service.

To delete a corpus call the `DeleteCorpus` operation of the `LiveDocsService` service.

DocumentForAlignment class

```
/// <summary>
/// Specifies one of the documents and its import settings for alignment.
/// </summary>
public class DocumentForAlignment
{
    /// <summary>
    /// The guid of the file to be imported. The file has to be uploaded before calling
    /// invoking the alignment. IFileManagerService.BeginChunkedFileUpload and related
    /// operations. This guid is returned by IFileManagerService.BeginChunkedFileUpload.
    /// </summary>
    [DataMember]
    public Guid FileId;
   /// <summary>
    /// The language of the document.
    /// </summary>
    [DataMember]
    public string Language;
    /// <summary>
    /// The identifier of the Filter Configuration resource published by memoQ Server
    /// which spcifies the filter and the configuration for importing the document.
    /// This setting also determines the filter (txt, doc, etc.) to be used. Either
    /// this, or ImportConfigurationResourceAsString must specify the configuration.
    /// </summary>
    [DataMember]
    public Guid? ImportConfigurationResourceId;
    /// <summary>
    /// The settings of the import in memoQ import filter XML format or memoQ
    /// resource former. This setting also determines the filter (txt, doc, etc.)
    /// to be used. Either this, or ImportConfigurationResourceId must specify
    /// the configuration.
    /// </summary>
    [DataMember]
    public string ImportConfigurationResourceAsString;
    /// <summary>
    /// The identifier of the Segmentation rule resource published by memoQ Server
    /// which spcifies the segmentation rule for importing the document. If the value
    /// is not specified (null), the document will not be segmented. If Guid.Empty
    /// is specified, the language's default segmentation rule specified by the
```

```
    /// server will be used. If the value is a valid Guid, the segmentation rule
    /// identified by the Guid is used. The default is null/missing value, which
    /// means no segmentation.
    /// </summary>
    [DataMember]
    public Guid? SegmentationRuleResourceId;
}
```

To create a TMX from two documents based on alignment, two files have to be uploaded using the `IFileManagerService`. The identifier supplied by the `IFileManagerService` after a successful upload can be used to create an instance of the `DocumentForAlignment` class. A string typed three (+two) letter language code specifies the language of the document. This have to be a supported code by memoQ Server and the two documents has to have different languages. The import configuration of the document is specified either using the unique identifier if a filter configuration resource published by the server (`ImportConfigurationResourceId`) or `ImportConfigurationResourceAsString` contains the complete filter configuration resource as exported by memoQ. The documents can be imported with or without segmentation rules. If no segmentation rules are specified (null or missing value), the documents will be segmented on structural boundaries only. To use language specific segmentation rules, a resource id must be specified, which identifies the Segmentation rules data in memoQ Server; or specifying an all zero Guid (not null!) the server's language specific default segmentation rules are used.

AlignmentOptions class

```
/// <summary>
/// Specifies the options of the alignment.
/// </summary>
public class AlignmentOptions
{
    /// <summary>
    /// True to zip the resulting TMX file; false to keep the resulting
    /// TMX as-is.
    /// </summary>
    [DataMember]
    public bool ZipTmx;
    /// <summary>
    /// The user name to write into the TMX for each TU as the creator user.
    /// </summary>
    [DataMember]
    public string TmxUserName;
    /// <summary>
    /// Switch to turn structural alignment on or off. If value is not specified,
    /// structural aligment is used.
    /// </summary>
    [DataMember]
    public bool? PerformStructuralAlignment;
}
```

This class contains the options of the alignment. The `TmxUserName` field must be specified. Turn enable or disable structural alignment, set the `PerformStructuralAlignment` field. The default of this field, if unspecified/missing, is true.

AlignmentResourceInfo class

```
/// <summary>
/// The result object describing an aligment process.
/// </summary>
public class AlignmentResultInfo : ResultInfo
{
    /// <summary>
    /// If the process completed successfully it contains the guid of the resulting
    /// file which can be downloaded using IFileManagerService. Unspecified if the
    /// result is not success.
```

```
    /// </summary>
    [DataMember]
    public Guid TmxFileGuid;
    /// <summary>
    /// The number of unaligned segments in the source document. Unspecified if the
    /// result is not success.
    /// </summary>
    [DataMember]
    public int UnalignedSegmentsInSource;
    /// <summary>
    /// The number of unaligned segments in the target document. Unspecified if the
    /// result is not success.
    /// </summary>
    [DataMember]
    public int UnalignedSegmentsInTarget;
    /// <summary>
    /// The number of aligned segments (the number of entries in the TMX). Unspecified
    /// if the result is not success.
    /// </summary>
    [DataMember]
    public int AlignedSegments;
}
```

After the alignment an instance of this class encapsulates information about the alignment. The `TMXFileGuid` is an identifier to the created TMX file which can be downloaded using the `IFileManagerService`. The other fields contain quantitative data about the alignment.

3.5 Light Resource API

3.5.1 Overview

TM and TB are called heavy resources. Opposed to them, the following entities belong to light resources: auto translatables, non translatables, autocorrect lists, ignore lists, segmentation rules, TM settings, filter configurations, keyboard shortcuts, export path rules, QA settings, stopword lists, LQA settings, LiveDocs settings, web search settings, font substitution settings and project templates.

The light resource API provides the following functionality:

- Creating, publishing, cloning, deleting and listing light resources
- Importing from and exporting to memoQ resource format (different for each resource type)

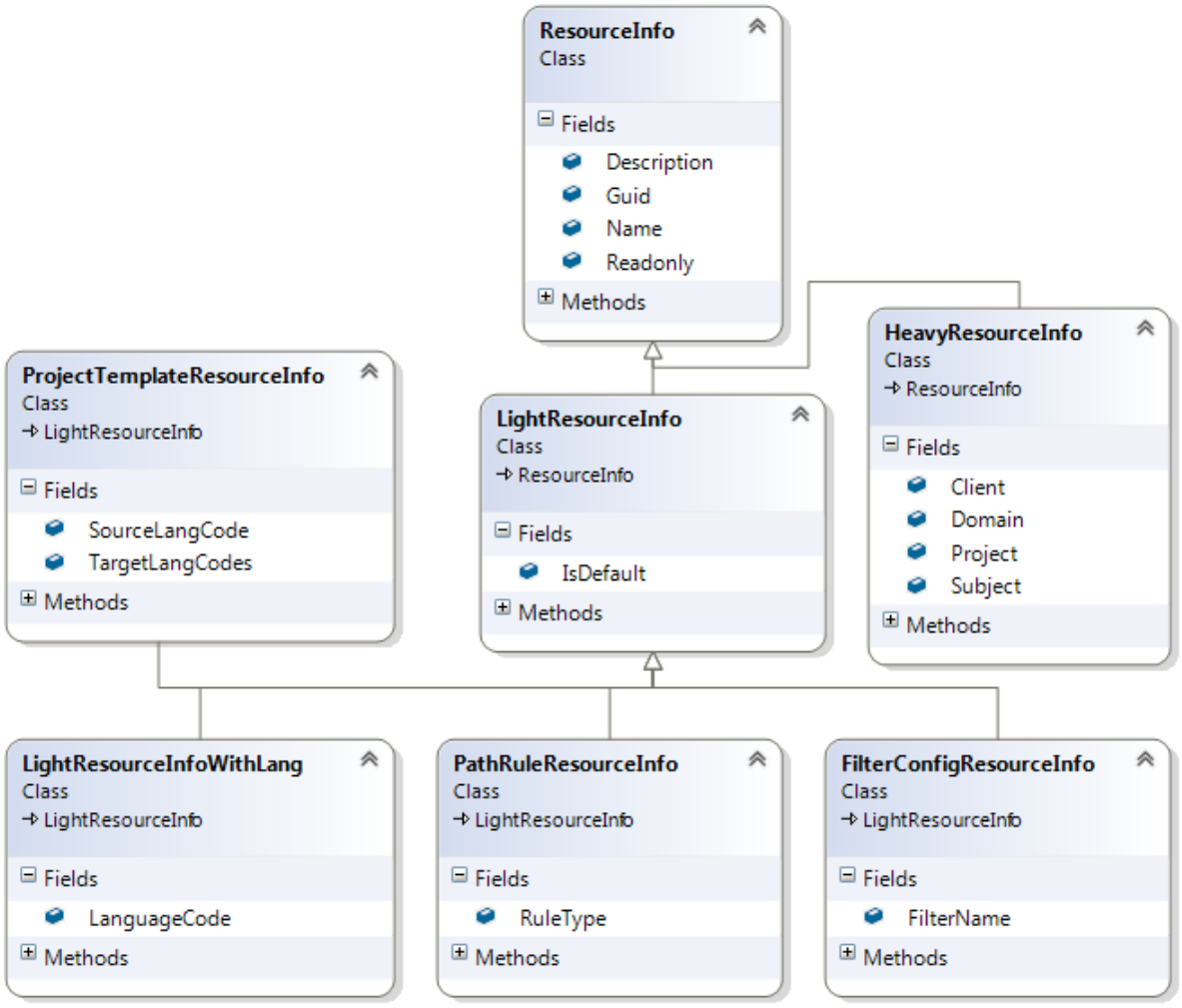
3.5.2 Brief description of the interface

The security related interfaces and entity classes of the memoQ Server WS API are the following:

ResourceType enum: Enumeration representing different light resource types. Please read the code comments for this enum type below for the description of different resource types.

LightResourceInfo: Encapsulates the description of a light resource. The some resource types are directly described by objects of this class (e.g. AutoTrans), other ones (e.g. IgnoreList) by objects of derived classes. To find out exactly which class is used for a resource type see the "Described by class" section of the documentation of the appropriate ResourceType enumeration member. See enum ResourceType for more information.

The next class diagram shows LightResourceInfo and its parent/derived classes:



LightResourceInfoWithLang: Encapsulates the description of a language dependent light resource.

PathRuleResourceInfo: Encapsulates the description of an export path rule light resource.

PathRuleType enum: Represents the two different types of export path rule.

FilterConfigResourceInfo: Encapsulates the description of a filter configuration light resource.

FilterNames: Defines the names of filters (document converters) supported by memoQ.

ProjectTemplateResourceInfo: Encapsulates the description of a project template light resource.

IResourceService (possibly exposed as `ResourceService` proxy class): This interface has operations for managing light resources.

3.5.3 Detailed description of the interface

Below is the C# code for the interfaces/classes with their description. The source is annotated by .NET Framework WCF attributes (ServiceContract, OperationContract, DataContract, DataMember), which can be ignored.

IResourceService interface

```
/// <summary>
/// This interface has operations for light resource management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IResourceService
{
    /// <summary>
    /// Creates a new resource with the parameters specified by the resourceInfo parameter.
    /// FilterConfig resources can not be created this way, use ImportNewAndPublish
    /// instead to create them.
    /// </summary>
    /// <param name="resourceType">The type of the resource to be created.</param>
    /// <param name="resourceInfo">The parameters of the resource to be created.
    /// The Guid and IsDefault members of the resourceInfo parameter are ignored.
    /// To find out exactly which LightResourceInfo derived class is to be used here for a
    /// resource type, see the "Described by class" section of the documentation
    /// of the appropriate "ResourceType" enum member (e.g.ResourceInfoWithLang
    /// for SegRules).
    /// </param>
    /// <returns>The Guid of the newly created resource.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    Guid CreateAndPublish(ResourceType resourceType, LightResourceInfo resourceInfo);
    /// <summary>
    /// Gets information about the specified resource.
    /// </summary>
    /// <param name="resourceType">The type of the resource.</param>
    /// <param name="resourceGuid">The guid of the resource.</param>
    /// <returns>
    /// The LightResourceInfo (or derived object, depending the type of the resource)
    /// object describing the resource.
    /// To find out exactly which derived class is returned here for a
    /// resource type, see the "Described by class" section of the documentation of
    /// the appropriate "ResourceType" enum member (e.g.ResourceInfoWithLang for SegRules).
    /// </returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    LightResourceInfo GetResourceInfo(ResourceType resourceType, Guid resourceGuid);
    /// <summary>
    /// Deletes a resource.
    /// If the resource is readonly, an exception is thrown.
    /// </summary>
    /// <param name="resourceType">The type of the resource.</param>
    /// <param name="tmGuid">The guid of the resource to be deleted.
    /// </param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void DeleteResource(ResourceType resourceType, Guid resourceGuid);
    /// <summary>
    /// Creates a clone of an existing resource. The newly created resource will have the
    /// same data as the original one, but the properties (name, description, etc.) will
    /// be as defined by the resourceInfo parameter.
    /// </summary>
    /// <param name="resourceType">The type of the resource to be cloned.</param>
    /// <param name="resourceInfo">
    /// The parameters of the resource to be created.
    /// - The Guid member identifies the resource to be cloned.
    /// - The IsDefault member is ignored.
    /// - For all resource types class LightResourceInfo can be used directly. As an
    ///   alternative, the LightResourceInfo derived class belonging to the specific
    ///   resource type (as defined by the "Described by class" section of the
    ///   documentation of the appropriate "ResourceType" enum member) can also be used.
    ///   However, in this latter case the RuleType for PathRule, and FilterName for
    ///   FilterConfig resources can not be changed (these are ignored). For language
    ///   dependent resources (e.g. SegRules) ResourceInfoWithLang can be used, and
    ///   the resource can be cloned to a different language. One way to keep the
    ///   original language is to use null as the language code.
    /// </param>
    /// <returns>The Guid of the newly created resource.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    Guid CloneAndPublish(ResourceType resourceType, LightResourceInfo resourceInfo);
    /// <summary>
    /// Updates the properties of a resource.
```

```
    /// </summary>
    /// <param name="resourceType">The type of the resource to be created.</param>
    /// <param name="resourceInfo"> The new properties of the resource.
    /// - The Guid member identifies the resource to be updated.
    /// - The IsDefault member is ignored.
    /// - For read only resources (for which the Readonly proprty is true) only the
    ///   Readonly property can be changed.
    /// - For all resource types class LightResourceInfo can be used directly. As an
    ///   alternative, the LightResourceInfo derived class belonging to the specific
    ///   resource type (as defined by the "Described by class" section of the
    ///   documentation of the appropriate "ResourceType" enum member) can also be used.
    ///   However, in this latter case the following properties can not be changed
    ///   (these are ignored):
    ///   - LanguageCode for langauge dependent resources (can be null)
    ///   - RuleType for PathRule resources
    ///   - FilterName for FilterConfig resources (can be null)
    /// </param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void UpdateResourceProperties(ResourceType resourceType, LightResourceInfo resourceInfo);
    /// <summary>
    /// Returns the list of the light resources available on the server.
    /// </summary>
    /// <param name="resourceType">The type of the resource to be listed.</param>
    /// <returns>
    /// An array of LightResourceInfo (or derived, depending the type of the resource) objects
    /// is returned.
    /// To find out exactly which derived class is returned here for a resource type, see
    /// the "Described by class" section of the documentation of the appropriate
    /// "ResourceType" enum member (e.g.ResourceInfoWithLang for SegRules).
    /// </returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    LightResourceInfo[] ListResources(ResourceType resourceType);
    /// <summary>
    /// Creates a new resource and imports data from the specified file into it.
    /// Certain properties of the new resource are taken from the resourceInfo
    /// parameter, while others are taken from the header of the import file and
    /// can not be changed. The Name, Description and IsReadonly is taken from
    /// the resourceInfo parameter. For other rules see documentation for the
    /// resourceInfo parameter below.
    /// </summary>
    /// <param name="resourceType">The type of the resource to be newly imported.
    /// </param>
    /// <param name="resourceInfo">
    /// The properties of the resource to be created.
    /// - The IsDefault and Guid members of the resourceInfo parameter are ignored.
    /// - For all resource types class LightResourceInfo can be used directly. As an
    ///   alternative, the LightResourceInfo derived class belonging to the specific
    ///   resource type (as defined by the "Described by class" section of the
    ///   documentation of the appropriate "ResourceType" enum member) can also be used.
    ///   However, in this latter case the following properties can not be changed
    ///   (are taken from the header of the import file):
    ///   - LanguageCode for langauge dependent resources (can be null)
    ///   - RuleType for PathRule resources
    ///   - FilterName for FilterConfig resources (can be null)
    /// </param>
    /// <returns>The Guid of the newly imported resource. Alhough the header of
    /// the import file contains a guid, it is ignored, and a new guid is assigned.
    /// </returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    Guid ImportNewAndPublish(ResourceType resourceType, Guid fileGuid,
        LightResourceInfo resourceInfo);
    /// <summary>
    /// Exports the resource to a file. The format of the file is the same as used
    /// by memoQ import/export for the specific resource type.
    /// </summary>
    /// <param name="resourceType">The type of the resource.</param>
    /// <param name="resourceGuid">The guid of the resource.</param>
    /// <returns>The guid of the resulting file of the operation.
    /// Can be dowloaded using FileManagerService.
    /// </returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    Guid ExportResource(ResourceType resourceType, Guid resourceGuid);
}
```

ResourceType enumeration

```
/// <summary>
/// Enumeration representing different light resource types.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public enum ResourceType
{
    /// <summary>
    /// Auto translatable resource.
    /// Language neutral.
    /// 0..N can be assigned to each target language of a server project
    /// (independently).
    /// Described  by class: LightResourceInfo
    /// </summary>
    [EnumMember] AutoTrans,
    /// <summary>
    /// Non-translatable resource.
    /// Language neutral.
    /// 0..N can be assigned to a server project (shared for all target languages
    /// of the project). One of them is set as primary.
    /// Described  by class: LightResourceInfo
    /// </summary>
    [EnumMember] NonTrans,
    /// <summary>
    /// Auto-correct resource.
    /// Language dependent (but the language can be Neutral).
    /// Can not be assigned to server projects.
    /// Described by class: ResourceInfoWithLang
    /// </summary>
    [EnumMember] AutoCorrect,
    /// <summary>
    /// Ignore list resource.
    /// Language dependent.
    /// 0..N (with exact maching language) can be assigned independently to
    /// each target language of a server project, with one primary per target
    /// language (if there is at least one assigned to the specific target
    /// language).
    /// Described by class: ResourceInfoWithLang
    /// </summary>
    [EnumMember] IgnoreLists,
    /// <summary>
    /// Segmentation rule resource.
    /// Language dependent.
    /// Exactly one has to be assigned to a project, which defines the segmentation
    /// rules for the source language of the project. The language of the resource
    /// has to exact match the source language of the project.
    /// Described by class: ResourceInfoWithLang
    /// </summary>
    [EnumMember] SegRules,
    /// <summary>
    /// TM settings resource.
    /// Language independent.
    /// Exactly one has to be assigned to a project, which defines the default
    /// TM settings for the project. Optionaly one can be assigned for each TM
    /// assigned to the project that overrides the project default for the
    /// specific TM. When a server project is created, the default TM setting
    /// resource is automatically assigned to the project.
    /// Described by class: LightResourceInfo
    /// </summary>
    [EnumMember] TMSettings,
    /// <summary>
    /// Filter configuration resource.
    /// Language independent, can not be assigned to projects.
    /// Described by class: FilterConfigResourceInfo
    /// </summary>
    [EnumMember] FilterConfigs,
    /// <summary>
```

```
/// Keyboard shortcut resource.
/// Language independent, can not be assigned to projects.
/// Described by class: LightResourceInfo
/// </summary>
[EnumMember] KeyboardShortcuts,
/// <summary>
/// Export path rules resource.
/// Language independent.
/// Exactly two have to be assigned to a project: one defining file export
/// path rule, and another one defining the folder export path rule.
/// When a server project is created, the default file and default folder
/// PathRule resources are automatically assigned to the project.
/// Described by class: PathRuleResourceInfo
/// </summary>
[EnumMember] PathRules,
/// <summary>
/// QA settings resource.
/// Language independent.
/// Exactly one has to be assigned to each target language of the project.
/// When a server project is created, the default QA setting resource is
/// automatically assigned to each target language of the project.
/// Described by class: LightResourceInfo
/// </summary>
[EnumMember] QASettings,
/// <summary>
/// Stopword list resource.
/// Language dependent.
/// Can not be assigned to server projects.
/// Described by class: ResourceInfoWithLang
/// </summary>
[EnumMember] Stopwords
/// <summary>
/// LQA model resource.
/// Language independent.
/// At most one has to be assigned to a the project.
/// Described by class: LightResourceInfo
/// </summary>
[EnumMember] LQA,
/// <summary>
/// LiveDocs corpora settings resource.
/// Language independent.
/// Exactly one has to be assigned to a project, which defines the default
/// LiveDocs settings for the project. Optionaly one can be assigned for each
/// corpus assigned to the project that overrides the project default for the
/// specific corpus. When a server project is created, the default corpus
/// setting resource is automatically assigned to the project.
/// Described by class: LightResourceInfo
/// </summary>
[EnumMember] LiveDocsSettings,
/// <summary>
/// WebSearch settings resource.
/// Language independent.
/// Cannot be assigned to server projects.
/// Described by class: LightResourceInfo
/// </summary>
[EnumMember] WebSearchSettings
/// <summary>
/// Font substitution settings resource.
/// Language independent.
/// At most one has to be assigned to a the project.
/// Described by class: LightResourceInfo
/// </summary>
[EnumMember] FontSubstitution,
/// <summary>
/// Keyboard shortcut witch context resource.
/// Language independent, can not be assigned to projects.
/// Described by class: LightResourceInfo
/// </summary>
```

```
    [EnumMember]
    KeyboardShortcuts2,
    /// <summary>
    /// Project template settings resource.
    /// Language independent.
    /// </summary>
    [EnumMember]
    ProjectTemplate
}
```

LightResourceInfo class

```
/// <summary>
/// Encapsulates the description of a light resource. This class is derived
/// from ResourceInfo. Some resource types are directly described by
/// objects of this class (e.g. AutoTrans), other ones (e.g. IgnoreList)
/// by objects of derived classes. To find out exactly which class is used
/// for a resource type see the "Described by class" section of the
/// documentation of the appropriate ResourceType enumeration member.
/// See enum ResourceType for more information.
/// </summary>
[KnownType(typeof(LightResourceInfoWithLang))]
[KnownType(typeof(PathRuleResourceInfo))]
[KnownType(typeof(FilterConfigResourceInfo))]
[KnownType(typeof(ProjectTemplateResourceInfo))]
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class LightResourceInfo : ResourceInfo
{
    /// <summary>
    /// Indicates if this is a default resource. Default resources are deployed
    /// by the installer and can not be modified/deleted.
    /// </summary>
    [DataMember]
    public bool IsDefault;
}
```

LightResourceInfoWithLang class

```
/// <summary>
/// Encapsulates the description of a language dependent light resource.
/// This class is derived from LightResourceInfo.
/// See class LightResourceInfo and enum ResourceType for more information.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class LightResourceInfoWithLang : LightResourceInfo
{
    /// <summary>
    /// The three+(two) letter code of the language of the resource.
    /// Can not be changed after the resource has been created.
    /// </summary>
    [DataMember]
    public string LanguageCode;
}
```

PathRuleType enumeration

```
/// <summary>
/// Represents the two different types of export path rule.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public enum PathRuleType
{
    [EnumMember] File,
    [EnumMember] Folder
}
```

PathRuleResourceInfo class

```
/// <summary>
/// Encapsulates the description of an export path rule light resource.
/// This class is derived from LightResourceInfo.
/// See class LightResourceInfo and enum ResourceType for more information.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PathRuleResourceInfo : LightResourceInfo
{
    /// <summary>
    /// Indicates if this object represents a file or folder export path
    /// rule.
    /// Can not be changed after the resource has been created.
    /// </summary>
    [DataMember]
    public PathRuleType RuleType;
}
```

FilterConfigResourceInfo class

```
/// <summary>
/// Encapsulates the description of a filter configuration light resource.
/// This class is derived from LightResourceInfo.
/// See class LightResourceInfo and enum ResourceType for more information.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class FilterConfigResourceInfo : LightResourceInfo
{
    /// <summary>
    /// Non-localized name of the filter this filter config belongs to
    /// (the filters class name e.g.MemoQ.DocConverters.XML.XMLConverter).
    /// Can not be changed after the resource has been created.
    /// </summary>
    [DataMember]
    public string FilterName;
}
```

ProjectTemplateResourceInfo class

```
/// <summary>
/// Encapsulates the description of a project template light resource.
/// This class is derived from LightResourceInfo.
/// See class LightResourceInfo and enum ResourceType for more information.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ProjectTemplateResourceInfo : LightResourceInfo
{
    /// <summary>
    /// The source language stored in the template; can be null.
    /// Can not be changed.
    /// </summary>
    [DataMember]
    public string SourceLangCode;
    /// <summary>
    /// The target languages stored in the template; can be null.
    /// Can not be changed.
    /// </summary>
    [DataMember]
    public string[] TargetLangCodes;
}
```

3.6 Security API

3.6.1 Overview

The security API provides the following functionality:

- Manage memoQ Server users:
  - Create new users
  - Update existing users
  - Delete users
- Manage memoQ Server groups:
  - Create new groups
  - Update existing groups
  - Delete groups
- Manage user-group memberships
- Manage (set and list) permissions for translation memories and term bases
- Login and logout functions. Deprecated, will be eliminated in 8.0.

3.6.2 Brief description of the interface

The security related interfaces and entity classes of the memoQ Server WS API are the following:

UserInfo: Represents a memoQ Server user.

UserPackageWorkflowType: Represents the package workflow type of the user.

GroupInfo: Represents a memoQ Server group.

ObjectPermission: Represents a permission for a memoQ Server object (TM/TB), such as performing lookup on translation memory X for user Y (or for group Z).

ISecurityService (possibly exposed as `SecurityService` proxy class): This interface has operations for managing users, groups and permissions.

3.6.3 Detailed description of the interface

Below is the C# code for the interfaces/classes with their description. The source is annotated by .NET Framework WCF attributes (ServiceContract, OperationContract, DataContract, DataMember), which can be ignored.

ISecurityService interface

```
/// <summary>
/// Provides operations for:
///  - User management
///  - Group management
///  - Permission management.
/// </summary>
/// <remarks>
/// ---------------
/// User management
```

```
/// ---------------
/// Provides operations for cretating new users, updating users,
/// deleting users and set/get user-group memberships.
/// Each user is identified by a GUID.
/// There is one built in user that can not be deleted/disabled:
/// admin, Administrator, "00000000-0000-0000-0001-000000000001".
///
/// ----------------
/// Group management
/// ----------------
/// Provides operations for cretating new groups, updating groups,
/// deleting groups and set/get group-user memberships (same as
/// user-group membership, but from the opposite perspective).
/// Each group is identified by a GUID.
/// There are some built in users that can not be deleted:
/// Administrators, "00000000-0000-0000-0000-000000000001"
/// ProjectManagers, "00000000-0000-0000-0000-000000000002"
/// Translators, "00000000-0000-0000-0000-000000000003"
/// Terminologists, "00000000-0000-0000-0000-000000000004"
/// Everyone, "00000000-0000-0000-0000-100000000000"
///
/// ---------------------
/// Permission management
/// ---------------------
/// Provides operations for setting and getting permissions for server objects.
/// Server objects can be of the following types: TM, TB, Corpus,
/// server project and light resources. Currently only TM, TB, Corpus and light
/// resources are supported.
/// Each permission assignment has the following attributes:
/// - ObjectGuid: The guid of the object (TM/TB/Corpus/light resource) the
///   permission is related to.
/// - GuidOfUserOrGroup (or Sid): The GUID of the user or group the permisson is
///   related to. If a permission is assigned to a group, all users in that group
///   inherit the permission.
/// - PermissionId: an integer representing the permission. Valid values are the
///   following:
///   For TMs:
///     1: Lookup - Users can perform lookup on the TM
///     2: Update - Lookup plus users can add new entries and update existing ones.
///     1000: Admin - Update plus users can edit the TM and set permissions on it.
///   For TBs:
///     1: Lookup - Users can perform lookup on the TB
///     2: Update - Lookup plus users can add new entries and update existing ones.
///     3: Review - Modify plus users can perform review operations on a
///                 reviewable TB.
///     1000: Admin - Review plus users can edit the TB and set permissions on it.
///   For Corpora:
///     1: Lookup - Users can add corpus to porject and can lookup results in
///                 translation grid
///     2: MassLookup - Lookup + batch lookup for statistics and pre-translate
///     3: View - MassLookup + access to documents and open documents
///               for view only
///     4: Edit - View + edit documents
///     5: Approve - Edit + approve documents
///     1000: Admin - Approve + change permissions for the corpus, delete the
///                   corpus  remove the corpus from the server,add/remove
///                   documents from the corpus, change readonly attribute,
///                   change corpus properties, unpublish
///   For light resources:
///     1: Use - Users can use the resource, but can not change it.
///     2: Change - Use + users can change the resource (e.g its content and
///                 properties)
///     1000: Admin - Change + change permissions for the resource, delete, clone,
///                   import and export the resource.
/// ---------------------
/// Server Projects
/// ---------------------
/// It should be noted that special permissions apply (besides global permissions)
```

```
/// when a user is connected to a server project from MemoQ Client based on his/her
/// server project role. This topic will be covered in the Server Project
/// webservice interface specification.
/// </remarks>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface ISecurityService
{
    #region User management
    /// <summary>
    /// Returns the list of MemoQ Server users.
    /// </summary>
    /// <returns>The list of users.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    UserInfo[] ListUsers();
    /// <summary>
    /// Returns information about the user.
    /// </summary>
    /// <param name="userGuid">The guid of the user.</param>
    /// <returns>Information about the specific user.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    UserInfo GetUser(Guid userGuid);
    /// <summary>
    /// Creates a new user.
    /// </summary>
    /// <param name="userInfo">The parameters of the user to be created.
    /// The Guid member of the info parameter is ignored.</param>
    /// <returns>The Guid of the newly created user.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    Guid CreateUser(UserInfo userInfo);
    /// <summary>
    /// Updates an existing user.
    /// </summary>
    /// <param name="userInfo">A UserInfo object holding information about
    /// the user to be updated. Can not be null.</param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void UpdateUser(UserInfo userInfo);
    /// <summary>
    /// Deletes a specific user. The user is not deleted internally but is
    /// set to inactive so that it is no longer listed.
    /// </summary>
    /// <param name="userGuid">The GUID of the user to be deleted.</param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void DeleteUser(Guid userGuid);
    /// <summary>
    /// Returns the groups the user of member of.
    /// </summary>
    /// <param name="userGuid">The GUID of the user.</param>
    /// <returns>A list of GroupInfo objects, each representing a group
    /// the user is member of.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    GroupInfo[] ListGroupsOfUser(Guid userGuid);
    /// <summary>
    /// Sets the groups the user is to be member of.
    /// </summary>
    /// <param name="userGuid">The GUID of the user.</param>
    /// <param name="groupGuids">A list of GUIDs each representing a group
    /// the user is to be added to.
    /// Existing group assignments are deleted. Can not be null.</param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void SetGroupsOfUser(Guid userGuid, Guid[] groupGuids);
```

```
#endregion

#region Group management
/// <summary>
/// Returns the list of MemoQ Server groups.
/// </summary>
/// <returns>The list of groups.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
GroupInfo[] ListGroups();
/// <summary>
/// Returns the list of memoQ Server subvendor groups.
/// </summary>
/// <returns>The list of groups.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
GroupInfo[] ListSubvendorGroups();
/// <summary>
/// Returns information about a group.
/// </summary>
/// <param name="groupGuid">The GUID od the group.</param>
/// <returns>Information about the specific group.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
GroupInfo GetGroup(Guid groupGuid);
/// <summary>
/// Creates a new group.
/// </summary>
/// <param name="groupInfo">The parameters of the group to be created.
/// The Guid member of the info parameter is ignored.</param>
/// <returns>The Guid of the newly created group.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
Guid CreateGroup(GroupInfo groupInfo);
/// <summary>
/// Updates an existing group.
/// </summary>
/// <param name="groupInfo">A GroupInfo object holding information about
/// the group to be updated. Can not be null.</param>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
void UpdateGroup(GroupInfo groupInfo);
/// <summary>
/// Deletes a specific group. The group is not deleted internally but is
/// set to inactive so that it is no longer listed.
/// </summary>
/// <param name="groupGuid">The GUID of the group to be deleted.</param>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
void DeleteGroup(Guid groupGuid);
/// <summary>
/// Returns the users of the group.
/// </summary>
/// <param name="userGuid">The GUID of the group.</param>
/// <returns>A list of UserInfo objects, each representing a user
/// that is member of the group.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
UserInfo[] ListUsersOfGroup(Guid groupGuid);
/// <summary>
/// Sets the user members of a group.
/// </summary>
/// <param name="groupGuid">The GUID of the group.</param>
/// <param name="groupGuids">A list of GUIDs each representing a user
/// that are added to the group. Existing members are removed
/// from the group. Can not be null.</param>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
```

```
    void SetUsersOfGroup(Guid groupGuid, Guid[] userGuids);
    #endregion

    #region Permission management
    /// <summary>
    /// Sets the new (global) permissions on an object.
    /// Existing (global) permissions are deleted.
    /// </summary>
    /// <param name="objectGuid">The object (TM/TB/Corpus/light resource) for
    /// which the permissions are to be set.</param>
    /// <param name="permissions">The new permissions to be assigned.</param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void SetObjectPermissions(Guid objectGuid, ObjectPermission[] permissions);
    /// <summary>
    /// Returns the currently assigned (global) permissions on an
    /// object(TM/TB/Corpus/light resource).
    /// </summary>
    /// <param name="objectGuid">The object (TM/TB/Corpus/light resource) for
    /// which the permissions are to be returned.</param>
    /// <returns>The list of permissions for the object.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    ObjectPermission[] ListObjectPermissions(Guid objectGuid);

    #endregion
    #region Login/logout
    /// <summary>
    /// !!! DEPRECATED !!!
    /// Logs in a memoQ user and creates a session in the memoQ server.
    /// The session is identified by the string returned by this function.
    /// The session is used to authorize the user and check permissions for
    /// certain functions. The session identifier is expected as a parameter
    /// in the functions requring a valid session, hence the return value
    /// should be stored by the caller. The session has a 2 hour
    /// expiry window. A user can have multiple parallel sessions.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="passwordHash">The hash of the users password.</param>
    /// <returns>A session identifier which will be required by functions
    /// using the user session. Null is returned if login fails.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    string Login(string userName, string passwordHash);
    /// <summary>
    /// !!! DEPRECATED !!!
    /// Terminates a user session. After this function is called the session
    /// identifier will no longer be valid.
    /// </summary>
    /// <param name="sessionId">The idenfier of the session to terminate.
    /// </param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void Logout(string sessionId);
    #endregion

}
```

UserInfo class

```
/// <summary>
/// Represents a MemoQ Server user.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class UserInfo
{
    /// <summary>
    /// The Guid of the user.
```

```
/// </summary>
[DataMember]
public Guid UserGuid;
/// <summary>
/// The user name of the user.
/// </summary>
[DataMember]
public string UserName;
/// <summary>
/// The password hash of the user. See chapter password management.
/// </summary>
[DataMember]
public string Password;
/// <summary>
/// Indicates whether the user is enabled/disabled.
/// Disabled users can not log in to the MemoQ Server.
/// </summary>
[DataMember]
public bool IsDisabled;
/// <summary>
/// Full name of the user.
/// </summary>
[DataMember]
public string FullName;
/// <summary>
/// Address of the user.
/// </summary>
[DataMember]
public string Address;
/// <summary>
/// E-mail address of the user.
/// </summary>
[DataMember]
public string EmailAddress;
/// <summary>
/// Phone number of the user.
/// </summary>
[DataMember]
public string PhoneNumber;
/// <summary>
/// Mobile phone number of the user.
/// </summary>
[DataMember]
public string MobilePhoneNumber;
/// <summary>
/// Language pairs of the user. Null, if the user has no languages assigned.
/// '#' is used to separate the 3+(2) iso language codes in the language pairs.
/// ';' is used to separate language pairs.
/// E.g.: eng#hun;eng-US#ger
/// </summary>
[DataMember]
public string LanguagePairs;
/// <summary>
/// The user's package workflow type. Possible values:
/// - UserPackageWorkflowType.Online: the user can only work on "classic"
///   online projects.
/// - UserPackageWorkflowType.Both: the user can download packages if she
///   wants, but she can also use online projects.
/// - UserPackageWorkflowType.PackagesOnly: the user can only download
///   packages.
/// </summary>
[DataMember]
public UserPackageWorkflowType PackageWorkflowType;
/// <summary>
/// Indicates whether the user is subvendor manager. This is a read-only
/// field. The value of this field will be ignored when the entity class
/// is used as the input parameter of the CreateUser and UpdateUser
/// functions.
```

```
    /// </summary>
    [DataMember]
    public bool IsSubvendorManager;
}
```

UserPackageWorkflowType enumeration

```
/// <summary>
/// Enum representing the package workflow type of the user.
/// </summary>
public enum UserPackageWorkflowType
{
    /// <summary>
    /// The user can only work on "classic" online projects.
    /// </summary>
    Online = 0,
    /// <summary>
    /// The user can download packages if she wants, but she
    /// can also use online projects.
    /// </summary>
    Both = 1,
    /// <summary>
    /// The user can only download packages.
    /// </summary>
    PackagesOnly = 2
}
```

GroupInfo class

```
/// <summary>
/// Represents a MemoQ Server group.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class GroupInfo
{
    /// <summary>
    /// The Guid of the group.
    /// </summary>
    [DataMember]
    public Guid GroupGuid;
    /// <summary>
    /// The name of the group.
    /// </summary>
    [DataMember]
    public string GroupName;
    /// <summary>
    /// The descirption of the group.
    /// </summary>
    [DataMember]
    public string Description;
    /// <summary>
    /// Indicates whether the group is enabled/disabled.
    /// </summary>
    [DataMember]
    public bool IsDisabled;
    /// <summary>
    /// Indicates whether the group is subvendor group. Can be set
    /// only when the entity class is used as the input parameter
    /// of the CreateGroup function. The value of this field will
    /// be ignored when the entity class is used as the input
    /// parameter of the UpdateGroup function.
    /// </summary>
    [DataMember]
    public bool IsSubvendorGroup;
}
```

ObjectPermission class

```
/// <summary>
/// Represents a permission for a server object (TM/TB).
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class ObjectPermission
{
    /// <summary>
    /// The GUID of the user or group the permisson is related to.
    /// </summary>
    [DataMember]
    public Guid GuidOfUserOrGroup;
    /// <summary>
    /// The ID of the permission. See ISecurityService for
    /// explanation.
    /// </summary>
    [DataMember]
    public int PermissionId;
    /// <summary>
    /// The expiration date of the permission. Use values above or equal
    /// 2500.01.01 for permission assignments that never expire.
    /// </summary>
    [DataMember]
    public DateTime ExpirationDate;

}
```

3.6.4 Session management

DEPRECATED, Login, Logout and session management will be eliminated in 8.0.

Some operations use the identity of a memoQ user and complete authorization to check if the user has permissions to perform the actions. In order to validate the identity of the user, memoQ server and the API services employ sessions. A session is created through explicit login and is terminated in case of either explicit logout or session expiry.

The login and logout functions are exposed by the security service. The login function requires a user name (as used in memoQ, the `UserName` field of `UserInfo`) and a password hash. See Section 3.6.5 Password management for details about the password and hashing. This function returns a sting, which is the session identifier. This identifier is unique to this session and is valid only while the session is valid (i.e. until logout or session expiry). This session identifier must be specified as input parameters for the functions that require an existing user session. These functions verify the session identifier and report error if the session is invalid.

The caller must store the session identifier and reuse it for subsequent calls. It is not recommended to use a session for a single call and terminate the session afterwards. The session identifier should be cached by the caller and be reused while it is valid. The caller must also pay attention to the session expiry as detailed below.

The session must be terminated after it is no longer used. The logout function terminates the session explicitly. If the session is not terminated explicitly but is not used for more than 2 hours (no operations are initiated in the name of the user) the session is invalidated automatically. Please note that no notification is available when this happens. The caller will discovers an invalidated session the next time the session identifier is used. An error will be reported in this case.

Two types of faults are used in conjunction with sessions. An `InvalidSessionIdFault` signals that the specified session identifier is not valid or has expired, and an

`UnauthorizedAccessFault` is used to report that the user has no rights to perform the requested operation. (This is not the fault of the caller; this error should be reported to the user.)

3.6.5 Password management

The `Password` field of the `UserInfo` class represents the password hash of the user. For security reasons the original password is never sent on the network. Instead, a salted SHA1 algorithm based hash is to be used (with the `UserInfo` parameter of the `CreateUser` and `UpdateUser` operations). The pseudo code of the algorithm is the following:

```
UserInfo userInfo;
string password = <password entered by the user>;
const string salt = "fgad s d f sgds g  sdg gfdg";
byte[] bytesToHash = Encoding.UTF8.GetBytes(password + salt);
HashAlgorithm algorithm = SHA1.Create();
byte[] hash = algorithm.ComputeHash(bytesToHash);
userInfo.Password = ByteArrayToHexString(hash);
```

The `ByteArrayToHexString` is a function that should create a string representation of a byte array: e.g. "0AFF0B" for {10, 255, 11}.

In .NET C# there is a simple way to create the hash:

```
const string salt = "fgad s d f sgds g  sdg gfdg";

public static string HashPassword(string password)
{
    return System.Web.Security.
      FormsAuthentication.HashPasswordForStoringInConfigFile(password  +
      salt, "sha1");
}

void CreateUser(..., string passwordFromUser)
{
      UserInfo userInfo = new UserInfo();
      ...
      userInfo.Password = HashPassword(passwordFromUser);
      ...
}
```

A demo implementation of the ByteArrayToHexString function in C#:

```
public string ByteArrayToHexString(byte[] b)
{
    StringBuilder sb = new StringBuilder(b.Length * 2);
    for (int i = 0; i < b.Length; i++)
    {
        int v = b[i] & 0xff;
        sb.Append(v.ToString("X2"));
    }
    return sb.ToString().ToUpper();
}
```

3.7 Server projects API

3.7.1 Overview

The server project API provides the following functionality:

Getting global information:

- Getting the version of the memoQ Server API (that is the version of the memoQ Server);

Managing server projects:

- Creating a new server project with Live Documents or Desktop Documents;
- Updating the header information for an existing server project (such as description, deadline, etc.);
- Retrieving the list of server projects available on the server, optionally with filtering;
- Closing, unclosing and deleting server projects (close/unclose was called as archive/unarchive in previous versions);
- Assigning translation memories to a server project;
- Retrieving the list of translation memories assigned to a server project
- Assigning corpora to a server project
- Retrieving the list of corpora assigned to a server project
- Assigning term bases to a server project;
- Retrieving the list of term bases assigned to a server project;
- Assigning light resources to a server project
- Retrieving the list of light resources assigned to a server project;
- Assigning users to a server project with their roles specified;
- Retrieving the list of users assigned to a server project;
- Retrieving the list of translation documents imported to a server project;
- Delete a translation document from the server
- Assigning users to a translation document of a server project with their document assignment role;
- Manually changing the workflow status of documents;

Translation document import/export

- Importing documents in primary format, such as plain text, XML, RTF, DOCX, etc.
- Importing document from a bilingual file: memoQ Bilingual (MDB) and XLIFF formats are supported (we plan to add two column RTF support shortly).
- Updating a document that is already part of the project from a bilingual file. memoQ Bilingual (MDB) and XLIFF formats are supported.
- Exporting a translation document to its primary (original) format, such as plain text, XML, RTF, DOCX, etc.

- Exporting a translation document to a bilingual format: memoQ Bilingual (MDB), XLIFF and Bilingual RTF formats are supported.
- Reimporting translation documents and replacing existing documents with new versions of the document.
- The import of the old Office file formats are not supported: doc, xls and ppt.

Statistics

- Get statistics on all or specific documents of the server project;

Pre-translation

- Pre-translate on all or specific documents of the server project;
- Pre-translation using Asia Online services.

Confirm and update

- Confirm and update on all or specific documents of the server project.

X-translate

- X-translate on specific documents of the server project.

Document history

- Get the document history related to all or specific documents of the server project.

QA

- Run QA checks and get a report of QA errors and warnings in a server project.

Server project operations involve the transfer of large files (translation documents and bilinguals). Therefore, a unified and reliable transfer method has been introduced, that can be used throughout the API, involving chunked transfer and an option to handle compressed data. File management services are covered in details in chapter 0.

3.7.2 Draft interface

In the following chapters source code in C# language will be provided for the service interfaces and entity classes with their description. The source is annotated by .NET Framework WCF attributes (ServiceContract, OperationContract, DataContract, DataMember), which can be ignored.

All server project related operations are accessible via the **IServerProjectService** interface (possibly exposed as **ServerProjectService** proxy class):

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    // Operations are defined here, and will be described in the
    // following chapters

}
```

The operations supported by the **IServerProjectService** interface with the entity classes appearing as function parameters and return values will be categorized and described in details in the following chapters.

3.7.3 Getting global information

The GetApiVersion operation returns the current version of the API (that is the version of the memoQ Server).

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    /// <summary>
    /// Gets the version of the API (that is the version of the MemoQ Server)
    /// in the following format (an example): 5.0.12
    /// It is possible that the number increases even if the API does not have
    /// any change (when a new server is released with an unchanged API).
    /// </summary>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    string GetApiVersion();
}
```

3.7.4 Basic project operations

The **IServerProjectService** operations for managing server projects and related entity classes are discussed in this chapter.

The operations and entity classes involved are the following:

- Creating a new server project
  - IServerProjectService.CreateProject2
  - ServerProjectDesktopDocsCreateInfo
- Creating a new serve project based on a template
  - IserverProjectService.CreateProjectFromTemplate
  - TemplateBasedProjectCreateInfo
  - TemplateBasedProjectCreationResultInfo
- Updating the header information for an existing server project (such as description, deadline, etc.)
  - IServerProjectService.UpdateProject
  - ServerProjectUpdateInfo
- Retrieving a server project available on the server
  - IServerProjectService.GetProject
- Retrieving the list of server projects available on the server, optionally with filtering
  - IServerProjectService.ListProjects
  - ServerProjectListFilter
  - ServerProjectInfo
  - ServerProjectResourcesInPackages

- Wrapping up a server project
  - IServerProjectService.WrapUpProject
- Closing, un-closing a server project
  - IServerProjectService.SetClosedStatusOfProject
- Deleting a server project
  - IServerProjectService.DeleteProject
- Adding a target language to a server project
  - IServerProjectService.AddLanguageToProject
  - ServerProjectAddLanguageInfo
  - AddProjectLanguageTBHandlingBehavior
- Distributing a project;
  - IServerProjectService.DistributeProject

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    …
    /// <summary>
    /// Creates a new server project.
    /// </summary>
    /// <param name="spInfo"> The class encapsulating every information required
    /// to create a new server project.
    /// The user identified by the CreatorUser member of the spInfo parameter
    /// is automatically added to the project with the Project Manager role.
    /// </param>
    /// <returns>The guid of the newly created server project.</returns>
    [OperationContract]
    [ServiceKnownType(typeof(ServerProjectDesktopDocsCreateInfo))]
    Guid CreateProject2(ServerProjectDesktopDocsCreateInfo spInfo);
    /// <summary>
    /// Creates a new server project based on a template.
    /// </summary>
    /// <param name="createInfo">The class encapsulating every information required
    /// to create a new server project based on a template. The user identified by
    /// the CreatorUser member of the spInfo parameter is automatically added to
    /// the project with the Project Manager role.
    /// </param>
    /// <returns>The guid of the newly created server project.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault)),
    FaultContract(typeof(TemplateBasedProjectCreationFault)),
    FaultContract(typeof(TemplateBasedProjectCreationInvalidMetaFault))]
    TemplateBasedProjectCreationResultInfo CreateProjectFromTemplate(
        TemplateBasedProjectCreateInfo createInfo);
    /// <summary>
    /// Updates the header information (such as description, deadline) of an
    /// existing server project.
    /// </summary>
    /// <param name="spInfo">A ServerProjectInfo object holding information about
    /// the server project to be updated. Can not be null.</param>
```

```
    [OperationContract]
    void UpdateProject(ServerProjectUpdateInfo spInfo);
    /// <summary>
    /// Gets information about the specified project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <returns>The ServerProjectInfo object describing the server project.
    /// </returns>
    [OperationContract]
    ServerProjectInfo GetProject(Guid spGuid);
    /// <summary>
    /// Returns the list of server projects.
    /// </summary>
    /// <param name="filter">The filter object. If null, the returned list is
    /// not filtered.</param>
    /// <returns>
    /// An array of ServerProjectInfo objects each describing one server project.
    /// </returns>
    [OperationContract]
    ServerProjectInfo[] ListProjects(ServerProjectListFilter filter);
    /// <summary>
    /// Wraps up the project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    [OperationContract]
    [FaultContract(typeof(WrapUpProjectFault))]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    void WrapUpProject(Guid serverProjectGuid);
    /// <summary>
    /// Sets the closed status of project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="isClosed">The new closed status of the project.</param>
    [OperationContract]
    void SetClosedStatusOfProject(Guid serverProjectGuid, bool isClosed);
    /// <summary>
    /// Deletes the specified project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project
    /// to be deleted.</param>
    [OperationContract]
    void DeleteProject(Guid serverProjectGuid);
    /// <summary>
    /// Adds a new target language to the server project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.
    /// </param>
    /// <param name="addLanguageInfo">Encapsulates the parameters of adding
    /// the new target language to the server projects.</param>
    [OperationContract]
    void AddLanguageToProject(Guid serverProjectGuid,
        ServerProjectAddLanguageInfo addLanguageInfo);
    /// <summary>
    /// Distributes a project and sends out notifications to the members of the
    /// project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid if the server project.</param>
    [OperationContract]
    void DistributeProject(Guid serverProjectGuid);
    …
}
```

ServerProjectCreateInfo class

```
/// <summary>
/// Encapsulates information required to create a server project with Live
/// Docs.
/// </summary>
```

```
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
[KnownType( typeof( ServerProjectDesktopDocsCreateInfo ) )]
[Obsolete("This class is obsolete and will be removed in a future version. New
project will be created as if it was created using
ServerProjectDesktopDocsCreateInfo.")]
public partial class ServerProjectCreateInfo
{
    /// <summary>
    /// The name of the Project. Can not be null or empty.
    /// The name can not contain characters that are invalid in file names.
    /// </summary>
    [DataMember]
    public string Name;
    /// <summary>
    /// The description of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Description;
    /// <summary>
    /// The three+(two) letter code of the source language of the server project.
    /// (such as fre, eng, eng-US).
    /// </summary>
    [DataMember]
    public string SourceLanguageCode;
    /// <summary>
    /// The array of the three+(two) letter codes of the target languages of the
    /// server project (such as fre, eng, eng-US).  Can not be null and the items
    /// have to represent valid language codes.
    /// </summary>
    [DataMember]
    public string[] TargetLanguageCodes;
    /// <summary>
    /// The deadline of the project.
    /// </summary>
    [DataMember]
    public DateTime Deadline;
    /// <summary>
    /// The domain attribute of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Domain;
    /// <summary>
    /// The subject attribute of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Subject;
    /// <summary>
    /// The client attribute of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Client;
    /// <summary>
    /// The project attribute of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Project;
    /// <summary>
    /// The guid of the user creating the project. This user is automatically
    /// assigned to the project with ProjectManager role right after project
    /// creation. This ensures the there is a valid user assigned that has
    /// the permission to manipulate the projet from the MemoQ Client.
    /// This has to be the guid of a valid, existing user.
    /// </summary>
    [DataMember]
    public Guid CreatorUser;
    /// <summary>
    /// If true, online communication (chat) is enabled for the project.
```

```
/// No forum will be created/assigner to the project even if true.
/// </summary>
[DataMember]
public bool EnableCommunication;
/// <summary>
/// True if document versioning is enabled for the project.
/// </summary>
[DataMember]
public bool RecordVersionHistory;
/// <summary>
/// Indicates whether QA errors should prevent document deliver or not.
/// </summary>
[DataMember]
public bool PreventDeliveryOnQAError;
/// <summary>
/// Indicates whether the package creation is allowed or not.
/// </summary>
[DataMember]
public bool AllowPackageCreation;
/// <summary>
/// Indicates whether overlapping workflow phases are allowed or not.
/// It should be false if the AllowPackageCreation property is true.
/// </summary>
[DataMember]
public bool AllowOverlappingWorkflow;
/// <summary>
/// Includes the skeleton files (required for exporting documents)
/// when a user checks out the project or downloads the project
/// as a package. Without the files export is not possible.
/// Project managers always recieve these files.
/// Setting this flag has the same effect as setting DownloadSkeleton.
/// Either flag is true, the options is treated as enabled.
/// </summary>
[DataMember]
public bool DownloadSkeleton2;
/// <summary>
/// Creates preview when documents are imported. Includes the
/// preview of the documents when a user checks out the project
/// or downloads the project as a package. Without the preview
/// information no preview is available while translating.
/// Setting this flag has the same effect as setting DownloadPreview.
/// Either flag is true, the options is treated as enabled.
/// </summary>
[DataMember]
public bool DownloadPreview2;
/// <summary>
/// Specifies how the TMs, TBs and LiveDocs corpora are included
/// in the packages.
/// </summary>
[DataMember]
public ServerProjectResourcesInPackages PackageResourceHandling;
/// <summary>
/// The default values of custom fields that server saves to
/// the primary translation memory when you confirm segments
/// during translation.
/// Each line has three or more columns, separated by tabs:
///     metaName   metaType   value   [value2...valueN]
/// Possible metaType values:
///     FreeText
///     Number
///     DateTime
///     PickListSingle
///     PickListMultiple
/// </summary>
```

```
    [DataMember]
    public string CustomMetas;
}
```

The below example demonstrates how to use the CustomMetas field if you have three meta fields: one free text, one number and one single picklist. The <tab> strings mark the tabulator characters.

```
FirstMeta<tab>FreeText<tab>The value of the first meta

SecondMeta<tab>Number<tab>12

ThirdMeta<tab>PickListSingle<tab>Value 1<tab>Value 2<tab>Value 3
```

ServerProjectDesktopDocsCreateInfo class

```
/// <summary>
/// Encapsulates information required to create a server project with
/// Desktop Docs.
/// </summary>
[DataContract( Namespace = "http://kilgray.com/memoqservices/2007" )]
public partial class ServerProjectDesktopDocsCreateInfo : ServerProjectCreateInfo
{
    /// <summary>
    /// Enables the users to split and join segments.
    /// </summary>
    [DataMember]
    public bool EnableSplitJoin;
    /// <summary>
    /// When the project is checked out by a user, she will automatically
    /// get an offine copy of the Term bases and Translation memories
    /// (as opposed to having them used as remote resources).
    /// </summary>
    [DataMember]
    public bool CreateOfflineTMTBCopies;
    /// <summary>
    /// Downloads the skeleton files (required for exporting documents)
    /// when a user checks out the project. Without the files export is
    /// not possible. Project managers always recieve these files.
    /// Setting this flag has the same effect as setting DownloadSkeleton2.
    /// Either flag is true, the options is treated as enabled.
    /// </summary>
    [DataMember]
    public bool DownloadSkeleton;
    /// <summary>
    /// Downloads the preview of the documents when a user check out
    /// the project. Without the preview information no preview is
    /// available while translating.
    /// Setting this flag has the same effect as setting DownloadPreview2.
    /// Either flag is true, the options is treated as enabled.
    /// </summary>
    [DataMember]
    public bool DownloadPreview;
    /// <summary>
    /// This member is obsolete and value is ignored. All projects are
    /// enabled for web translation.
    /// </summary>
    [DataMember]
    [Obsolete("The value of this member is ignored.")]
    public bool EnableWebTrans;
    /// <summary>
    /// The callback url of an external web service that memoQ Server
    /// calls to notify of certain actions. The web service must meet
    /// the expected contract. Setting this turns on notification
    /// automatically.
    /// </summary>
    [DataMember]
```

```
    public bool CallbackWebServiceUrl;
}
```

TemplateBasedProjectCreateInfo class

```
/// <summary>
/// Encapsulates information required to create a server project from
/// a project template.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TemplateBasedProjectCreateInfo
{
    /// <summary>
    /// The guid of the template to use.
    /// </summary>
    [DataMember]
    public Guid TemplateGuid;
    /// <summary>
    /// The name of the Project. Can be null or empty if the specified
    /// template contains the name of the project.
    /// The name can not contain characters that are invalid in file
    /// names (e.g.: ",:,/,\,tab, ...).
    /// </summary>
    [DataMember]
    public string Name;
    /// <summary>
    /// The description of the project. Can be null and it will be taken
    /// into consideration if the template does not specify the description.
    /// </summary>
    [DataMember]
    public string Description;
    /// <summary>
    /// The three+(two) letter code of the source language of the server project
    /// (such as fre, eng, eng-US). Can be null if the template specifies the
    /// source language of the project. Overwrites the source language specified
    /// by the template if filled.
    /// </summary>
    [DataMember]
    public string SourceLanguageCode;
    /// <summary>
    /// The array of the three+(two) letter codes of the target languages of the
    /// server project (such as fre, eng, eng-US). Can be null if the template
    /// specifies the target languages of the project. Overwrites the target
    /// languages specified by the template if filled.
    /// </summary>
    [DataMember]
    public string[] TargetLanguageCodes;
    /// <summary>
    /// The domain attribute of the project. Can be null and it will be taken
    /// into consideration if the template does not specify the domain. It must
    /// be filled if the template refers to the domain placeholder.
    /// </summary>
    [DataMember]
    public string Domain;
    /// <summary>
    /// The subject attribute of the project. Can be null and it will be taken
    /// into consideration if the template does not specify the subject. It must
    /// be filled if the template refers to the subject placeholder.
    /// </summary>
    [DataMember]
    public string Subject;
    /// <summary>
    /// The client attribute of the project. Can be null and it will be taken
    /// into consideration if the template does not specify the client. It must
    /// be filled if the template refers to the client placeholder.
    /// </summary>
    [DataMember]
    public string Client;
```

```
    /// <summary>
    /// The project attribute of the project. Can be null and it will be taken
    /// into consideration if the template does not specify the project. It must
    /// be filled if the template refers to the project placeholder.
    /// </summary>
    [DataMember]
    public string Project;
    /// <summary>
    /// The special aspects of the project creation. The currently supported
    /// aspects are the following:
    /// - SkipUsersIfSourceLangHasBeenOverwritten: if present, then skip users
    ///   of the template if the source language specified by the template has
    ///   been overwritten.
    /// - SkipUsersAssignedToRemovedTargetLangs: if present, then skip users
    ///   assigned to target languages which were specified by the template but
    ///   the target languages of the template has been overwritten and some of
    ///   the assigned languages are missing.
    /// - AllowAmbiguousTMNamingRules: if present, then ambiguous TM naming
    ///   rules are allowed.
    /// </summary>
    [DataMember]
    public string[] ProjectCreationAspects;
    /// <summary>
    /// The guid of the user creating the project. This user is automatically
    /// assigned to the project with ProjectManager role right after project
    /// creation. This ensures the there is a valid user assigned that has
    /// the permission to manipulate the projet from the memoQ Client.
    /// This has to be the guid of a valid, existing user.
    /// </summary>
    [DataMember]
    public Guid CreatorUser;
}
```

TemplateBasedProjectCreationResultInfo class

```
/// <summary>
/// Encapsulates the result of a template-based project creation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TemplateBasedProjectCreationResultInfo : ResultInfo
{
    /// <summary>
    /// The guid of the created project.
    /// An empty guid if the project creation failed.
    /// </summary>
    [DataMember]
    public Guid ProjectGuid;
}
```

ServerProjectUpdateInfo class

```
/// <summary>
/// Encapsulates information required to update server project header information.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectUpdateInfo
{
    /// <summary>
    /// The guid of the server project.
    /// </summary>
    [DataMember]
    public Guid ServerProjectGuid;
    /// <summary>
    /// The description of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Description;
    /// <summary>
```

```
    /// The deadline of the project.
    /// </summary>
    [DataMember]
    public DateTime Deadline;
    /// <summary>
    /// The domain attribute of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Domain;
    /// <summary>
    /// The subject attribute of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Subject;
    /// <summary>
    /// The client attribute of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Client;
    /// <summary>
    /// The project attribute of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Project;
    /// <summary>
    /// The callback url of an external web service that memoQ Server
    /// calls to notify of certain actions. The web service must meet
    /// the expected contract. Setting this turns on notification
    /// automatically. Setting to null turns off notification.
    /// </summary>
    [DataMember]
    public bool CallbackWebServiceUrl;
    /// <summary>
    /// The default values of custom fields that server saves to
    /// the primary translation memory when you confirm segments
    /// during translation. The server will not change the default
    /// values if this field is null.
    /// Each line has three or more columns, separated by tabs:
    ///     metaName   metaType   value   [value2...valueN]
    /// Possible metaType values:
    ///     FreeText
    ///     Number
    ///     DateTime
    ///     PickListSingle
    ///     PickListMultiple
    /// </summary>
    [DataMember]
    public string CustomMetas;
}
```

The below example demonstrates how to use the CustomMetas field if you have three meta fields: one free text, one number and one single picklist. The <tab> strings mark the tabulator characters.

```
FirstMeta<tab>FreeText<tab>The value of the first meta

SecondMeta<tab>Number<tab>12

ThirdMeta<tab>PickListSingle<tab>Value 1<tab>Value 2<tab>Value 3
```

ServerProjectListFilter class

```
/// <summary>
/// Encapsulates filtering information for listing server projects.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectListFilter
{
```

```
    /// <summary>
    /// This parameter can be used to filter the list based on the SOURCE language
    /// of the server project. The three+(two) letter langusage code of the
    /// language is to be provided, such as fre, eng, eng-US. If a three letter
    /// language code is provided all three+two letter matches are returned (e.g.
    /// if eng is specified then server projects with source language eng-US,
    /// eng-gb, ... and eng are returned. If this parameter is null, no source
    /// language filtering is applied.
    /// </summary>
    [DataMember]
    public string SourceLanguageCode;
    /// <summary>
    /// This parameter can be used to filter the list based on the TARGET languages
    /// of the server project. A project is included in the filtered list if its
    /// target languages contain the language identified by TargetLanguageCode.
    /// The three+(two) letter langusage code of the
    /// language is to be provided, such as fre, eng, eng-US. If a three letter
    /// language code is provided all three+two letter matches are returned (e.g.
    /// if eng is specified then server projects with source language eng-US,
    /// eng-gb, ... and eng are returned. If this parameter is null, no target
    /// language filtering is applied.
    /// </summary>
    [DataMember]
    public string TargetLanguageCode;
    /// <summary>
    /// The domain attribute of the project. If null, no filtering is applied.
    /// </summary>
    [DataMember]
    public string Domain;
    /// <summary>
    /// The subject attribute of the project. If null, no filtering is applied.
    /// </summary>
    [DataMember]
    public string Subject;
    /// <summary>
    /// The client attribute of the project. If null, no filtering is applied.
    /// </summary>
    [DataMember]
    public string Client;
    /// <summary>
    /// The project attribute of the project. If null, no filtering is applied.
    /// </summary>
    [DataMember]
    public string Project;
    /// <summary>
    /// The TimeClosed attribute of the project. Replaces former IsArchived.
    /// Only projects that were closed after (or at the same time) TimeClosed are
    /// returned (including unclosed projects!).
    /// Recommended usage:
    /// - Set it to a distant future value (equal or greater than 1/1/2500) to list
    ///   unclosed projects only. This is the tipical way of listing server
    ///   projects.
    /// - Set to a datetime value in the past to include projects closed
    ///   (after the specified time). If set to a small datetime value
    ///   (e.g 1/1/1900)
    ///   all projects are returned. If you need the list of projects not including
    ///   unclosed ones, then remove unclosed projects from the result set in
    ///   your system (based on the TimeClosed attribute of the returned
    ///   ServerProjectInfo objects.
    ///   List server projects including closed ones only if you have a reason
    ///   to do so, as the returned result set may be significantly larger in this
    ///   case.
    /// </summary>
    [DataMember]
    public DateTime TimeClosed;
}
```

ServerProjectStatus enum

```
/// <summary>
/// Describes the possible statuses of an online project.
/// </summary>
public enum ServerProjectStatus
{
    /// <summary>
    /// The project is live, the users are working on it.
    /// </summary>
    Live = 0,
    /// <summary>
    /// The project is finished, it has been wrapped up.
    /// </summary>
    WrappedUp = 1
}
```

ServerProjectInfo class

```
/// <summary>
/// Encapsulates information about a server project.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectInfo
{
    /// <summary>
    /// The guid of the server project.
    /// </summary>
    [DataMember]
    public Guid ServerProjectGuid;
    /// <summary>
    /// The name of the Project. Can not be null or empty.
    /// </summary>
    [DataMember]
    public string Name;
    /// <summary>
    /// The description of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Description;
    /// <summary>
    /// The three+(two) letter code of the source language of the server project.
    /// (such as fre, eng, eng-US).  Can not be null or empty.
    /// </summary>
    [DataMember]
    public string SourceLanguageCode;
    /// <summary>
    /// The three+(two) letter code of the target languages of the server project.
    /// (such as fre, eng, eng-US).  Can not be null and the items can not be
    /// null or empty.
    /// </summary>
    [DataMember]
    public string[] TargetLanguageCodes;
    /// <summary>
    /// The deadline of the project.
    /// </summary>
    [DataMember]
    public DateTime Deadline;
    /// <summary>
    /// The domain attribute of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Domain;
    /// <summary>
    /// The subject attribute of the project. Can be null.
    /// </summary>
    [DataMember]
    public string Subject;
```

```
/// <summary>
/// The client attribute of the project. Can be null.
/// </summary>
[DataMember]
public string Client;
/// <summary>
/// The project attribute of the project. Can be null.
/// </summary>
[DataMember]
public string Project;
/// <summary>
/// The guid of the user creating the project. T
/// </summary>
[DataMember]
public Guid CreatorUser;
/// <summary>
/// The date and time the server project was created.
/// </summary>
[DataMember]
public DateTime CreationTime;
/// <summary>
/// If true, online communication (chat) is enabled for the project.
/// No forum will be created/assigned to the project even if true.
/// </summary>
[DataMember]
public bool EnableCommunication;
/// <summary>
/// The total number of segments of all documents of the server project,
/// including the number of locked segments.
/// </summary>
[DataMember]
public int TotalSegmentCount;
/// <summary>
/// The total number of confirmed segments of all documents of the server
/// project.
/// </summary>
[DataMember]
public int ConfirmedSegmentCount;
/// <summary>
/// The total number of reviewer 1confirmed segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int ReviewerConfirmedSegmentCount;
/// <summary>
/// The total number of proofred segments of all documents of the server
/// project.
/// </summary>
[DataMember]
public int ProofreadSegmentCount;
/// <summary>
/// The total number of locked segments of all documents of the server
/// project.
/// </summary>
[DataMember]
public int LockedSegmentCount;
/// <summary>
/// The total number of characters of all documents of the server project,
/// including the number of characters in locked segments.
/// </summary>
[DataMember]
public int TotalCharacterCount;
/// <summary>
/// The total number of characters in confirmed segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int ConfirmedCharacterCount;
```

```
/// <summary>
/// The total number of characters in reviewer 1confirmed segments of all
/// documents of the server project.
/// </summary>
[DataMember]
public int Reviewer1ConfirmedCharacterCount;
/// <summary>
/// The total number of characters in proofred segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int ProofreadCharacterCount;
/// <summary>
/// The total number of characters in locked segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int LockedCharacterCount;
/// <summary>
/// The total number of words of all documents of the server project,
/// including the number of words in locked segments.
/// </summary>
[DataMember]
public int TotalWordCount;
/// <summary>
/// The total number of words in confirmed segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int ConfirmedWordCount;
/// <summary>
/// The total number of words in reviewer 1 confirmed segments of all
/// documents of the server project.
/// </summary>
[DataMember]
public int Reviewer1ConfirmedWordCount;
/// <summary>
/// The total number of words in proofred segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int ProofreadWordCount;
/// <summary>
/// The total number of words in locked segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int LockedWordCount;
/// <summary>
/// The status of the document.
/// </summary>
[DataMember]
public DocumentStatus DocumentStatus;
/// <summary>
/// The date and time when the project was closed. If the project is not
/// closed, the value is 01/01/2500.
/// </summary>
[DataMember]
public DateTime TimeClosed;
/// <summary>
/// True if the project uses DesktopDocs; false for LiveDocs.
/// </summary>
[DataMember]
public bool DesktopDocs;
/// <summary>
/// True if document versioning is enabled for the project.
/// </summary>
[DataMember]
```

```
public bool RecordVersionHistory;
/// <summary>
/// Enables the users to split and join segments.
/// </summary>
[DataMember]
public bool EnableSplitJoin;
/// <summary>
/// When the project is checked out by a user, she will automatically
/// get an offine copy of the Term bases and Translation memories
/// (as opposed to having them used as remote resources).
/// </summary>
[DataMember]
public bool CreateOfflineTMTBCopies;
/// <summary>
/// Downloads the skeleton files (required for exporting documents)
/// when a user checks out the project. Without the files export is
/// not possible. Project managers always recieve these files.
/// </summary>
[DataMember]
public bool DownloadSkeleton;
/// <summary>
/// Downloads the preview of the documents when a user check out
/// the project. Without the preview information no preview is
/// available while translating.
/// </summary>
[DataMember]
public bool DownloadPreview;
/// <summary>
/// Enables web based translation for the project. Split/join is not
/// supported for web based translation, therefore both EnableWebTrans
/// and EnableSplitJoin can not be true (an exception is thrown in
/// this case). For projects created since memoQ server verion 7.0
/// this member is always true.
/// </summary>
[DataMember]
public bool EnableWebTrans;
/// <summary>
/// Indicates whether QA errors should prevent document deliver or not.
/// </summary>
[DataMember]
public bool PreventDeliveryOnQAError;
/// <summary>
/// Indicates whether the package creation is allowed or not.
/// </summary>
[DataMember]
public bool AllowPackageCreation;
/// <summary>
/// Indicates whether overlapping workflow phases are allowed or not.
/// It should be false if the AllowPackageCreation property is true.
/// </summary>
[DataMember]
public bool AllowOverlappingWorkflow;
/// <summary>
/// Specifies how the TMs, TBs and LiveDocs corpora are included
/// in the packages. Only for projects that allow package creation.
/// </summary>
[DataMember]
public ServerProjectResourcesInPackages PackageResourceHandling;
/// <summary>
/// The default values of custom fields that server saves to
/// the primary translation memory when you confirm segments
/// during translation. It can be an empty string if there
/// are no custom meta values set in the project.
/// Each line has three or more columns, separated by tabs:
///     metaName   metaType   value   [value2...valueN]
/// Possible metaType values:
///     FreeText
///     Number
```

```
    ///     DateTime
    ///     PickListSingle
    ///     PickListMultiple
    /// </summary>
    [DataMember]
    public string CustomMetas;
    /// <summary>
    /// The status of the project.
    /// </summary>
    [DataMember]
    public ServerProjectStatus ProjectStatus;
}
```

The below example demonstrates how to use the CustomMetas field if you have three meta fields: one free text, one number and one single picklist. The <tab> strings mark the tabulator characters.

```
FirstMeta<tab>FreeText<tab>The value of the first meta

SecondMeta<tab>Number<tab>12

ThirdMeta<tab>PickListSingle<tab>Value 1<tab>Value 2<tab>Value 3
```

ServerProjectResourcesInPackages enum

```
/// <summary>
/// Describes the options how the TMs, TBs and LiveDocs corpora are included
/// in packages.
/// </summary>
public enum ServerProjectResourcesInPackages
{
    /// <summary>
    /// Create a project TM and TB and include them in the package.
    /// </summary>
    CreateProjectTMTB,
    /// <summary>
    /// Link TMs, TBs and LiveDocs corpora as remote resources.
    /// </summary>
    LinkRemote,
    /// <summary>
    /// Include copies of TMs, TBs and LiveDocs corpora in the package.
    /// </summary>
    IncludeCopy
}
```

ServerProjectAddLanguageInfo class

```
/// <summary>
/// Encapsulates the parameters of adding a new target language to a
/// server projects.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectAddLanguageInfo
{
    /// <summary>
    /// The three+(two) letter code
    /// (such as fre, eng, eng-US)of the language that is to be added as
    /// target language to the server project.
    /// </summary>
    [DataMember]
    public string TargetLangCode;
    /// <summary>
    /// If not null, documents are copied to the new target language.
    /// In this case identifies which version of the documents are
    /// to be used as the source of the copy operation via their target
    /// language code (as each document may have a different segmentation
    /// set for each target language).
```

```
    /// Either has to be null, or has to have a valid target language
    /// code of the project.
    /// </summary>
    [DataMember]
    public string TargetLangOfSourceDoc;
    /// <summary>
    /// If TargetLangOfSourceDoc is not null (that is documents are
    /// to be copied to the new target language), then:
    /// - If false, only the source segments are copied (the target
    ///   segments are left empty).
    /// - If true, the target segments are also copied.
    /// If TargetLangOfSourceDoc is null, its value is ignored.
    /// </summary>
    [DataMember]
    public bool CopyTargetSideFromSourceDocument;
    /// <summary>
    /// The termbase language handling behavior during the operations.
    /// </summary>
    [DataMember]
    public AddProjectLanguageTBHandlingBehavior TBLanguageHandlingBehavior;
}
```

AddProjectLanguageTBHandlingBehavior enumeration

```
/// <summary>
/// Represents termbase handling behavior during the add
/// project language operation when the project termbases
/// do not contain the neccessary languages.
/// </summary>
public enum AddProjectLanguageTBHandlingBehavior
{
    /// <summary>
    /// The new languages will be added automatically.
    /// </summary>
    AddMissingLanguages,
    /// <summary>
    /// The termbases which do not contain the neccessary
    /// languages will be removed.
    /// </summary>
    RemoveTBs,
    /// <summary>
    /// A specific fault will be thrown if the project
    /// termbases do no contain the neccessary languages.
    /// </summary>
    ThrowFault
}
```

Reported errors

Important note: When using the Server Project API make sure to enable advanced error handling mode.

The WrapUpProject, the AddLanguageToProject and the CreateProjectFromTemplate functions report various types of errors. Besides the generally used UnexpectedFault and GenericFault (see 2.5.2 Exception categories for the details) some specific faults are used (WrapUpProjectFault, AddProjectLanguageFault, TemplateBasedProjectCreatinFault and TemplateBasedProjectCreationInvalidMetaFault). These faults have an ErrorCode member so that the caller can differentiate between different types of errors. Using older WS technologies it is difficult to access the ErrorCode: you can check the fault code instead, which has the same value as the ErrorCode. Possible fault types, ErrorCode/fault code values with their fault message are the following:

Fault type	Fault code (same as ErrorCode in case of AddProjectLanguageFault)	Fault message
WrapUpProjectFault	WrapUpProject_ProjectAlreadyWrappedUp	The project is already wrapped up.
AddProjectLanguageFault	AddProjectLanguage_MissingTBLanguage	Some of the project termbases does not contain the new target language.
TemplateBasedProjectCreationFault	TemplateBasedProjectCreation_ProjectNameNotSpecified	The name of the project is not specified.
TemplateBasedProjectCreationFault	TemplateBasedProjectCreation_SourceLanguageNotSpecified	The source language of the project is not specified.
TemplateBasedProjectCreationFault	TemplateBasedProjectCreation_TargetLanguagesNotSpecified	The target languages of the project are not specified.
TemplateBasedProjectCreationFault	TemplateBasedProjectCreation_ProjectNotSpecified	The project meta of the project is not specified.
TemplateBasedProjectCreationFault	TemplateBasedProjectCreation_ClientNotSpecified	The client meta of the project is not specified.
TemplateBasedProjectCreationFault	TemplateBasedProjectCreation_DomainNotSpecified	The domain meta of the project is not specified.
TemplateBasedProjectCreationFault	TemplateBasedProjectCreation_SubjectNotSpecified	The subject meta of the project is not specified.
TemplateBasedProjectCreationFault	TemplateBasedProjectCreation_AmbiguousWorkingTMNamingRule	The working TM naming rule is ambiguous.
TemplateBasedProjectCreationFault	TemplateBasedProjectCreation_AmbiguousMasterTMNamingRule	The master TM naming rule is ambiguous.
TemplateBasedProjectCreationInvalidMetaFault	TemplateBasedProjectCreation_ProjectIsNotInThePredefinedSet	The project meta is not in the pre-defined set defined in the template.
TemplateBasedProjectCreationInvalidMetaFault	TemplateBasedProjectCreation_ClientIsNotInThePredefinedSet	The client meta is not in the pre-defined set defined in the template.
TemplateBasedProjectCreationInvalidMetaFault	TemplateBasedProjectCreation_DomainIsNotInThePredefinedSet	The domain meta is not in the pre-defined set defined in the template.

TemplateBasedProjectCreationInvalidMetaFault	TemplateBasedProjectCreation_SubjectIsNotInThePredefinedSet	The subject meta is not in the pre-defined set defined in the template.
GenericFault	Generic	<The message of the original exception is used as fault message.>
UnexpectedFault	Unexpected	<The message of the original exception is used as fault message.>

Please note that using C# and WCF these faults can be caught based on their type using regular exceptions:

```
try
{
    // call operation
}
// This catches TMFault errors only
catch (System.ServiceModel.FaultException<AddProjectLanguageFault> e)
{

    // Check e.Code.Name or e.ErrorCode for the fault code
}
// This catches all other API errors
catch (System.ServiceModel.FaultException e)
{

    // Check e.Code.Name or e.ErrorCode for the fault code if you want to
    // differentiate between errors
}
```

Using older WS technologies, such as ASMX, you need to check the fault code:

```
try
{
    // call operation
}
// Important: this catches all API errors, not only AddProjectLanguageFault
// errors!!!
catch (SoapException e)
{
    // Check e.Code.Name or e.ErrorCode for the fault code if you want to
    // differentiate between errors
}
```

3.7.5 Translation document import/export

The **IServerProjectService** operations for translation document import/export and related entity classes are discussed in this chapter.

The operations and entity classes involved are the following:

- Common entity classes:
  - `ResultStatus`
  - `ResultInfo`
  - `FileResultInfo`
- Importing documents in primary format, such as plain text, XML, RTF, etc.

    - IServerProjectService.ImportTranslationDocument
    - IServerProjectService.ImportTranslationDocuments
    - IServerProjectService.ImportTranslationDocumentsWithOptions
    - ImportTranslationDocumentOptions
    - TranslationDocImportResultInfo
- Reimporting documents in primary format, such as plain text, XML, RTF, etc.
    - ReimportDocumentOptions
- Importing document from a bilingual file: memoQ Bilingual (MDB) and XLIFF formats are supported.
    - IServerProjectService.ImportBilingualTranslationDocument
    - BilingualDocFormat
- Updating a document that is already part of the project from a bilingual file.
    - IServerProjectService.UpdateTranslationDocumentFromBilingual
    - BilingualDocFormat
- Exporting a translation document to its primary (original) format, such as plain text, XML, RTF, etc.
    - IServerProjectService. ExportTranslationDocument
    - IServerProjectService. ExportTranslationDocument2
    - TranslationDocExportResultInfo
- Exporting a translation document to a bilingual format: XLIFF, Two-column RTF and Bilingual RTF formats are supported.
    - ServerProjectCreateInfo.ExportTranslationDocumentAsXliffBilingual
    - ServerProjectCreateInfo.ExportTranslationDocumentAsRtfBilingual
    - FileResultInfo
    - XliffBilingualExportOptions
    - RtfBilingualExportOptions
    - TwoColumnRtfBilingualExportOptions
    - DocumentExportOptions

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// Imports the translation document, that can be of the file formats
/// supported by memoQ.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="fileGuid">The guid of the file to be imported. The file
/// has to be uploaded before calling this operation by
/// IFileManagerService.BeginChunkedFileUpload and related operations.
/// This guid is returned by IFileManagerService.BeginChunkedFileUpload.
/// </param>
/// <param name="targetLangCodes">The three+(two) letter code of the target
/// languages the document is to be imported into. If null, the document
/// is imported into all target languages of the project. If not null,
/// only target languages that are subset of the target languages of the
/// project can be specified.</param>
```

```
/// <param name="importSettingsXML">The settings of the import in memoQ
/// import filter XML format. This setting also determines the filter
/// (txt, doc, etc.) to be used. If null, the extension of the file is used
/// to determine the import filter.
/// The format of the file has slightly changed in memoQ version 4.2: it
/// is the new export format of filter configuration resources (you can
/// create one by exporting a memoQ filter configuration resource using
/// memoQ client).
/// </param>
/// <returns>
/// The TranslationDocImportResultInfo object providing information about the
/// success or failure of the operation, and holding the Guid(s) of the newly
/// imported document(s).
/// </returns>
[OperationContract]
TranslationDocImportResultInfo ImportTranslationDocument(
    Guid serverProjectGuid, Guid fileGuid,
    string[] targetLangCodes, string importSettingsXML);
/// <summary>
/// Imports multiple translation documents with the same import settings,
/// The documents can be of the file formats supported by memoQ).
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="fileGuids">The guids of the files to be imported. The files
/// have to be uploaded before calling this operation by
/// IFileManagerService.BeginChunkedFileUpload and related operations.
/// Each guid is retuned by IFileManagerService.BeginChunkedFileUpload.</param>
/// <param name="targetLangCodes">The three+(two) letter code of the target
/// languages the document is to be imported into. If null, the document
/// is imported into all target languages of the project. If not null,
/// only target languages that are subset of the target languages of the
/// project can be specified.</param>
/// <param name="importSettingsXML">The settings of the import in memoQ
/// import filter XML format. This setting also determines the filter
/// (txt, doc, etc.) to be used.
/// If null, the extension of the file is used to determine the import
/// filter.
/// The format of the file has slightly changed in memoQ version 4.2: it
/// is the new export format of filter configuration resources (you can
/// create one by exporting a memoQ filter configuration resource using
/// memoQ client).
/// </param>
/// <returns>
/// The TranslationDocImportResultInfo objects providing information about
/// the success or failure of the operation, and holding the Guids of the
/// newly imported documents. This array has one matching item for each
/// item in the fileGuids array (in the same order).
/// </returns>
[OperationContract]
TranslationDocImportResultInfo[] ImportTranslationDocuments(
    Guid serverProjectGuid, Guid[] fileGuids,
    string[] targetLangCodes, string importSettingsXML);
/// <summary>
/// Imports a translation document with the import settings (filter
/// configuration) defined by the guid of an existing filter configuration
/// resource. Documents can be of the file formats supported by memoQ.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="fileGuid">The guid of the file to be imported. The file
/// has to be uploaded before calling this operation by
/// IFileManagerService.BeginChunkedFileUpload and related operations.
/// This guid is returned by IFileManagerService.BeginChunkedFileUpload.
/// </param>
/// <param name="targetLangCodes">The three+(two) letter code of the target
/// languages the document is to be imported into. If null, the document
/// is imported into all target languages of the project. If not null,
/// only target languages that are subset of the target languages of the
/// project can be specified.</param>
```

```
/// <param name="filterConfigResGuid">
/// The guid of the filter configuration resource to be used during the
/// import. The selected filter configuration resource defines the settings
/// of the import. This setting also determines the filter
/// (txt, doc, etc.) to be used.
/// </param>
/// <returns>
/// The TranslationDocImportResultInfo object providing information about the
/// success or failure of the operation, and holding the Guid(s) of the newly
/// imported document(s).
/// </returns>
[OperationContract]
TranslationDocImportResultInfo
 ImportTranslationDocumentWithFilterConfigResource(
    Guid serverProjectGuid, Guid fileGuid,
    string[] targetLangCodes, Guid filterConfigResGuid);
/// <summary>
/// Imports multiple translation documents with the import settings (filter
/// configuration) defined by the guid of an existing filter configuration
/// resource. Documents can be of the file formats supported by memoQ.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="fileGuids">The guids of the files to be imported. The files
/// have to be uploaded before calling this operation by
/// IFileManagerService.BeginChunkedFileUpload and related operations.
/// Each guid is retuned by IFileManagerService.BeginChunkedFileUpload.</param>
/// <param name="targetLangCodes">The three+(two) letter code of the target
/// languages the document is to be imported into. If null, the document
/// is imported into all target languages of the project. If not null,
/// only target languages that are subset of the target languages of the
/// project can be specified.</param>
/// <param name="filterConfigResGuid">
/// The guid of the filter configuration resource to be used during the
/// import. The selected filter configuration resource defines the settings
/// of the import. This setting also determines the filter
/// (txt, doc, etc.) to be used.
/// </param>
/// <returns>
/// The TranslationDocImportResultInfo objects providing information about
/// the success or failure of the operation, and holding the Guids of the
/// newly imported documents. This array has one matching item for each
/// item in the fileGuids array (in the same order).
/// </returns>
[OperationContract]
TranslationDocImportResultInfo[]
    ImportTranslationDocumentsWithFilterConfigResource(
    Guid serverProjectGuid, Guid[] fileGuids, string[] targetLangCodes,
    Guid filterConfigResGuid);
/// <summary>
/// Imports multiple translation documents into a server project. The options
/// of the import are configured for each file separately using an instance
/// of ImportTranslationDocumentOptions.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="importDocOptions">The files to import and their respecive
/// settings. Each item in the array corresponds to a file to import.</param>
/// <returns>
/// The TranslationDocImportResultInfo objects providing information about
/// the success or failure of the operation, and holding the Guids of the
/// newly imported documents. This array has one matching item for each
/// item in the importDocOptions array (in the same order).
/// </returns>
[OperationContract]
TranslationDocImportResultInfo[]
    ImportTranslationDocumentsWithOptions(
    Guid serverProjectGuid,
    ImportTranslationDocumentOptions[] importDocOptions);
/// <summary>
```

```
/// Imports the bilingual translation document that can be in memoQ
/// bilingual, XLIFF bilingual format or TwoColumnRTF format.
/// If the document with the same document Guid is already part of the project
/// an exception is thrown (call UpdateTranslationDocumentFromBilingual
/// instead).
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="fileGuid">The guid of the bilingual file to be imported.
/// The file has to be uploaded before calling this operation by
/// IFileManagerService.BeginChunkedFileUpload and related operations.
/// This guid is retuned by IFileManagerService.BeginChunkedFileUpload.</param>
/// <param name="docFormat">The format of the bilingual document to be
/// imported.</param>
/// <param name="targetLangCodes">The three+(two) letter codes of the target
/// languages the document is to be imported into.
/// - If the document is in memoQ bilingual format (mbd), then:
///     - If targetLangCodes is null, the document is imported to all target
///       languages of the project.
///     - Otherwise, it is imported to the target languages defined by
///       targetLangCodes.
/// - If the document is XLIFF, the targetlangcode can only be null, or can
///   contain only the xliff's target language. In both case the document is
///   imported into one target language (the one defined by the document
///   itself.
///   If the XLIFF document contains more than one document, all have to have
///   the same source and target language.
/// <returns>
/// The TranslationDocImportResultInfo objects providing information about
/// the success or failure of the operation. This array has one item unless
/// an XLIFF document containing more than one doument is imported. If a
/// document is imported into more than one language, the DocumentGuids
/// member of the TranslationDocImportResultInfo object has more than one
/// item (one for each target language).
/// </returns>
/// <remarks>Bilingual documents exported by pre 4.0 versions of memoQ
/// are not supported.</remarks>
[OperationContract]
TranslationDocImportResultInfo[] ImportBilingualTranslationDocument(
    Guid serverProjectGuid, Guid fileGuid,
    BilingualDocFormat docFormat, string[] targetLangCodes);
/// <summary>
/// Updates an existing translation translation document based on the
/// bilingual document specified by fileGuid. The bilingual document can
/// be XLIFF bilingual format or TwoColumnRTF, memoQ Bilingual (mbd) is not
/// supported. If the document to be updated is not part of the
/// project, an exception is thrown (call ImportBilingualTranslationDocument
/// instead).
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="fileGuid">The guid of the bilingual file to be imported.
/// The file has to be uploaded before calling this operation by
/// IFileManagerService.BeginChunkedFileUpload and related operations.
/// This guid is retuned by IFileManagerService.BeginChunkedFileUpload.
/// </param>
/// <param name="docFormat">The format of the bilingual document to be
/// imported.</param>
/// <returns>
/// The TranslationDocImportResultInfo objects providing information about
/// the success or failure of the operation. This array has one item unless
/// an XLIFF document containing more than one doument is imported. If a
/// document is updated into more than one language, the DocumentGuids
/// member of the TranslationDocImportResultInfo object has more than one
/// item (one for each target language). The Guid of the document is
/// preserved during the update (so the document Guid found in the
/// document is returned).
/// </returns>
/// <remarks>The translation document to be updated is determined by
/// the Guid stored in the bilingual document.Bilingual documents exported
```

```
/// by pre 4.0 versions of memoQ are not supported.
/// </remarks>
[OperationContract]
TranslationDocImportResultInfo[] UpdateTranslationDocumentFromBilingual(
    Guid serverProjectGuid, Guid fileGuid, BilingualDocFormat docFormat);
/// <summary>
/// Exports the translation document in its original format.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="docGuid">The unique identifier of the document, the
/// DocumentGuid property of the ServerProjectTranslationDocInfo
/// instance which represents the document.</param>
/// <returns>
/// The TranslationDocExportResultInfo providing information about the result
/// of the operation.
/// In case of success or warning, it holds guid of the resulting file.
/// The file can be downloaded using
/// IFileManagerService.BeginChunkedFileDownload
/// and related operations.
/// Exporting a document which has linked child documents (see
/// ServerProjectTranslationDocInfo.ParentDocumentId) will result in
/// the export of all child documents as well, embedded in the parent
/// document.
/// </returns>
[OperationContract]
TranslationDocExportResultInfo ExportTranslationDocument(
    Guid serverProjectGuid, Guid docGuid);
/// <summary>
/// Exports the translation document in its original format.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="docGuid">The unique identifier of the document, the
/// DocumentGuid property of the ServerProjectTranslationDocInfo
/// instance which represents the document.</param>
/// <param name="options">Options for controlling the export behavor.</param>
/// <returns>
/// The TranslationDocExportResultInfo providing information about the result
/// of the operation.
/// In case of success or warning, it holds guid of the resulting file.
/// The file can be downloaded using
/// IFileManagerService.BeginChunkedFileDownload
/// and related operations.
/// Exporting a document which has linked child documents (see
/// ServerProjectTranslationDocInfo.ParentDocumentId) will result in
/// the export of all child documents as well, embedded in the parent
/// document.
/// Exporting a multilingual document (imported using the Multilingual
/// filters) will result in exporting all multilingual siblings into
/// a single result file.
/// </returns>
[OperationContract]
TranslationDocExportResultInfo ExportTranslationDocument2(
    Guid serverProjectGuid, Guid docGuid, DocumentExportOptions options);
/// <summary>
/// Exports the translation document as XLIFF bilingual.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="docGuid">The unique identifier of the document, the
/// DocumentGuid property of the ServerProjectTranslationDocInfo
/// instance which represents the document.</param>
/// <param name="options">The options of the export.</param>
/// <returns>
/// The FileResultInfo providing information about the result
/// of the operation.
/// In case of success or warning, it holds guid of the resulting file.
/// The file can be downloaded using
/// IFileManagerService.BeginChunkedFileDownload and related operations.
/// </returns>
```

```
[OperationContract]
FileResultInfo ExportTranslationDocumentAsXliffBilingual(
    Guid serverProjectGuid, Guid docGuid,
    XliffBilingualExportOptions options);
/// <summary>
/// Exports the translation document as Trados-like RTF bilingual.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="docGuid">The unique identifier of the document, the
/// DocumentGuid property of the ServerProjectTranslationDocInfo
/// instance which represents the document.</param>
/// <param name="options">The options of the export.</param>
/// <returns>
/// The FileResultInfo providing information about the result
/// of the operation.
/// In case of success or warning, it holds guid of the resulting file.
/// The file can be downloaded using
/// IFileManagerService.BeginChunkedFileDownload  and related operations.
/// </returns>
[OperationContract]
FileResultInfo ExportTranslationDocumentAsRtfBilingual(Guid serverProjectGuid,
    Guid docGuid, RtfBilingualExportOptions options );
/// <summary>
/// Exports the translation document as two-column RTF bilingual.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="docGuid">The unique identifier of the document, the
/// DocumentGuid property of the ServerProjectTranslationDocInfo
/// instance which represents the document.</param>
/// <param name="options">The options of the export.</param>
/// <returns>
/// The FileResultInfo providing information about the result
/// of the operation.
/// In case of success or warning, it holds guid of the resulting file.
/// The file can be downloaded using
/// IFileManagerService.BeginChunkedFileDownload  and related operations.
/// </returns>
[OperationContract]
FileResultInfo ExportTranslationDocumentAsTwoColumnRtf( Guid serverProjectGuid,
    Guid docGuid, TwoColumnRtfBilingualExportOptions options );
 /// <summary>
/// Re-imports translation documents overwriting existing documents in the
/// server project.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="reimportActions">The reimport actions for each import file.
/// Each re-imported file replaces one or more documents in the server project.
/// The document(s) replaced by a single re-imported file correspond to
/// different target languages. The documents to overwrite are identified by
/// their unique identifier DocumentGuid property of the
/// ServerProjectTranslationDocInfo instance which represents the
/// document. This array must contain as many items as many files are
/// re-imported.</param>
/// <param name="importSettingsXML">The settings of the import in memoQ
/// import filter XML format. This setting also determines the filter
/// (txt, doc, etc.) to be used.
/// If null, the extension of the file is used to determine the import
/// filter.
/// The format of the file has slightly changed in memoQ version 4.2: it
/// is the new export format of filter configuration resources (you can
/// create one by exporting a memoQ filter configuration resource using
/// memoQ client).
/// </param>
/// <returns>
/// The TranslationDocImportResultInfo objects providing information about
/// the success or failure of the operation, and holding the Guids of the
/// newly imported documents. This array has one matching item for each
/// item in the fileGuids array (in the same order).
```

```
    /// </returns>
    [OperationContract]
    TranslationDocImportResultInfo[] ReImportTranslationDocuments(
        Guid serverProjectGuid, ReimportDocumentOptions[] reimportActions,
            string importSettingsXML);
    /// <summary>
    /// Re-imports translation documents overwriting existing documents in the
    /// server project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="reimportActions">The reimport actions for each import file.
    /// Each re-imported file replaces one or more documents in the server project.
    /// The document(s) replaced by a single re-imported file correspond to
    /// different target languages. The documents to overwrite are identified by
    /// their unique identifier DocumentGuid property of the
    /// ServerProjectTranslationDocInfo instance which represents the
    /// document. This array must contain as many items as many files are
    /// re-imported.</param>
    /// <param name="filterConfigResGuid">
    /// The guid of the filter configuration resource to be used during the
    /// import. The selected filter configuration resource defines the settings
    /// of the import. This setting also determines the filter
    /// (txt, doc, etc.) to be used.
    /// </param>
    /// <returns>
    /// The TranslationDocImportResultInfo objects providing information about
    /// the success or failure of the operation, and holding the Guids of the
    /// newly imported documents. This array has one matching item for each
    /// item in the fileGuids array (in the same order).
    /// </returns>
    [OperationContract]
    TranslationDocImportResultInfo[]
        ReImportTranslationDocumentsWithFilterConfigResource (
            Guid serverProjectGuid, ReimportDocumentOptions[] reimportActions,
            Guid filterConfigResGuid);
}
```

ResultStatus enum

```
/// <summary>
/// Indicates the result of a server operation.
/// </summary>
public enum ResultStatus
{
    /// <summary>
    /// The operation has successfully ended without any error/warning.
    /// </summary>
    Success,
    /// <summary>
    /// The operation has successfully ended with warnings.
    /// </summary>
    Warning,
    /// <summary>
    /// The operation has finihes with an error.
    /// </summary>
    Error
}
```

ResultInfo class

```
/// <summary>
/// Encapsulates the result of a server operation (such as document
/// import, export, etc.)
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ResultInfo
{
    /// <summary>
```

```
    /// The ResultStatus of the operation.
    /// </summary>
    [DataMember]
    public ResultStatus ResultStatus;
    /// <summary>
    /// Some textual information related to the operation that can be displayed
    /// to the user. This can be summary information on success or the main
    /// error message in can of an error. Can be null.
    /// </summary>
    [DataMember]
    public string MainMessage;
    /// Some detailed textual information related to the operation. This message is
    /// tipically not to be displayed to the user. E.g. contains the stack trace
    /// in case of an exception.
    [DataMember]
    public string DetailedMessage;
}
```

FileResultInfo class

```
/// <summary>
/// Encapsulates the result of a server operation whose output is a file.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class FileResultInfo : ResultInfo
{
    /// <summary>
    /// The Guid of the resulting file of the operation. It contains a valid
    /// value only if the ResultStatus member is Success or Warning.
    /// </summary>
    [DataMember]
    public Guid FileGuid;
}
```

ImportTranslationDocumentOptions class

```
/// <summary>
/// Describes the configuration options of importing a file as a document
/// into a server projects.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ImportTranslationDocumentOptions
{
    /// <summary>
    /// The guid of the file to importe. The file have to be uploaded
    /// before using IFileManagerService. The guid is returned by
    /// IFileManagerService.BeginChunkedFileUpload.
    /// </summary>
    [DataMember] public Guid FileGuid;
    /// <summary>
    /// The three+(two) letter code of the target languages the document
    /// is to be imported into. If null, the document is imported into all
    /// target languages of the project. If not null, only target languages
    /// that are subset of the target languages of the project can be specified.
    /// </summary>
    [DataMember] public string[] TargetLangCodes;
    /// <summary>
    /// The settings of the import in memoQ import filter XML format.
    ///
    /// When this member is set, FilterConfigResGuid must not be set.
    /// If neither this member nor FilterConfigResGuid is set, the file
    /// will be imported with default configuration matching the file
    /// extension.
    /// </summary>
    [DataMember] public string ImportSettingsXML;
    /// <summary>
    /// The guid of the filter configuration resource to be used during the
```

```
    /// import.
    ///
    /// When this member is set, ImportSettingsXML must not be set.
    /// If neither this member nor ImportSettingsXML is set, the file
    /// will be imported with default configuration matching the file
    /// extension.
    /// </summary>
    [DataMember] public Guid FilterConfigResGuid;
    /// <summary>
    /// Determines whether to import embedded images from the file. The
    /// images will appear as translatable documents. Embedded images
    /// can be imported from Office Open XML documents (docx, pptx, xlsx).
    /// </summary>
    [DataMember] public bool ImportEmbeddedImages;
    /// <summary>
    /// Determines whether to import embedded objects from the file.
    /// Embedded objects are for example an embedded Excel spreadsheet
    /// in a Word document. Supported embedded objects will appear as
    /// translatable documents. Embedded objects can be imported from
    /// Office Open XML documents (docx, pptx, xlsx). If the specified
    /// filter configuration does not specify the configuration of
    /// embedded objects, they will be imported with default
    /// configurations.
    /// </summary>
    [DataMember] public bool ImportEmbeddedObjects;
    /// <summary>
    /// An identifier of this document that is provided by the caller when
    /// and stored by memoQ. The value is not interpreted, only stored
    /// and used for example when signalling delivery through a callback
    /// to the external tool.
    /// </summary>
    [DataMember]
    public string ExternalDocumentId;
    /// <summary>
    /// The full path which should be set as the import path of the
    /// file. Since the files are uploaded via IFileManagerService,
    /// their original location is not retained during import. If this
    /// path is specified, this string will be saved as the import path
    /// of the document, and will be the bases for calculating the
    /// export path.
    /// </summary>
    [DataMember]
    public string PathToSetAsImportPath;
}
```

TranslationDocImportResultInfo class

```
/// <summary>
/// Encapsulates the result of a translation document import operation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocImportResultInfo : ResultInfo
{
    /// <summary>
    /// The array of Guids assigned by the system to the imported documents.
    /// There is one Guid for each target language the document is imported
    /// into.
    /// A document Guid uniquely identifies a document (of a specific
    /// target language) within  a server project.
    /// Has a valid (non empty Guid) value only if the import was
    /// successful (when the inherited ResultStatus is ResultStatus.Success or
    /// ResultStatus.Warning).
    /// </summary>
    [DataMember]
    public Guid[] DocumentGuids;
}
```

ReimportDocumentOptions class

```
/// <summary>
/// Encapsulates details of a document reimport action for a particular
/// source file.
/// </summary>
[DataContract( Namespace = "http://kilgray.com/memoqservices/2007" )]
public class ReimportDocumentOptions
{
    /// <summary>
    /// The guid of the file uploaded to the FileManagerService which is to
    /// be imported replacing documents in the project.
    /// </summary>
    [DataMember]
    public Guid FileGuid;
    /// <summary>
    /// The guids of the documents in the server projects which are to be
    /// replaces by the imported document. The documents should belong to
    /// different target languages. To replace all target language copies
    /// of a document all document guids must be specified.
    /// </summary>
    [DataMember]
    public Guid[] DocumentsToReplace;
}
```

BilingualDocFormat enum

```
/// <summary>
/// Represents bilingual document formats.
/// </summary>
public enum BilingualDocFormat
{
    /// <summary>
    /// The document format is XLIFF.
    /// </summary>
    XLIFF
    /// <summary>
    /// Two-column RTF document format.
    /// </summary>
    TwoColumnRTF
}
```

TranslationDocExportResultInfo class

```
/// <summary>
/// Encapsulates the result of a translation document export
/// operation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocExportResultInfo : FileResultInfo
{
    /// <summary>
    /// If the export is not possible, contains the the idices of segments
    /// (segment numbers) that have errors.
    /// </summary>
    [DataMember]
    public int[] ErrorSegmentIndices;
}
```

XliffBilingualExportOptions class

```
/// <summary>
/// Represents options for XLIFF bilingual export.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class XliffBilingualExportOptions
{
    /// <summary>
    /// If true, the exported XLIFF file contains the skeleton required for
    /// export into the original format. If true, SaveCompressed must be true.
    /// </summary>
    [DataMember]
    public bool IncludeSkeleton;
    /// <summary>
    /// If true, the exported XLIFF file is compressed.
    /// </summary>
    [DataMember]
    public bool SaveCompressed;
    /// <summary>
    /// Include full version history including all previous major versions.
    /// If true, SaveCompressed must be true too.
    /// </summary>
    [DataMember]
    public bool FullVersionHistory;
}
```

RtfBilingualExportOptions class

```
/// <summary>
/// Represents options for Trados-like RTF bilingual export.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class RtfBilingualExportOptions
{
    /// <summary>
    /// If true, 101% matches appear in the exported bilingual files as 100%
    /// matches.
    /// </summary>
    [DataMember]
    public bool SuppressContext;
    /// <summary>
    /// If true, only the tags are inserted as translation for empty segments
    /// during export.
    /// </summary>
    [DataMember]
    public bool SegmentedContextEmptyTranslation;
}
```

TwoColumnRtfBilingualExportOptions class

```
/// <summary>
/// Represents options for Two-column RTF bilingual export.
/// </summary>
[DataContract( Namespace = "http://kilgray.com/memoqservices/2007" )]
public class TwoColumnRtfBilingualExportOptions
{
    /// <summary>
    /// Export the comment of the segment into a new column.
    /// </summary>
    [DataMember]
    public bool ExportComment;
    /// <summary>
    /// Export the segment status with the match rate (if applicable) to a
    /// new column.
    /// </summary>
```

```
    [DataMember]
    public bool ExportSegmentStatus;
    /// <summary>
    /// Export two target columns.
    /// </summary>
    [DataMember]
    public bool ExportTwoTargetColumns;
    /// <summary>
    /// When exporting two target columns, leave the second column empty;
    /// if false, both target columns will contain the target segment.
    /// </summary>
    [DataMember]
    public bool SecondTargetColumnEmpty;
}
```

DocumentExportOptions class

```
/// <summary>
/// Represents options for exporting documents into their original format.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class DocumentExportOptions
{
    /// <summary>
    /// Copy source to target in case of empty target segments.
    /// The change is done only for the export, it does not affect the
    /// document itself.
    /// </summary>
    [DataMember]
    public bool CopySourceToEmptyTarget;
    /// <summary>
    /// Copy source to target if the row has tag errors. When false,
    /// tag errors are fixed by adding the missing tags to the target.
    /// The change is done only for the export, it does not affect the
    /// document itself.
    /// </summary>
    [DataMember]
    public bool RevertFaultyTargetsToSource;
    /// <summary>
    /// Copy source to target in case of unconfirmed rows.
    /// The change is done only for the export, it does not affect the
    /// document itself.
    /// </summary>
    [DataMember]
    public bool CopySourceToUnconfirmedRows;
    /// <summary>
    /// Multilingual document formats (imported using the Multilingual
    /// filters) are "unified" into a single result file. All siblings
    /// of the specified document (all document that originated from
    /// the same multilingual input file) are exported and unified,
    /// and a single result export is provided.
    /// If false, only the specified document is exported and multilingual
    /// siblings are ignored; the result file will contain the translation
    /// of this document and the original content of other languages.
    /// Applicable only to multilingual formats; has no effect otherwise.
    /// Default is true.
    /// </summary>
    [DataMember]
    public bool ExportAllMultilingualSiblings;
}
```

3.7.6 Resource handling

The **IServerProjectService** operations for resource handling and related entity classes are discussed in this chapter.

The operations and entity classes involved are the following:

- Assigning translation memories to a server project
  - IServerProjectService.SetProjectTMs2
  - ServerProjectTMAssignmentDetails
- Retrieving the list of translation memories assigned to a server project (new in version 4.5)
  - IServerProjectService.ListProjectTMs2
  - ServerProjectTMAssignmentsForTargetLang
  - TMInfo (defined of the *Translation memory* WS API)
- Assigning term bases to a server project
  - IServerProjectService.SetProjectTBs
  - IServerProjectService.SetProjectTBs2
- Retrieving the list of term bases assigned to a server project
  - IServerProjectService.ListProjectTBs
  - ServerProjectTBAssignments
  - TBInfo (defined of the *Term base* WS API)
- Assigning corpora to a server project
  - IServerProjectService.SetProjectCorpora
- Retrieving the list of corpora assigned to a server project
  - IServerProjectService.ListProjectCorpora
  - ServerProjectCorporaAssignments
  - CorpusInfo (defined of the *LiveDocs* WS API)
- Assigning light resources to a server project
  - IServerProjectService.SetProjectResourceAssignments
  - ServerProjectResourceAssignmentForResourceType
  - ServerProjectResourceAssignment
- Retrieving the list of light resources assigned to a server project
  - IServerProjectService.ListProjectResourceAssignments
  - IServerProjectService.ListProjectResourceAssignmentsForMultipleTypes
  - ServerProjectResourceAssignmentDetails

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
```

```
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    …
    /// <summary>
    /// Returns the current translation memory assignments of the specified
    /// project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="targetLangCodes">The target language codes for which the
    /// translation memory assignments are to be returned. If null, assignments
    /// for all target languages are returned.</param>
    /// <returns>
    /// The current translation memory assignments for the requested target
    /// languages.
    /// </returns>
    [OperationContract]
    ServerProjectTMAssignmentDetails[] ListProjectTMs2(Guid serverProjectGuid,
        string[] targetLangCodes);
    /// <summary>
    /// Sets the new translation memory assignments for the project (for a subset
    /// of the target languages of the project). Also sets the primary translation
    /// memories for these target languages.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="tmAssignments">Defines the new TM assignments for the
    /// project. Each object in this array defines the new assignment and
    /// primary TM for one target language. TM assignment for project target
    /// languages for which no ServerProjectTMAssignments object is
    /// included in the array is not changed.</param>
    /// <remarks>Translation memories can be created, published and deleted
    /// by the operations of the ITMService service interface.</remarks>
    [OperationContract]
    void SetProjectTMs2(Guid serverProjectGuid,
        ServerProjectTMAssignmentsForTargetLang[] tmAssignments);
    /// <summary>
    /// Assigns the term bases identifed by the tbGuids parameter to
    /// the specified project. Also sets the primary term base of the
    /// project. All existing term base assignments are deleted. The
    /// existing primary term base assignment is also deleted.
    /// Only those term bases can be assigned to the project, whose languages
    /// include the source and all target languages of the project.
    /// When matching languages lazy match is performed (e.g. a term base
    /// including language "eng" but not including "eng-US" can also be used
    /// with a project with language "eng-US"). However, if the
    /// TB and project languages do not match exactly, translators will not be
    /// able to add new entries to the TB working in the project).
    /// Remark: pre 6.2.15 versions of the API required exact language match:
    /// this has been changed to make the API more consistent with the memoQ
    /// application behavior).
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="tbGuids">The guids of the term bases to be
    /// assigned to the project.</param>
    /// <param name="primaryTB">The guid of the primary term base to
    /// be set for the project. This guid have to be included in the guid array
    /// of the tbGuids parameter. If tbGuids array is empty, set this to an
    /// empty Guid (Guid of full zeros).
    /// </param>
    /// <remarks>Term bases can be created, published and deleted
    /// by the operations of the ITBService service interface.</remarks>
    [Obsolete]
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    void SetProjectTBs(Guid serverProjectGuid, Guid[] tbGuids, Guid primaryTB);
    /// <summary>
    /// Assigns the term bases identifed by the tbGuids parameter to
```

```
/// the specified project. Also sets the ranks of term bases of the
/// project based on the order of the TB guids. The TB with rank 1 will be
/// the primary term base. All existing term base assignments are deleted.
/// Only those term bases can be assigned to the project, whose languages
/// include the source and all target languages of the project. When matching
/// languages lazy match is performed (e.g. a term base including language
/// "eng" but not including "eng-US" can also be used with a project with
/// language "eng-US"). However, if the TB and project languages do not match
/// exactly, translators will not be able to add new entries to the TB working
/// in the project). Remark: pre 6.2.15 versions of the API required exact
/// language match: this has been changed to make the API more consistent
/// with the memoQ application behavior).
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="tbs">The term bases to be assigned to the project.</param>
/// <remarks>Term bases can be created, published and deleted
/// by the operations of the ITBService service interface.</remarks>
[OperationContract]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
void SetProjectTBs2(Guid serverProjectGuid, Guid[] tbGuids);
/// <summary>
/// Returns the current term base assignments of the specified project.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <returns>The current term base assignments.</returns>
[OperationContract]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
ServerProjectTBAssignments ListProjectTBs(Guid serverProjectGuid);
/// <summary>
/// Assigns the corpora identifed by the corporaGuids parameter to
/// the specified project. All existing corpora assignments are deleted.
/// Only those corpora can be assigned to the project, whose languages
/// include the source language of the project.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="tbGuids">The guids of the corpora to be
/// assigned to the project.</param>
/// </param>
/// <remarks>Corpora can be created, published and deleted
/// by the operations of the ILiveDocService service interface.</remarks>
[OperationContract]
void SetProjectCorpora(Guid serverProjectGuid, Guid[] corporaGuids);
/// <summary>
/// Returns the current corpora assignments of the specified project.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <returns>The current corpora assignments.</returns>
[OperationContract]
ServerProjectCorporaAssignments ListProjectCorpora(Guid serverProjectGuid);
/// <summary>
/// Sets the new light resource assignments for the project for the specified
/// resource types. See the documentation of the
/// ServerProjectResourceAssignment  member of class
/// ServerProjectResourceAssignmentForResourceType to find out what rules
/// apply to the assignment of different resource types.
/// </summary>
/// <param name="serverProjectGuid">The guid of the server project.</param>
/// <param name="assignments">Defines the light resource assignments for the
/// project. Each object in this array defines the new assignment for one
/// resource type. Resource assignment for resource types for which no
/// ServerProjectResourceAssignmentForResourceType object is included in the
/// array is not changed.</param>
/// <remarks>Light resources can be created, imported and deleted
/// by the operations of the IResourceService service interface.</remarks>
[OperationContract]
void SetProjectResourceAssignments(Guid serverProjectGuid,
    ServerProjectResourceAssignmentForResourceType[] assignments);
/// <summary>
```

```
    /// Lists the project (light) resource assignments for the specified resource
    /// type. If you would like to get the list of resource assignments for
    /// more than one resource types in one step you can use
    /// ListProjectResourceAssignmentsForMultipleTypes instead.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="resourceType">The type of the resource.</param>
    /// <returns>The current resource assignments for the specified resource
    /// type.</returns>
    [OperationContract]
    ServerProjectResourceAssignmentDetails[] ListProjectResourceAssignments
        (Guid serverProjectGuid, ResourceType resourceType);
    /// <summary>
    /// Lists the project (light) resource assignments for the specified resource
    /// types.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="resourceTypes">An array of resource types for which the
    /// assignments are to be returned.</param>
    /// <returns>The current resource assignments for the specified resource
    /// types. An array of ServerProjectResourceAssignmentDetails objects is
    /// returned for each resource type requested in the resourceTypes
    /// parameter in the same order as requested.
    /// </returns>
    [OperationContract]
    ServerProjectResourceAssignmentDetails[][]
        ListProjectResourceAssignmentsForMultipleTypes
        (Guid serverProjectGuid, ResourceType[] resourceTypes);
    …
}
```

ServerProjectTMAssignmentDetails class

```
/// <summary>
/// Encapsulates server project translation memory assignments for
/// a specific target language.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectTMAssignmentDetails
{
    /// <summary>
    /// The three+(two) letter code of the target language of the project.
    /// (such as fre, eng, eng-US) to which the TM assignments are returned
    /// in the "TMs" member of this object.
    /// </summary>
    [DataMember]
    public string TargetLangCode;
    /// <summary>
    /// The list of TMInfo objects each describing a TM assigned to the
    /// project for target language specified by TargetLangCode.
    /// </summary>
    [DataMember]
    public TMInfo[] TMs;
    /// <summary>
    /// The Guid of the working (primary) TM assigned to the project
    /// for the target language specified by TargetLangCode. It is
    /// Guid.Empty if the project does not contain woring (primary)
    /// TM for the target language specified by TargetLangCode.
    /// </summary>
    [DataMember]
    public Guid PrimaryTMGuid;
}
```

ServerProjectTMAssignmentsForTargetLang class

```
/// <summary>
/// Encapsulates server project translation memory assignments for
/// a specific target language.
```

```
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectTMAssignmentsForTargetLang
{
    /// <summary>
    /// The three+(two) letter code of the target language of the project.
    /// (such as fre, eng, eng-US) for which the TMs are assigned to.
    /// Existing TM assignments for this project target language are
    /// deleted.
    /// </summary>
    [DataMember]
    public string TargetLangCode;
    /// <summary>
    /// The Guids of the TMs assigned to the project (for the
    /// target language specified by TargetLangCode). Can not be null.
    //  Only those translation memories can be assigned to the project, whose
    /// source language matches the source language of the project, and whose
    /// target language matches the TargetLangCode member of this object.
    /// When matching languages loose match is performed (e.g. a translation
    /// memory with source language "eng" can be used even if the source language
    /// of the project is "eng-US", and the same stands for target languages).
    /// </summary>
    [DataMember]
    public Guid[] TMGuids;
    /// <summary>
    /// The Guid of the working (primary) TM assigned to the project
    /// for the target language specified by TargetLangCode. It has
    /// to be included in TMGuids. Or, if TMGuids is an empty array,
    /// use an empty Guid (containing zeros only).
    /// </summary>
    [DataMember]
    public Guid PrimaryTMGuid;
    /// <summary>
    /// The Guid of the master TM assigned to the project for the
    /// target language specified by TargetLangCode. It has to be
    /// included in TMGuids. Or, if TMGuids is an empty array,
    /// use an empty Guid (containing zeros only). The primary TM
    /// will be the master as well if this is an empty guid.
    /// </summary>
    [DataMember]
    public Guid MasterTMGuid;
}
```

ServerProjectTBAssignments class

```
/// <summary>
/// Encapsulates server project term base assignments.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectTBAssignments
{
    /// <summary>
    /// The list of term bases assigned to the project.
    /// </summary>
    [DataMember]
    public TBInfo[] TBs;
    /// <summary>
    /// The guid of the primary term base of the project, or an
    /// empty Guid (Guid of zeros), if no term base is assigned to the
    /// project.
    /// </summary>
    [DataMember]
    public Guid PrimaryTBGuid;
}
```

ServerProjectCorporaAssignments class

```
/// <summary>
```

```
/// Encapsulates server project corpora assignments.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectCorporaAssignments
{
    /// <summary>
    /// The list of corpora assigned to the project.
    /// </summary>
    [DataMember]
    public CorpusInfo[] Corpora;
}
```

ServerProjectResourceAssignment class

```
/// <summary>
/// Represents the assignment of a single resource to the involved server
/// project.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectResourceAssignment
{
    /// <summary>
    /// The guid of the resource assigned to the project.
    /// </summary>
    [DataMember]
    public Guid ResourceGuid;
    /// <summary>
    /// Indicates if this resource is assigned as a primary resource to the
    /// project. Currently non-translatable and ignore list resources use
    /// primary resources (see enum ResourceType for more information).
    /// For other resource types its value is ignored.
    /// </summary>
    [DataMember]
    public bool Primary;
    /// <summary>
    /// Extra parameter for the assignment, has different meaning for different
    /// resource  types:
    /// - AutoCorrect, NonTran, FilterConfigs, KeyboardShortcuts: Not used, has
    ///   to be null.
    /// - For other resource types see documentation of the
    ///   ServerProjectResourceAssignment member of class
    ///   ServerProjectResourceAssignmentForResourceType
    /// </summary>
    [DataMember]
    public string ObjectId;
}
```

ServerProjectResourceAssignmentForResourceType class

```
/// <summary>
/// Represents assignments of light resources of a specific resource type to
/// a server project.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectResourceAssignmentForResourceType
{
    /// <summary>
    /// The type of the resource.
    /// </summary>
    [DataMember]
    public ResourceType ResourceType;

    /// <summary>
    /// An array of ServerProjectResourceAssignment objects, each representing
    /// the assignment of a single resource to the involved server project. All
    /// existing assignments for resource type identified by the ResourceType
    /// member are deleted before applying the new set.
    /// Each resource has to identify a resource of type defined by the
```

```
/// ResourceType member. The rules for different resource types are the
/// following:
/// - AutoCorrect, FilterConfigs, KeyboardShortcuts: can not be assigned to
///   server projects.
/// - AutoTrans: 0..N can be assigned to each target language of a server
///   project (independently). The ObjectId has to have the three+(two)
///   letter code of the project target language to which the specific
///   resource is to be assigned to.
/// - NonTrans: 0..N can be assigned to a server project (shared for all
///   target languages of the project. ObjectId has to be null for the
///   assignment.
///   If there is at least one assigned, one of them has to be marked as
///   primary.
/// - IgnoreLists: 0..N can be assigned independently to each target
///   language of a server project. The ObjectId has to be the three+(two)
///   letter code of the project target language to which the specific
///   resource is to be assigned to. Only IgnoreList resources with
///   language exact matching the target language (ObjectId) of the
///   assignment can be assigned (e.g. no resource with lang "eng" can be
///   assigned to target lang "eng-US").
///   If there is at least one assigned to a target language, one of them
///   has to be marked as primary (one for each target lang).
/// - SegRules: Exactly one has to be assigned to a project, which defines
///   the segmentation rules for the source language of the project.
///   ObjectId has to be null for the assignment. Only SegRule resources
///   with language exact matching the source language of the project
///   can be assigned (e.g. no resource with lang "eng" can be assigned to
///   project with source lang "eng-US").
///   The default segmentation rule deployed with memoQ is automatically
///   assigned to the server project when the project is created. To reset
///   the assignment to this deployed default later on (if it has been
///   changed) use an empty (full zero) guid resource ID.
/// - TMSettings: Exactly one has to be assigned to a project, which
///   defines the default TM settings for the project.  The ObjectId for
///   this project default assignment has to be null. Optionaly one
///   TMSettings resource can be assigned for each TM assigned to the
///   project (can be different for each one), that overrides the project
///   default for the specific TM. The ObjectId has to the the guid of
///   the TM for which the setting is to be applied.
///   The default TMSetting resource deployed with memoQ is automatically
///   assigned to the server project to define the project default TM
///   settings when the project is created. To reset the assignment
///   to this deployed default later on (if it has been changed) use an
///   empty (full zero) guid resource ID. To remove the override setting
///   for a specific TM simply do not provide any
///   ServerProjectResourceAssignment object for the specific TM when
///   setting the new assignment for resource type TMSettings.
/// - PathRules: Exactly two have to be assigned to a project: one
///   defining file export path rule, and another one defining the folder
///   export path rule. Assigning the file export use ObjectId "File",
///   assigning the folder rule use ObjectId "Folder".
///   The default file and folder rule deployed with memoQ are automatically
///   assigned to the server project when the project is created. To reset
///   the assignment to these deployed defaults later on (if it has been
///   changed) use an empty (full zero) guid resource ID.
/// - QASettings: Exactly one has to be assigned to each target language
///   of the project (independently). The ObjectId has to be the
///   three+(two) be the letter code of the project target language to
///   which the specific resource is to be assigned to.
///   The default QASettings resource deployed with memoQ is automatically
///   assigned to each target language of the server project when the
///   project is created. To reset the assignment to this deployed default
///   later on (if it has been changed) use an empty (full zero) guid
///   resource ID.
/// - LiveDocsSettings: Exactly one has to be assigned to a project, which
///   defines the default LiveDocs settings for the project. The ObjectId
///   for this project default assignment has to be null. Optionaly one
///   LiveDocsSettings resource can be assigned for each LiveDocs assigned
```

```
    ///   to the project (can be different for each one), that overrides the
    ///   project default for the specific LiveDocs corpus. The ObjectId has
    ///   to the the guid of the LiveDocs corpus for which the setting is to
    ///   be applied. The default LiveDocsSettings resource deployed with
    ///   memoQ is automatically assigned to the server project to define
    ///   the project default LiveDocs settings when the project is created.
    ///   To reset the assignment to this deployed default later on (if it
    ///   has been changed) use an empty (full zero) guid resource ID. To
    ///   remove the override setting for a specific LiveDocs corpus simply
    ///   do not provide any  ServerProjectResourceAssignment object for the
    ///   specific LiveDocs corpus when setting the new assignment for
    ///   resource type LiveDocsSettings.
    /// - LQA: a single one can be assigned to a server project (shared for
    ///   all target languages of the project. ObjectId has to be null for
    ///   the assignment.
    /// </summary>
    [DataMember]
    public ServerProjectResourceAssignment[] ServerProjectResourceAssignment;
}
```

ServerProjectResourceAssignmentDetails class

```
/// <summary>
/// Represents the assignment of a single resource to the involved server
/// project with detailed information about the assigned resource.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectResourceAssignmentDetails
{
    /// <summary>
    /// Holds information about the assigned resource. To find out exactly
    /// which derived class is returned here for a resource type, see
    /// the "Described by class" section of the documentation of the appropriate
    /// "ResourceType" enum member (e.g.ResourceInfoWithLang for SegRules).
    /// </summary>
    [DataMember]
    public LightResourceInfo ResourceInfo;
    /// <summary>
    /// Indicates if this resource is assigned as a primary resource to the
    /// project. Currently non-translatable and ignore list resources use
    /// primary resources (see enum ResourceType for more information).
    /// For other resource types its value is false.
    /// </summary>
    [DataMember]
    public bool Primary;
    /// <summary>
    /// Extra parameter for the assignment, has different meaning for different
    /// resource  types:
    /// - AutoCorrect, NonTran, FilterConfigs, KeyboardShortcuts: Not used.
    /// - For other resource types see documentation of the
    ///   ServerProjectResourceAssignment member of class
    ///   ServerProjectResourceAssignmentForResourceType.
    /// </summary>
    [DataMember]
    public string ObjectId;
}
```

3.7.7 User management

The **IServerProjectService** operations for user management and related entity classes are discussed in this chapter.

The operations and entity classes involved are the following:

- Assigning users to a server project with their project roles specified
  - IServerProjectService.SetProjectUsers
  - ServerProjectUserInfo
  - ServerProjectRoles
- Retrieving the list of users assigned to a server project
  - IServerProjectService.ListProjectUsers
  - ServerProjectUserInfoHeader
  - ServerProjectRoles
  - UserInfo (defined by the *Security* WS API)

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    …
    /// <summary>
    /// Assigns the users identifed by the userInfos parameter to the specified
    /// project. The ServerProjectUserInfo objects identify the user and his/her
    /// role in the server project.
    /// All existing user assignments are deleted.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="userInfos">The new list of server project user assignments.
    /// </param>
    [OperationContract]
    void SetProjectUsers(Guid serverProjectGuid, ServerProjectUserInfo[]
        userInfos);
    /// <summary>
    /// Returns the current user assignments of the specified project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <returns>The current user assignments.</returns>
    [OperationContract]
    ServerProjectUserInfoHeader[] ListProjectUsers(Guid serverProjectGuid);
    …
}
```

ServerProjectUserInfo class

```
/// <summary>
/// Represents a server project user assignment. Only the guid of the user
/// is included.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class ServerProjectUserInfo
{
    /// <summary>
    /// The guid of the user assigned to the project.
    /// </summary>
    [DataMember]
    public Guid UserGuid;
    /// <summary>
    /// The roles of the user in the project.
```

```
    /// </summary>
    [DataMember]
    public ServerProjectRoles ProjectRoles;
    /// <summary>
    /// Indicates if the user can take an ELM license so the she can user her
    /// memoQ client to work on the project.
    /// </summary>
    [DataMember]
    public bool PermForLicense;
}
```

ServerProjectRoles class

```
/// <summary>
/// Represents user roles in server projects.
/// A user can have any combination of the ProjectManager and Terminologist
/// project roles.
/// </summary>
public partial class ServerProjectRoles
{
    /// <summary>
    /// True if the user is in the ProjectManager project role.
    /// </summary>
    public bool ProjectManager;
    /// <summary>
    /// True if the user is in the Terminologist project role.
    /// </summary>
    public bool Terminologist;
}
```

ServerProjectUserInfoHeader class

```
/// <summary>
/// Represents a server project user assignment. All information about
/// the user is included.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectUserInfoHeader
{
    /// <summary>
    /// Information about the user as defined by the Security API of the
    /// MemoQ Server WS API.
    /// </summary>
    [DataMember]
    public UserInfo User;
    /// <summary>
    /// The roles of the user in the project.
    /// </summary>
    [DataMember]
    public ServerProjectRoles ProjectRoles;
    /// <summary>
    /// Indicates if the user can take an ELM license so the she can user her
    /// memoQ client to work on the project.
    /// </summary>
    [DataMember]
    public bool PermForLicense;
}
```

3.7.8 Translation document management

The **IServerProjectService** operations for user management and related entity classes are discussed in this chapter.

The operations and entity classes involved are the following:

- Retrieving the list of translation documents imported to a server project
  - IServerProjectService.ListProjectTranslationDocuments
  - ServerProjectTranslationDocInfo
  - IServerProjectService.ListProjectTranslationDocuments2
  - ListServerProjectTranslationDocument2Options
  - ServerProjectTranslationDocInfo2
  - DocumentStatus
  - WorkflowStatus
  - TranslationDocumentUserRoleAssignmentDetails
  - UserInfoHeader
- Delete a translation document from the server
  - IServerProjectService.DeleteTranslationDocument
- Changing the workflow status of documents;
  - IServerProjectService.SetDocumentWorkflowStatus
  - ServerProjectTranslationDocumentWorkflowStatusChange
  - WorkflowStatus
- Delivering documents/returning documents to the previous actor
  - IServerProjectService.DeliverDocument
  - DeliverDocumentRequest

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    …
    /// <summary>
    /// Returns the current translation documents of the specified project.
    /// </summary>
    /// <remarks>The result does not contain information about the FirstAccept,
    /// GroupSourcing and subvendor group assignments. You can use the function
    /// <see cref="ListProjectTranslationDocuments2"/>, or you can use the
    /// function <see cref="ListTranslationDocumentAssignments"/> if you would
    /// like to get these information as well.
    /// </remarks>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <returns>The translation documents of the project.</returns>
    [OperationContract]
    ServerProjectTranslationDocInfo[] ListProjectTranslationDocuments(
      Guid serverProjectGuid);
    /// <summary>
    /// Returns the current translation documents of the specified project.
    /// The result contains the details of the assignments if the field
    /// FillInAssignmentInformation is true in the options parameter. Setting this
    /// to false has better performance. It is possible to query the assignments
    /// in a separate step using the
```

```
    /// <see cref="ListTranslationDocumentAssignments"/> operation.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="options">The listing options.</param>
    /// <returns>The translation documents of the project.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    ServerProjectTranslationDocInfo2[] ListProjectTranslationDocuments2(
        Guid serverProjectGuid,
        ListServerProjectTranslationDocument2Options options);
    /// <summary>
    /// Deletes the specified translation document.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.
    /// </param>
    /// <param name="documentGuid">The Guid of the document, that uniquely
    /// identifies a document within a server project.
    /// </param>
    [OperationContract]
    void DeleteTranslationDocument(Guid serverProjectGuid, Guid documentGuid);
     /// <summary>
    /// Changes the workflow status of documents in the server project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="workflowChanges">The workflow status changes to apply.
    /// Each item represents a change for a single document. Workflow status
    /// of documents not listed in this list are nnot changed.</param>
    [OperationContract]
    void SetDocumentWorkflowStatus( Guid serverProjectGuid,
        ServerProjectTranslationDocumentWorkflowStatusChange[] workflowChanges );
    /// <summary>
    /// Delivers a document (returns them to the previous actor or to the next
    /// person in the assignment chain) by a user to whom the document is
    /// assigned. Delivery requires all segments to be confirmed/proofread.
    /// Throws exception if the document is not assigned to the delivering
    /// user or if the document cannot be delivered because not all segments
    /// are confirmed/proofread. Also throws exception if there is no previos
    /// actor to whom the document is to be returned. Throws if the project's
    /// PreventDeliveryOnQAError flag is true and the document contains QA errors.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="deliveryOption">The delivery options.</param>
    /// <returns>The new workflow status of the document.</returns>
    [OperationContract]
    WorkflowStatus DeliverDocument(Guid serverProjectGuid,
        DeliverDocumentRequest deliveryOption);
    …
}
```

Several operations expect the Guid of a translation document as parameter: when a file is imported into a project, it receives a Guid (called „document Guid") that uniquely identifies the document within the project. When a document is imported into multiple target languages, it receives a separate Guid for each target language. Please note that two documents within two separate projects may have the same document Guid (e.g. when a document is exported as a bilingual document and then imported into two separate projects: the imported documents will have the same Guid as the exported one). The document Guid together with the server project Guid is unique within the system.

Next, a few thoughts about the naming of translation documents. Translation document names for a specific target language are unique within a project. The document name is generated during document import: this is the name of the uploaded file (the `fileName` parameter of `IFileManagerService.BeginChunkedFileUpload`) without the extension. E.g.: "mydoc" for the uploaded doc "mydoc.txt". If a file is imported into a

project with a name that conflicts with the name of an existing project document, then a number is appended to the name of the newly imported document. E.g. if documents with name "mydoc" and "mydoc1" exist within a project, then the importing of a file with name "mydoc.rtf" will have the "mydoc2" name.

ServerProjectTranslationDocInfo class

```
/// <summary>
/// Represents information about a server project translation document.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectTranslationDocInfo
{
    /// <summary>
    /// The Guid of the document that uniquely identifies the document within
    /// a server project.
    /// Two documents within two separate projects can have the same Guid:
    /// e.g. exporting a bilingual doc from projectA, and
    /// then importing it into projectB the two documents will have the same
    /// Guid (it is preserved during a bilingual export/import).
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The name of the document. Unique within a project for a specific target
    /// language. This is the name
    /// of the uploaded file without the extension. E.g.: "mydoc" for the
    /// uploaded doc "mydoc.txt". If a file is imported into a project
    /// with a name that conflicts with the name of an existing project document,
    /// then a number is appended to the name of the newly imported document.
    /// E.g. if documents with name "mydoc" and "mydoc1" exist within a project,
    /// then the importing of a file with name "mydoc.rtf" will have the
    /// "mydoc2" name.
    /// </summary>
    [DataMember]
    public string DocumentName;
    /// <summary>
    /// The three+(two) letter code of the target language of the document.
    /// (such as fre, eng, eng-US).
    /// </summary>
    [DataMember]
    public string TargetLangCode;
    /// <summary>
    /// The users assigned to the translation document with their document
    /// assignment role and deadline.
    /// </summary>
    /// <remarks>It does not contain information about the FirstAccept,
    /// GroupSourcing and subvendor group assignments. It is covered by
    /// the.<see cref="ServerProjectInfo2"/> class.</remarks>
    [DataMember]
    public TranslationDocumentUserRoleAssignmentDetails[] UserAssignments;
    /// <summary>
    /// The total number of segments of all documents of the server project,
    /// including the number of locked segments.
    /// </summary>
    [DataMember]
    public int TotalSegmentCount;
    /// <summary>
    /// The total number of confirmed segments of all documents of the server
    /// project.
    /// </summary>
    [DataMember]
    public int ConfirmedSegmentCount;
    /// <summary>
    /// The total number of reviewer 1 confirmed segments of all documents
```

```
/// of the server project.
/// </summary>
[DataMember]
public int Reviewer1ConfirmedSegmentCount;
/// <summary>
/// The total number of proofred segments of all documents of the server
/// project.
/// </summary>
[DataMember]
public int ProofreadSegmentCount;
/// <summary>
/// The total number of locked segments of all documents of the server
/// project.
/// </summary>
[DataMember]
public int LockedSegmentCount;
/// <summary>
/// The total number of characters of all documents of the server project,
/// including the number of characters in locked segments.
/// </summary>
[DataMember]
public int TotalCharacterCount;
/// <summary>
/// The total number of characters in confirmed segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int ConfirmedCharacterCount;
/// <summary>
/// The total number of characters in reviewer 1 confirmed segments of all
/// documents of the server project.
/// </summary>
[DataMember]
public int Reviewer1ConfirmedCharacterCount;
/// <summary>
/// The total number of characters in proofred segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int ProofreadCharacterCount;
/// <summary>
/// The total number of characters in locked segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int LockedCharacterCount;
/// <summary>
/// The total number of words of all documents of the server project,
/// including the number of words in locked segments.
/// </summary>
[DataMember]
public int TotalWordCount;
/// <summary>
/// The total number of words in confirmed segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int ConfirmedWordCount;
/// <summary>
/// The total number of words in reviewer 1 confirmed segments of all
/// documents of the server project.
/// </summary>
[DataMember]
public int Reviewer1ConfirmedWordCount;
/// <summary>
/// The total number of words in proofred segments of all documents
/// of the server project.
/// </summary>
```

```
[DataMember]
public int ProofreadWordCount;
/// <summary>
/// The total number of words in locked segments of all documents
/// of the server project.
/// </summary>
[DataMember]
public int LockedWordCount;
/// <summary>
/// The status of the document.
/// </summary>
[DataMember]
public DocumentStatus DocumentStatus;
/// <summary>
/// The workflow status of the document.
/// </summary>
[DataMember]
public WorkflowStatus WorkflowStatus;
/// <summary>
/// The documents current major version number if the document has recorded
/// version history; -1 otherwise.
/// </summary>
[DataMember]
public int MajorVersion;
/// <summary>
/// The documents current minor version number if the document has recorded
/// version history; -1 otherwise.
/// </summary>
[DataMember]
public int MinorVersion;
/// <summary>
/// The path the document was imported from.
/// </summary>
[DataMember]
public string ImportPath;
/// <summary>
/// The default export path of the document. The path is determined by the
/// Export Path rules associated to the project at the time of document
/// import. The value is null if the document was imported through the
/// webservice.
/// </summary>
[DataMember]
public string ExportPath;
/// <summary>
/// True if the document is an image file, false otherwise. Images can be
/// imported into projects and translated, but they require a transcription
/// first. Images are exported into an image localization package and
/// after localization the package is imported back (see ServerProjectService.
/// CreateImageLocalizationPack).
/// </summary>
[DataMember]
public string IsImage;
/// <summary>
/// Contains the unique identifier of the document which is the "parent"
/// document of this one. When embedded objects are imported (e.g. an
/// Excel spreadsheet embedded in a Word file), these documents are
/// losely connected. Embedded objects are treated as separate documents,
/// but certain operations link them to their parent (e.g. removing
/// a document will remove all of its "children" too). This member
/// contains the id of the parent document for child documents; e.g.
/// the id of the Word document if this document is the embedded
/// spredsheet in it. Value is empty Guid (all zeros) for documents
/// without a parent (most documents which are not embedded objects).
/// </summary>
[DataMember]
public Guid ParentDocumentId;
/// <summary>
/// The memoQWebTrans URL to open and edit this document. This URL is filled
```

```
    /// only when (1) the server has webTrans license, (2) memoQWebTrans URL is
    /// set in memoQ Server, and (3) the project is enabled for web access. The
    /// value is null or empty string otherwise..
    /// </summary>
    [DataMember]
    public string WebTransUrl;
    /// <summary>
    /// An identifier of this document which was provided by the caller when
    /// importing the document. The value is not created by memoQ; it comes
    /// from the caller of the WS API import. Null when not specified during
    /// import, or document not imported via the WS API.
    /// </summary>
    [DataMember]
    public string ExternalDocumentId;
}
```

ListServerProjectTranslationDocument2Options class

```
/// <summary>
/// Encapsulates the listing options of the server project
/// translation documents.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class ListServerProjectTranslationDocument2Options
{
    /// <summary>
    /// Indicates whether the assignment information field should
    /// be filled in or not.
    /// </summary>
    [DataMember]
    public bool FillInAssignmentInformation;
}
```

ServerProjectTranslationDocInfo2 class

```
/// <summary>
/// Encapsulates information about a server project translation document.
/// It contains detailed information about the FirstAccept, GroupSourcing
/// and subvendor assignments as well.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectTranslationDocInfo2
{
    /// <summary>
    /// The Guid of the document that uniquely identifies the document within
    /// a server project.
    /// Two documents within two separate projects can have the same Guid:
    /// e.g. exporting a bilingual doc from projectA, and
    /// then importing it into projectB the two documents will have the same
    /// Guid (it is preserved during a bilingual export/import).
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The name of the document. Unique within a project for a specific target
    /// language. This is the name
    /// of the uploaded file without the extension. E.g.: "mydoc" for the
    /// uploaded doc "mydoc.txt". If a file is imported into a project
    /// with a name that conflicts with the name of an existing project document,
    /// then a number is appended to the name of the newly imported document.
    /// E.g. if documents with name "mydoc" and "mydoc1" exist within a project,
    /// then the importing of a file with name "mydoc.rtf" will have the
    /// "mydoc2" name.
    /// </summary>
    [DataMember]
    public string DocumentName;
    /// <summary>
    /// The three+(two) letter code of the target language of the document.
```

```
    /// (such as fre, eng, eng-US).
    /// </summary>
    [DataMember]
    public string TargetLangCode;
    /// <summary>
    /// The users/user groups/subvendor groups assigned to the translation
document.
    /// </summary>
    [DataMember]
    public TranslationDocumentDetailedAssignmentInfo[] UserAssignments;
    /// <summary>
    /// The total number of segments in the document, including the number
    /// of locked segments.
    /// </summary>
    [DataMember]
    public int TotalSegmentCount;
    /// <summary>
    /// The total number of confirmed segments in the document.
    /// </summary>
    [DataMember]
    public int ConfirmedSegmentCount;
    /// <summary>
    /// The total number of reviewer1 confirmed segments in the document.
    /// </summary>
    [DataMember]
    public int Reviewer1ConfirmedSegmentCount;
    /// <summary>
    /// The total number of proofred segments in the document.
    /// </summary>
    [DataMember]
    public int ProofreadSegmentCount;
    /// <summary>
    /// The total number of locked segments in the document.
    /// </summary>
    [DataMember]
    public int LockedSegmentCount;
    /// <summary>
    /// The total number of characters in the document, including the number
    /// of characters in locked segments.
    /// </summary>
    [DataMember]
    public int TotalCharacterCount;
    /// <summary>
    /// The total number of characters in confirmed segments of the document.
    /// </summary>
    [DataMember]
    public int ConfirmedCharacterCount;
    /// <summary>
    /// The total number of characters in reviewer 1 confirmed segments of the
document.
    /// </summary>
    [DataMember]
    public int Reviewer1ConfirmedCharacterCount;
    /// <summary>
    /// The total number of characters in proofred segments of the document.
    /// </summary>
    [DataMember]
    public int ProofreadCharacterCount;
    /// <summary>
    /// The total number of characters in locked segments of the document.
    /// </summary>
    [DataMember]
    public int LockedCharacterCount;
    /// <summary>
    /// The total number of words in the document, including the number of
    /// words in locked segments.
    /// </summary>
    [DataMember]
```

```
public int TotalWordCount;
/// <summary>
/// The total number of words in confirmed segments of the document.
/// </summary>
[DataMember]
public int ConfirmedWordCount;
/// <summary>
/// The total number of words in reviewer 1 confirmed segments of the document.
/// </summary>
[DataMember]
public int Reviewer1ConfirmedWordCount;
/// <summary>
/// The total number of words in proofred segments of the document.
/// </summary>
[DataMember]
public int ProofreadWordCount;
/// <summary>
/// The total number of words in locked segments of the document.
/// </summary>
[DataMember]
public int LockedWordCount;
/// <summary>
/// The status of the document.
/// </summary>
[DataMember]
public DocumentStatus DocumentStatus;
/// <summary>
/// The workflow status of the document.
/// </summary>
[DataMember]
public WorkflowStatus WorkflowStatus;
/// <summary>
/// The documents current major version number if the document has recorded
/// version history; -1 otherwise.
/// </summary>
[DataMember]
public int MajorVersion;
/// <summary>
/// The documents current minor version number if the document has recorded
/// version history; -1 otherwise.
/// </summary>
[DataMember]
public int MinorVersion;
/// <summary>
/// The path the document was imported from.
/// </summary>
[DataMember]
public string ImportPath;
/// <summary>
/// The default export path of the document. The path is determined by the
/// Export Path rules associated to the project at the time of document
/// import. The value is null if the document was imported through the
/// webservice.
/// </summary>
[DataMember]
public string ExportPath;
/// <summary>
/// True if the document is an image file, false otherwise. Images can be
/// imported into projects and translated, but they require a transcription
/// first. Images are exported into an image localization package and
/// after localization the package is imported back (see ServerProjectService.
/// CreateImageLocalizationPack).
/// </summary>
[DataMember]
public bool IsImage;
/// <summary>
/// Contains the unique identifier of the document which is the "parent"
/// document of this one. When embedded objects are imported (e.g. an
```

```
    /// Excel spreadsheet embedded in a Word file), these documents are
    /// losely connected. Embedded objects are treated as separate documents,
    /// but certain operations link them to their parent (e.g. removing
    /// a document will remove all of its "children" too). This member
    /// contains the id of the parent document for child documents; e.g.
    /// the id of the Word document if this document is the embedded
    /// spredsheet in it. Value is empty Guid (all zeros) for documents
    /// without a parent (most documents which are not embedded objects).
    /// </summary>
    [DataMember]
    public Guid ParentDocumentId;
    /// <summary>
    /// The memoQWebTrans URL to open and edit this document. This URL is filled
    /// only when (1) the server has webTrans license, (2) memoQWebTrans URL is
    /// set in memoQ Server, and (3) the project is enabled for web access. The
    /// value is null or empty string otherwise..
    /// </summary>
    [DataMember]
    public string WebTransUrl;
    /// <summary>
    /// An identifier of this document which was provided by the caller when
    /// importing the document. The value is not created by memoQ; it comes
    /// from the caller of the WS API import. Null when not specified during
    /// import, or document not imported via the WS API.
    /// </summary>
    [DataMember]
    public string ExternalDocumentId;
}
```

DocumentStatus enum

```
/// <summary>
/// Translation status of the document.
/// </summary>
public enum DocumentStatus
{
    /// <summary>
    /// Translation is in progress. There is at least one not confirmed segment
    /// in the document.
    /// </summary>
    TranslationInProgress = 0,
    /// <summary>
    /// The translation of the document is complete and the translator has
    /// delivered the translation.
    /// </summary>
    TranslationFinished = 1,
    /// <summary>
    /// The proofreading of the document is complete, and the proofreader has
    /// delivered the translation.
    /// </summary>
    ProofreadingFinished = 2
}
```

WorkflowStatus enum

```
/// <summary>
/// Workflow status of the document.
/// </summary>
public enum WorkflowStatus
{
    /// <summary>
    /// Initial workflow state, or workflow state when it is not
    /// defined.
    /// </summary>
    Unknown = 0,
    /// <summary>
    /// The Translator is the next actor of the document but has
```

```
    /// not started working on it yet.
    /// </summary>
    TranslationNotStarted = 1,
    /// <summary>
    /// The Translator has started working on the document.
    /// </summary>
    TranslationInProgress = 2,
    /// <summary>
    /// The Translator has finished with the document, and it
    /// is now in the hands of the Review1 who has not started
    /// working with the document yet.
    /// </summary>
    Review1NotStarted = 3,
    /// <summary>
    /// The Reviewer1 has started working on the document.
    /// </summary>
    Review1InProgress = 4,
    /// <summary>
    /// The Reviewer1 has finished with the document, and it
    /// is now in the hands of the Review2 who has not started
    /// working with the document yet.
    /// </summary>
    Review2NotStarted = 5,
    /// <summary>
    /// The Reviewer2 has started working on the document.
    /// </summary>
    Review2InProgress = 6,
    /// <summary>
    /// The document has been finished by everyone.
    /// </summary>
    Completed = 7
}
```

TranslationDocumentUserRoleAssignmentDetails class

```
/// <summary>
/// Represents information about a user, with respect to its document assignment.
/// Used when listing documents with their user assignments.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentUserRoleAssignmentDetails
{
    /// <summary>
    /// Information about the user. When the assignment type is bidding
    /// or chaotsourcing, then the UserGuid field is
    /// "ffffffff-ffff-ffff-ffff-ffffffffffff", the UserName and the
    /// FullName fields are "Multiple".
    /// </summary>
    [DataMember]
    public UserInfoHeader UserInfoHeader;
    /// <summary>
    /// The document assignment role of the user. The currently accepted values are
    /// the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int DocumentAssignmentRole;
    /// <summary>
    /// The deadline for the specified user for the assigned document.
    /// </summary>
    [DataMember]
    public DateTime DeadLine;
}
```

UserInfoHeader class

```
/// <summary>
/// Represents brief information about a user.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class UserInfoHeader
{
    /// <summary>
    /// The guid of the user.
    /// </summary>
    [DataMember]
    public Guid UserGuid;
    /// <summary>
    /// The username (login name) of the user.
    /// </summary>
    [DataMember]
    public string UserName;
    /// <summary>
    /// The full name of the user.
    /// </summary>
    [DataMember]
    public string FullName;
}
```

ServerProjectTranslationDocumentWorkflowStatusChange class

```
/// <summary>
/// Represents manual Workflow status change of a particular document.
/// </summary>
[DataContract( Namespace = "http://kilgray.com/memoqservices/2007" )]
public partial class ServerProjectTranslationDocumentWorkflowStatusChange
{
    /// <summary>
    /// The Guid of the translation document (that has been returned by the
    /// ImportXXDocument operation.)
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The new Workflow status of the document.
    /// </summary>
    [DataMember]
    public WorkflowStatus WorkflowStatus;
}
```

DeliverDocumentRequest class

```
/// <summary>
/// Represents a delivery request of a particular document.
/// </summary>
[DataContract( Namespace = "http://kilgray.com/memoqservices/2007" )]
public partial class DeliverDocumentRequest
{
    /// <summary>
    /// The Guid of the translation document.
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The guid or the user who is delivering the document. The document
    /// must be assigned to the user in of of the roles (Translator,
    /// Reviewer1, Revier2) in order to be delivered.
    /// </summary>
    [DataMember]
    public Guid DeliveringUserGuid;
    /// <summary>
```

```
    /// Flag which determins if the document is returned to the previous
    /// actor (true), or to the next person in the assignment chain (false).
    /// </summary>
    [DataMember]
    public bool ReturnDocToPreviousActor;
}
```

3.7.9 User assignments – basic approach

The **IServerProjectService** operations for basic user management and related entity classes are discussed in this chapter. This chapter is only about the single user assignments, it does not contain information about the advanced assignment modes (group sourcing, first accept and subvendor assignments). If you would like to use the advanced assignment modes as well please read the next section of this document.

The operations and entity classes involved are the following:

- Assigning users to a translation document of a server project;
  - `IServerProjectService.SetProjectTranslationDocumentUserAssignments`
  - `ServerProjectTranslationDocumentUserAssignments`
  - `TranslationDocumentUserRoleAssignment`

The key concepts are the following:

- The `SetProjectTranslationDocumentUserAssignments` operation is to be used to set the assignments.
- To see the actual assignments there are different options:
  - If you use `ListProjectTranslationDocuments` operation to list the documents, the assignments are also included in the returned `ServerProjectTranslationDocInfo` objects. Please note thas this approach cannot return information about group sourcing, first accept and subvendor assignments.
  - Other approaches are described later in the chapter describing the advanced approach.

Actually you are free to mix the basic and advanced approach to set/list assignments, just keep in mind, that the basic approach cannot be used to set and return detailed information about advanced assignment scenarios (group sourcing, first accept and subvendor assignments).

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    …
    /// <summary>
```

```
    /// Sets the translation document to user assignments.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="assignments">The assignments to be set. Each item
    /// represents the assignment of users to one document. Existing assignments
    /// for the documents included are deleted and then set to the new values.
    /// Assignments of documents not included in the "assignments" array are
    /// not deleted.</param>
    [OperationContract]
    void SetProjectTranslationDocumentUserAssignments(Guid serverProjectGuid,
        ServerProjectTranslationDocumentUserAssignments[] assignments);
    …
}
```

ServerProjectTranslationDocumentUserAssignments class

```
/// <summary>
/// Represents assignments of users to a server project translation document.
/// Used when setting user-to-document assignments.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ServerProjectTranslationDocumentUserAssignments
{
    /// <summary>
    /// The Guid of the translation document (that has been returned by the
    /// ImportTranslationDocument operation.
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The list of the users assigned to the document(with their document
    /// assignment role and deadline). memoQ Server currently allows at most
    /// three three users to be assigned to a document, and the same user
    /// can not be assigned to the same document multiple times. At most one
    /// user can be assigned with a specific document assignment role (at most
    /// one translator, one reviewer1 and one reviewer2).
    /// Can not be null. According to the rules described above the array
    /// has to have 0-3 items.
    /// </summary>
    [DataMember]
    public TranslationDocumentUserRoleAssignment[] UserRoleAssignments;
}
```

TranslationDocumentUserRoleAssignment class

```
/// <summary>
/// Represents the assignment of a user to a server project translation document
/// for document assignment role with a deadline (e.g. userX to document Y with
/// document assignment role reviewer1 with deadline 1/1/2011).
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentUserRoleAssignment
{
    /// <summary>
    /// The Guid of the user.
    /// </summary>
    [DataMember]
    public Guid UserGuid;
    /// <summary>
    /// The document assignment role of the user. The currently accepted values are
    /// the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int DocumentAssignmentRole;
    /// <summary>
```

```
    /// The deadline for the specified user for the assigned document.
    /// </summary>
    [DataMember]
    public DateTime DeadLine;
}
```

3.7.10 User assignments - advanced

The **IServerProjectService** operations for advanced user management and related entity classes are discussed in this chapter. This chapter contains information about the advanced assignment modes (first accept, group sourcing and subvendor assignments) as well. Important: the advanced approach described here can also be used to set basic (single user) assignment: the difference is that using this approach you also will be able to set/list detailed information about advanced assignment scenarios as well (group sourcing, first accept and subvendor assignments). So even if you are not using these more exotic assignment types at this point, but plan to introduce them in the future, we recommend using this "advanced approach" from the beginning.

The key concepts are the following:

- The `SetTranslationDocumentAssignments` operation is to be used to set the assignments. Different subclasses of `TranslationDocumentAssignmentInfo` are to be used to set different king of assignments (e.g. `TranslationDocumentSingleUserAssignmentInfo` for single user assignment, `TranslationDocumentFirstAcceptAssignmentInfo` for first accept, etc.). The `TranslationDocumentNoUserAssignmentInfo` is to be used to remove a specific assignment.
- To see the actual assignments there are different options:
  - If you use the `ListProjectTranslationDocuments2` operation to list the documents, and the `FillInAssignmentInformation` is set to true in its `option` parameter, the assignment information is included in the returned `ServerProjectTranslationDocInfo2` objects (do this only when you really need the assignment info as well, as filling the assignment information has some performance penalty). The `UserAssignments` field of the returned objects is a `TranslationDocumentDetailedAssignmentInfo` derived class, the type depends on the type of the assignment (e.g. `TranslationDocumentDetailedSingleUserAssignmentInfo` in case of a single user assignment).
  - The `ListTranslationDocumentAssignments` operation can be used to return assignment information for specified documents. It returns an array of `TranslationDocumentDetailedAssignments` objects, each of which holds the assignment information about a specific document. The `Assignments` member is an array of `TranslationDocumentDetailedAssignmentInfo` derived objects, the type depends on the type of the assignment (e.g. `TranslationDocumentDetailedSingleUserAssignmentInfo` in case of a single user assignment).

The operations and entity classes involved are the following:

- Common classes
  - `TranslationDocumentAssignmentType`

- Assigning users to a translation document of a server project;
  - IServerProjectService.SetTranslationDocumentAssignments
  - SetTranslationDocumentAssignmentsOptions
  - TranslationDocumentAssignments
  - TranslationDocumentAssignmentInfo
  - TranslationDocumentRoleAssignmentInfo
  - TranslationDocumentSingleUserAssignmentInfo
  - TranslationDocumentFirstAcceptAssignmentInfo
  - TranslationDocumentGroupSourcingAssignmentInfo
  - TranslationDocumentSubvendorAssignmentInfo
  - TranslationDocumentNoUserAssignmentInfo
  - TranslationDocumentAssignmentResultInfo
  - TranslationDocumentRoleAssignmentResultInfo
- Retrieving the list of users assigned to translations documents
  - IServerProjectService.ListTranslationDocumentAssignments
  - ListTranslationDocumentAssignmentsOptions
  - TranslationDocumentDetailedAssignments
  - TranslationDocumentDetailedAssignmentInfo
  - TranslationDocumentAssigneeInfo
  - TranslationDocumentFirstAcceptUserInfo
  - FirstAcceptUserDecision
  - TranslationDocumentGroupSourcingUserInfo
  - TranslationDocumentDetailedRoleAssignmentInfo
  - TranslationDocumentDetailedSingleUserAssignmentInfo
  - TranslationDocumentDetailedFirstAcceptAssignmentInfo
  - TranslationDocumentDetailedGroupSourcingAssignmentInfo
  - TranslationDocumentDetailedSubvendorAssignmentInfo

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    …
    /// <summary>
    /// Sets the translation document to user/group of users/subvendor assignments.
    /// This is the advanced version of the
    /// SetProjectTranslationDocumentUserAssignments
    /// operations, as it enables advanced scenarios for assigning users to
    /// documents
    /// (first accept, group sourcing, subvendor group assignment).
    /// </summary>
```

```
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="options">The assignments to be set. Each item represents
    /// the assignment of user/group of users/subvendor to one document.
    /// Assignments  of documents not included in the "assignments" array are
    /// untouched (are not deleted).</param>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault)),
    FaultContract(typeof(TranslationDocumentAssignmentFault))]
    TranslationDocumentAssignmentResultInfo[] SetTranslationDocumentAssignments(
        Guid serverProjectGuid, SetTranslationDocumentAssignmentsOptions options);

    /// <summary>
    /// Lists the assignment information of the specified documents in the
    /// specified project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="options">The listing options.</param>
    /// <returns>The translation documents' assignment information.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    TranslationDocumentDetailedAssignments[] ListTranslationDocumentAssignments(
        Guid serverProjectGuid, ListTranslationDocumentAssignmentsOptions options);
    …
}
```

TranslationDocumentAssignmentType enum

```
/// <summary>
/// Describes the possible user-to-document assignment types.
/// </summary>
public enum TranslationDocumentAssignmentType
{
    /// <summary>
    /// Represents the single user assignment type.
    /// </summary>
    SingleUser,

    /// <summary>
    /// Represents the FirstAccept assignment type.
    /// </summary>
    FirstAccept,

    /// <summary>
    /// Represents the GroupSourcing assignment type.
    /// </summary>
    GroupSourcing,

    /// <summary>
    /// Represents the subvendor assignment type.
    /// </summary>
    Subvendor,

    /// <summary>
    /// Represents the no user assignment type.
    /// </summary>
    NoUser
}
```

3.7.10.1 Assigning users to a translation document of a server project

UML diagram of the entity classes involved:

SetTranslationDocumentAssignmentOptions class

```
/// <summary>
/// Encapsulates the assignment parameters of the server project
/// translation documents.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class SetTranslationDocumentAssignmentsOptions
{
    /// <summary>
    /// The assignments to be set. Each item represents the assignment
    /// of users/groups of users/subvendors to one document. Existing
    /// assignments of the specified documents will be changed only if
    /// the <see cref="DocumentAssignments"/> array of the document contains
    /// information about the already assigned roles.
    /// </summary>
    [DataMember]
    public TranslationDocumentAssignments[] DocumentAssignments;
```

```
    /// <summary>
    /// Indicates whether a specific fault should be thrown if an
    /// assignment problem occurs.
    /// </summary>
    [DataMember]
    public bool ThrowFault;
}
```

TranslationDocumentAssignments class

```
/// <summary>
/// Represents assignments of users to a server project translation document.
/// Used when setting user-to-document assignments.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentAssignments
{
    /// <summary>
    /// The guid of the translation document (that has been returned by the
    /// ImportXXDocument operation.)
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;

    /// <summary>
    /// The list of the user assignments of the document. Each role can
    /// appear at most once. Can not be null or empty. According to the
    /// rules described above the array has to have 1-3 items. An
    /// <see cref="ArgumentException"/> will be thrown if any of the
    /// above rules are broken.
    /// </summary>
    [DataMember]
    public TranslationDocumentAssignmentInfo[] Assignments;
}
```

TranslationDocumentAssignmentInfo class

```
/// <summary>
/// Base class of the assignment descriptor classes.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
[KnownType(typeof(TranslationDocumentRoleAssignmentInfo))]
public abstract partial class TranslationDocumentAssignmentInfo
{
    /// <summary>
    /// The deadline for the assigned user(s) or subvendor groups.
    /// </summary>
    [DataMember]
    public DateTime Deadline;

    /// <summary>
    /// Gets the assignment type.
    /// </summary>
    [DataMember]
    public TranslationDocumentAssignmentType AssignmentType;
}
```

TranslationDocumentRoleAssignmentInfo class

```
/// <summary>
/// Represents a user or a user group assignment for a role of a
/// document. Used when setting single user, FirstAccept, GroupSourcing
/// or subvendor group assignments.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
[KnownType(typeof(TranslationDocumentSingleUserAssignmentInfo))]
```

```
[KnownType(typeof(TranslationDocumentFirstAcceptAssignmentInfo))]
[KnownType(typeof(TranslationDocumentGroupSourcingAssignmentInfo))]
[KnownType(typeof(TranslationDocumentSubvendorAssignmentInfo))]
[KnownType(typeof(TranslationDocumentNoUserAssignmentInfo))]
public abstract partial class TranslationDocumentRoleAssignmentInfo :
TranslationDocumentAssignmentInfo
{
    /// <summary>
    /// The document assignment role of the user(s). The currently
    /// accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int RoleId;
}
```

TranslationDocumentSingleUserAssignmentInfo class

```
/// <summary>
/// Represents the assignment of a user to a server project translation document
/// for a document assignment role with a deadline (e.g. user X to document Y with
/// document assignment role reviewer1 with deadline 10/10/2011).
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentSingleUserAssignmentInfo :
TranslationDocumentRoleAssignmentInfo
{
    /// <summary>
    /// The guid of the user.
    /// </summary>
    [DataMember]
    public Guid UserGuid;
}
```

TranslationDocumentFirstAcceptAssignmentInfo class

```
/// <summary>
/// Represents the FirstAccept assignment of a group of users to a server project
/// translation document for a document assignment role with a deadline and a
/// FirstAccept deadline (e.g. user X, user Y to document Z with document
/// assignment role reviewer1 with deadline 10/10/2011 and FirstAccept deadline
/// 1/10/2011).
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentFirstAcceptAssignmentInfo :
TranslationDocumentRoleAssignmentInfo
{
    /// <summary>
    /// The deadline of the FirstAccept.
    /// </summary>
    [DataMember]
    public DateTime FirstAcceptDeadline;

    /// <summary>
    /// The guids of the assigned users.
    /// </summary>
    [DataMember]
    public Guid[] UserGuids;
}
```

TranslationDocumentGroupSourcingAssignmentInfo class

```
/// <summary>
/// Represents the GroupSourcing assignment of a group of users to a server project
/// translation document for a document assignment role with a deadline (e.g. user
```

```
/// X, user Y to document Z with document assignment role reviewer1 with deadline
/// 10/10/2011).
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentGroupSourcingAssignmentInfo :
TranslationDocumentRoleAssignmentInfo
{
    /// <summary>
    /// The guids of the assigned users.
    /// </summary>
    [DataMember]
    public Guid[] UserGuids;
}
```

TranslationDocumentSubvendorAssignmentInfo class

```
/// <summary>
/// Represents a subvendor group assignment. Used when setting
/// subvendor group to document assignments.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentSubvendorAssignmentInfo :
TranslationDocumentRoleAssignmentInfo
{
    /// <summary>
    /// The guid of the subvendor group.
    /// </summary>
    [DataMember]
    public Guid SubvendorGroupGuid;
}
```

TranslationDocumentNoUserAssignmentInfo class

```
/// <summary>
/// Represents an empty user assignment. Used when assigning
/// the document to no one.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentNoUserAssignmentInfo :
TranslationDocumentRoleAssignmentInfo
{ }
```

TranslationDocumentAssignmentResultInfo class

```
/// <summary>
/// Encapsulates the result of a document assignment operation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentAssignmentResultInfo : ResultInfo
{
    /// <summary>
    /// Encapsulates the result of the document role assignments.
    /// </summary>
    [DataMember]
    public TranslationDocumentRoleAssignmentResultInfo[] RoleAssignmentResults;

    /// <summary>
    /// Gets the error code of the assignment problem. It is null
    /// if the assignment was successful.
    /// </summary>
    [DataMember]
    public string ErrorCode;
}
```

TranslationDocumentRoleAssignmentResultInfo class

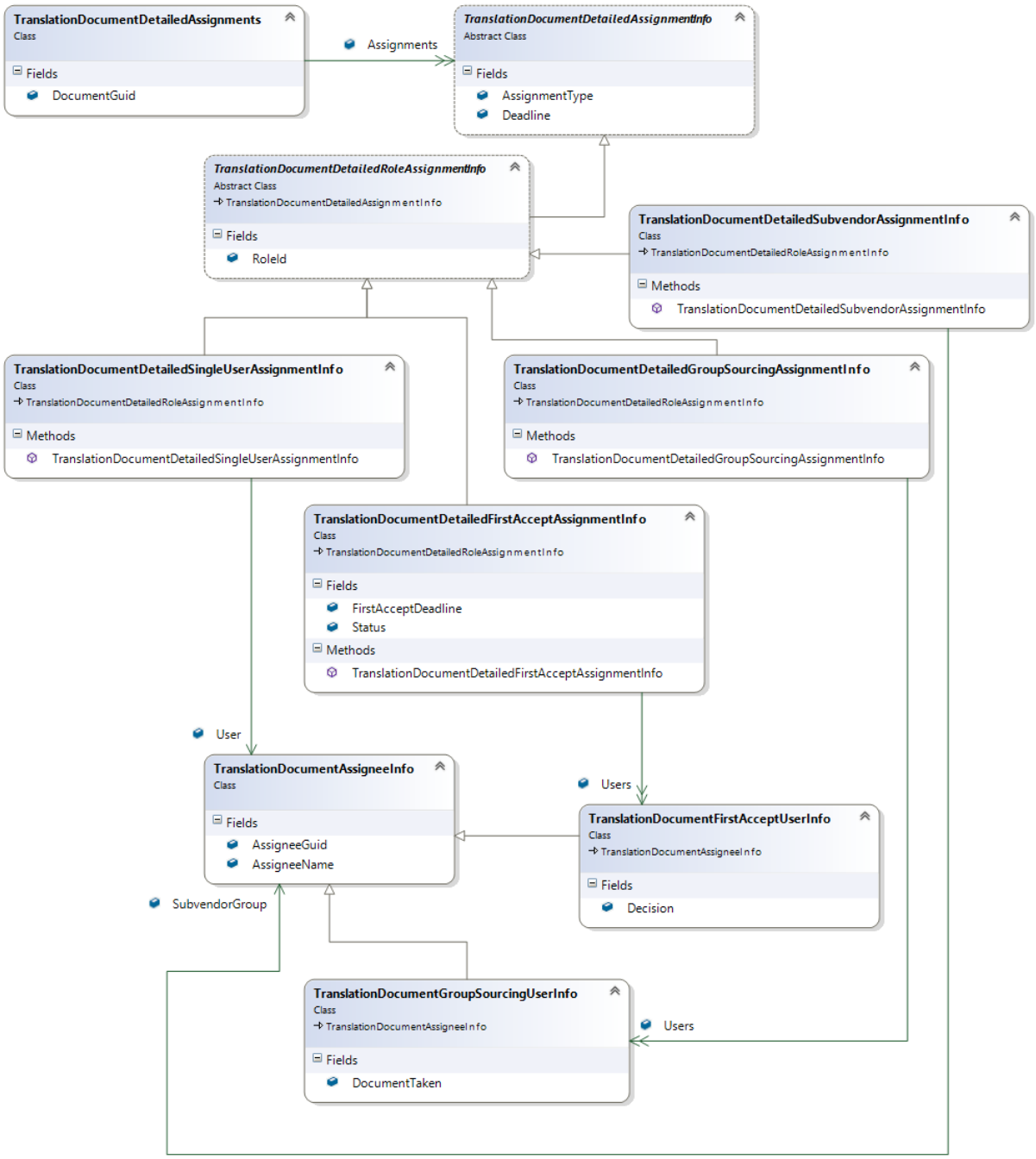
```
/// <summary>
```

```
/// Encapsulates the result of a document assignment operation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentRoleAssignmentResultInfo : ResultInfo
{
    /// <summary>
    /// Gets the role id.
    /// </summary>
    [DataMember]
    public int RoleId;

    /// <summary>
    /// Gets the error code of the role assignment problem.
    /// It is null if the assignment was successful.
    /// </summary>
    [DataMember]
    public string ErrorCode;
}
```

3.7.10.2 Retrieving the list of users assigned to translations documents

UML diagram of the entity classes involved:



ListTranslationDocumentAssignmentsOptions class

```
/// <summary>
/// Encapsulates the listing options of the server project
/// translation documents' assignments.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class ListTranslationDocumentAssignmentsOptions
{
    /// <summary>
    /// The guids of the documents which assigments should be listed.
    /// All documents' assignment will be listed if it is null.
    /// </summary>
    [DataMember]
    public Guid[] DocumentGuids;
}
```

TranslationDocumentDetailedAssignments class

```
/// <summary>
/// Represents the detailed assignments of users to a server project
/// translation document.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentDetailedAssignments
{
    /// <summary>
    /// The guid of the translation document (that has been returned by the
    /// ImportXXDocument operation.)
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;

    /// <summary>
    /// The list of the user assignments of the document. The same user
    /// can not be assigned to the same document multiple times. According
    /// to the rules described above the array has to have 0-3 items. The
    /// memoQ Server currently does not allow to assign different subvendor
    /// managers to the same document, but it will be supported later.
    /// Because of this the list has exactly three items in the case of the
    /// subvendor assignments and the threee items are the same, only the
    /// role identifiers are different.
    /// </summary>
    [DataMember]
    public TranslationDocumentDetailedAssignmentInfo[] Assignments;
}
```

TranslationDocumentDetailedAssignmentInfo class

```
/// <summary>
/// Base class of the detailed assignment descriptor classes.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
[KnownType(typeof(TranslationDocumentDetailedRoleAssignmentInfo))]
public abstract partial class TranslationDocumentDetailedAssignmentInfo
{
    /// <summary>
    /// The deadline for the assigned user(s) or subvendor managers.
    /// </summary>
    [DataMember]
    public DateTime Deadline;

    /// <summary>
    /// Gets the assignment type.
    /// </summary>
    [DataMember]
    public TranslationDocumentAssignmentType AssignmentType;
}
```

TranslationDocumentAssigneeInfo class

```
/// <summary>
/// Describes a user or a subvendor group who is assigned to a document.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
[KnownType(typeof(TranslationDocumentFirstAcceptUserInfo))]
[KnownType(typeof(TranslationDocumentGroupSourcingUserInfo))]
public class TranslationDocumentAssigneeInfo
{
    /// <summary>
    /// The guid of the user or the subvendor group.
    /// </summary>
    [DataMember]
    public Guid AssigneeGuid;
```

```
    /// <summary>
    /// The user name (login name) of the user or the name
    /// of the subvendor group.
    /// </summary>
    [DataMember]
    public string AssigneeName;
}
```

TranslationDocumentFirstAcceptUserInfo class

```
/// <summary>
/// Describes a user who is assigned to a document in a role with
/// FirstAccept assignment.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class TranslationDocumentFirstAcceptUserInfo :
TranslationDocumentAssigneeInfo
{
    /// <summary>
    /// Gets the user's FirstAccept decision.
    /// </summary>
    [DataMember]
    public FirstAcceptUserDecision Decision;
}
```

FirstAcceptUserDecision enum

```
/// <summary>
/// Describes the possible FirstAccept decisions of a user.
/// </summary>
public enum FirstAcceptUserDecision
{
    /// <summary>
    /// Used when the user is not decided yet whether he/she
    /// would like to work on the document.
    /// </summary>
    NotDecided,
    /// <summary>
    /// Used when the user rejected the document.
    /// </summary>
    Rejected,
    /// <summary>
    /// Used when the user accepted the document.
    /// </summary>
    Accepted
}
```

TranslationDocumentGroupSourcingUserInfo class

```
/// <summary>
/// Describes a user who is assigned to a document in a role with
/// GroupSourcing assignment.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class TranslationDocumentGroupSourcingUserInfo :
TranslationDocumentAssigneeInfo
{
    /// <summary>
    /// Gets the time in UTC when the document has been taken
    /// by the user. It is null is the document has not been
    /// taken yet.
    /// </summary>
    [DataMember]
    public DateTime? DocumentTaken;
}
```

TranslationDocumentDetailedRoleAssignmentInfo class

```
/// <summary>
/// Describes the details of a user/user group/subvendor group assignment.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
[KnownType(typeof(TranslationDocumentDetailedSingleUserAssignmentInfo))]
[KnownType(typeof(TranslationDocumentDetailedFirstAcceptAssignmentInfo))]
[KnownType(typeof(TranslationDocumentDetailedGroupSourcingAssignmentInfo))]
[KnownType(typeof(TranslationDocumentDetailedSubvendorAssignmentInfo))]
public abstract partial class TranslationDocumentDetailedRoleAssignmentInfo :
TranslationDocumentDetailedAssignmentInfo
{
    /// <summary>
    /// The document assignment role of the user(s). The currently
    /// accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int RoleId;
}
```

TranslationDocumentDetailedSingleUserAssignmentInfo class

```
/// <summary>
/// Describes the details of a single user assignment.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentDetailedSingleUserAssignmentInfo :
TranslationDocumentDetailedRoleAssignmentInfo
{
    /// <summary>
    /// Brief information about the assigned user.
    /// </summary>
    [DataMember]
    public TranslationDocumentAssigneeInfo User;
}
```

TranslationDocumentDetailedFirstAcceptAssignmentInfo class

```
/// <summary>
/// Describes the details of a FirstAccept assignment.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentDetailedFirstAcceptAssignmentInfo :
TranslationDocumentDetailedRoleAssignmentInfo
{
    /// <summary>
    /// The deadline of the FirstAccept.
    /// </summary>
    [DataMember]
    public DateTime FirstAcceptDeadline;

    /// <summary>
    /// Brief information about the FirstAccept users.
    /// </summary>
    [DataMember]
    public TranslationDocumentFirstAcceptUserInfo[] Users;

    /// <summary>
    /// Gets the status of the FirstAccept.
    /// </summary>
    [DataMember]
    public FirstAcceptStatus Status;
}
```

FirstAcceptStatus enum

```
/// <summary>
/// Describes the possible statuses of a FirstAccept.
/// </summary>
public enum FirstAcceptStatus
{
    /// <summary>
    /// Used when the FirstAccept is not started yet.
    /// </summary>
    NotStarted,
    /// <summary>
    /// Used when the FirstAccept is in pending status.
    /// </summary>
    Pending,
    /// <summary>
    /// Used when the FirstAccept document has been
    /// accepted by someone.
    /// </summary>
    DocTaken,
    /// <summary>
    /// Used when the FirstAccept document has been
    /// rejected by everyone.
    /// </summary>
    Failed,
    /// <summary>
    /// Used when the FirstAccept deadline expired.
    /// </summary>
    TimedOut
}
```

TranslationDocumentDetailedGroupSourcingAssignmentInfo class

```
/// <summary>
/// Describes the details of a GroupSourcing assignment.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentDetailedGroupSourcingAssignmentInfo :
TranslationDocumentDetailedRoleAssignmentInfo
{
    /// <summary>
    /// Brief information about the GroupSourcing users.
    /// </summary>
    [DataMember]
    public TranslationDocumentGroupSourcingUserInfo[] Users;
}
```

TranslationDocumentDetailedSubvendorAssignmentInfo class

```
/// <summary>
/// Describes the details of a subvendor assignment.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class TranslationDocumentDetailedSubvendorAssignmentInfo :
TranslationDocumentDetailedRoleAssignmentInfo
{
    /// <summary>
    /// Brief information about the subvendor group.
    /// </summary>
    [DataMember]
    public TranslationDocumentAssigneeInfo SubvendorGroup;
}
```

3.7.10.3 Reported errors

Important note: When using the Server Project API make sure to enable advanced error handling mode.

The SetTranslationDocumentAssginments function reports various types of errors. Besides the generally used UnexpectedFault and GenericFault one specific fault is used: TranslationDocumentAssignmentFault. This fault will be thrown only if you set the ThrowFault member of the SetTranslationDocumentAssignmentOptions class to true. Otherwise the ErrorCode member of the TranslationDocumentAssignmentResultInfo and the TranslationDocumentRoleAssignmentResultInfo will contain the fault code. The TranslationDocumentAssignmentFault has an ErrorCode member so that the caller can differentiate between different types of assignment related errors. Using older WS technologies it is difficult to access the ErrorCode: you can check the fault code instead, which has the same value as the TranslationDocumentAssignmentFault.ErrorCode. Possible fault types, ErrorCode/fault code values with their fault message are the following:

Fault type	**Fault code** (same as ErrorCode in case of TranslationDocumentAssignmentFault)	**Fault message**
TranslationDocumentAssignmentFault	TranslationDocumentAssignment_DocDoesNotExist	The document does not exist in the project.
TranslationDocumentAssignmentFault	TranslationDocumentAssignment_DocIsDivided	The document is sliced.
TranslationDocumentAssignmentFault	TranslationDocumentAssignment_DocIsGroupAssignedInLTConnectedProject	In LT-connected projects only single user assignments are allowed.
TranslationDocumentAssignmentFault	TranslationDocumentAssignment_UserIsNotInTheProject	The user is not assigned to the project.
TranslationDocumentAssignmentFault	TranslationDocumentAssignment_SVGroupDoesNotExist	The subvendor group does not exsit anymore.
TranslationDocumentAssignmentFault	TranslationDocumentAssignment_NormalGroupInSVAssignment	Can not assign document to normal group with subvendor assignment.
TranslationDocumentAssignmentFault	TranslationDocumentAssignment_SVUserOrManagerInNormalAssignment	Can not assign document to subvendor user or manager with single user assignment, FirstAccept or GroupSourcing.
GenericFault	Generic	<The message of the original exception is used as fault message.>
UnexpectedFault	Unexpected	<The message of the original exception is used as fault message.>

Please note that using C# and WCF these faults can be caught based on their type using regular exceptions:

```
try
{
    // call operation
}
// This catches TMFault errors only
catch (System.ServiceModel.FaultException<TranslationDocumentAssignmentFault> e)
{

    // Check e.Code.Name or e.ErrorCode for the fault code
}
// This catches all other API errors
catch (System.ServiceModel.FaultException e)
{

    // Check e.Code.Name or e.ErrorCode for the fault code if you want to
    // differentiate between errors
}
```

Using older WS technologies, such as ASMX, you need to check the fault code:

```
try
{
    // call operation
}
// Important: this catches all API errors, not only AddProjectLanguageFault
// errors!!!
catch (SoapException e)
{
    // Check e.Code.Name or e.ErrorCode for the fault code if you want to
    // differentiate between errors
}
```

3.7.11 Statistics

The **IServerProjectService** operations for statistics and related entity classes are discussed in this chapter.

The operations and entity classes involved are the following:

- Get statistics on all documents of a server project;
  - IServerProjectService.GetStatisticsOnProject
  - StatisticsOptions
  - StatisticsAlgorithm
  - StatisticsResultFormat
- Get statistics on specific documents of a server project;
  - IServerProjectService.GetStatisticsOnTranslationDocuments
  - StatisticsOptions
  - StatisticsAlgorithm
  - StatisticsResultFormat

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
```

```
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    ...
    /// <summary>
    /// Creates the statistics on the whole server project (includes all
    /// documents).
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="targetLangCodes">The three+(two) letter codes of target
    /// languages for which project level statistics is to be created.
    /// If null, the statistics is created for all target languages.</param>
    /// <param name="options">The options of the statistics.</param>
    /// <param name="resultFormat">The requested result format.</param>
    /// <returns>The result in the requested format.
    /// Important note: the returned StatisticsResultInfo.ResultsForTargetLangs
    /// does not inlude items for which target language no document exists
    /// in the project.
    /// </returns>
    [OperationContract]
    StatisticsResultInfo GetStatisticsOnProject(Guid serverProjectGuid,
        string[] targetLangCodes,
        StatisticsOptions options, StatisticsResultFormat resultFormat);
    /// <summary>
    /// Creates the statistics on specified documents of the server project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="translationDocNames">The guids of the documents
    /// on which statistics is to be created. Each document included has to have
    /// the same target language.
    /// </param>
    /// <param name="options">The options of the statistics.</param>
    /// <param name="resultFormat">The requested result format.</param>
    /// <returns>The result in the requested format. </returns>
    [OperationContract]
    StatisticsResultInfo GetStatisticsOnTranslationDocuments(
         Guid serverProjectGuid,
        Guid[] translationDocGuids, StatisticsOptions options,
        StatisticsResultFormat resultFormat);
    ...
}
```

StatisticsResultFormat enum

```
/// <summary>
/// Identifies different output formats for statistics.
/// </summary>
public enum StatisticsResultFormat
{
    /// <summary>
    /// The output is in HTML format.
    /// </summary>
    Html,
    /// <summary>
    /// The output is in CSV format reflecting the HTML format.
    /// </summary>
    CSV_WithTable,
    /// <summary>
    /// The output is in Trados CSV format (per file, trados compatible)
    /// </summary>
    CSV_Trados,
    /// <summary>
    /// The output is in MemoQ CSV format (per file, all information)
    /// </summary>
    CSV_MemoQ
```

```
}
```

StatisticsAlgorithm enum

```
/// <summary>
/// Identifies calculation algoritms used by statistics.
/// </summary>
public enum StatisticsAlgorithm
{
    /// <summary>
    /// Identifies MemoQ like counting.
    /// </summary>
    MemoQ,
    /// <summary>
    /// Identifies Trados like counting.
    /// </summary>
    Trados
}
```

StatisticsOptions class

```
/// <summary>
/// Encapsulates options for running statistics on server project translation
/// documents.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class StatisticsOptions
{
    /// <summary>
    /// If true, information for every single document is included in the
    /// output, not only aggregates.
    /// </summary>
    [DataMember]
    public bool ShowResultsPerFile;
    /// <summary>
    /// If true analysis if performed on the transation memories of the project.
    /// </summary>
    [DataMember]
    public bool Analysis_ProjectTMs;
    /// <summary>
    /// If true, analysis for each project TM is inluded in the output.
    /// Ignored if Analysis_ProjectTMs is false.
    /// </summary>
    [DataMember]
    public bool Analyzis_DetailsByTM;
    /// <summary>
    /// If true, homogenity analysis if permormed.
    /// </summary>
    [DataMember]
    public bool Analysis_Homogenity;
    /// <summary>
    /// Determines of translation segments (rows) locked by users are to
    /// be included in the statistic calculations.
    /// </summary>
    [DataMember]
    public bool IncludeLockedRows;
    /// <summary>
    /// If true, segment counts are included in the output.
    /// </summary>
    [DataMember]
    public bool ShowCounts;
    /// <summary>
    /// If true, segment counts for the target segments are included in the
    /// output. Ignored if ShowCounts is false.
    /// </summary>
    [DataMember]
    public bool ShowCounts_IncludeTargetCount;
    /// <summary>
```

```
    /// If true, status report is included in the Counts output.
    /// Ignored if ShowCounts is false.
    /// </summary>
    [DataMember]
    public bool ShowCounts_StatusReport;
     /// <summary>
     /// If true, status report character counts include whitespaces; false
     /// otherwise. Default behavior is false.
     /// </summary>
    [DataMember]
    public bool ShowCounts_IncludeWhitespacesInCharCount = false;
    /// <summary>
    /// Determines the algorithm to be used.
    /// </summary>
    [DataMember]
    public StatisticsAlgorithm Algorithm;
    /// <summary>
    /// Tag word weightening.
    /// Must be between 0 and 9.99 (both inclusive).
    /// If the word weight is 0.2, and a category contains 12 words
    /// and 5 tags, then the adjusted word count will be 13.
    /// </summary>
    [DataMember]
    public double TagWordWeight;
     /// <summary>
    /// Tag char weightening.
    /// Must be between 0 and 9.99 (both inclusive).
    /// </summary>
    [DataMember]
    public double TagCharWeight;
    /// <summary>
    /// If true, repetitive 100% matches are counted as repetitions; if
    /// false, they are counted as 100% matches. Default behavior is true.
    /// </summary>
    [DataMember]
    public bool RepetitionPreferenceOver100 = true;
    /// <summary>
    /// If true, identical segments in different documents are not counted
    /// as repetitions.
    /// </summary>
    [DataMember]
    public bool DisableCrossFileRepetition;
 }
```

StatisticsResultInfo class

```
/// <summary>
/// Encapsulates the result of a statistics operation for possibly
/// more than one target languages.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class StatisticsResultInfo : ResultInfo
{
    /// <summary>
    /// The array of statistic results: the array has one element for
    /// each target language. If the statistics operation has finished
    /// with some error, its value is undefined.
    /// </summary>
    [DataMember]
    public StatisticsResultForLang[] ResultsForTargetLangs;
}
```

StatisticsResultForLang class

```
/// <summary>
/// Encapsulates the result of a statistics operation for a single
```

```
/// target language.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class StatisticsResultForLang
{
    /// <summary>
    /// The target language code for which ResultData holds data for.
    /// </summary>
    [DataMember]
    public string TargetLangCode;
    /// <summary>
    /// The result in the requested format in Unicode encoding.
    /// </summary>
    [DataMember]
    public byte[] ResultData;
}
```

3.7.12 Pre-translation

The **IServerProjectService** operations for pre-translation and related entity classes are discussed in this chapter.

The operations and entity classes involved are the following:

- Pre-translate all documents of a server project;
  - IServerProjectService.PretranslateProject
  - PretranslateOptions
  - PretranslateLookupBehavior
  - PretranslateExpectedFinalTranslationState
  - PretranslateStateToConfirmAndLock
  - PretranslateCopySourceToTargetBehavior
  - PretranslateCopySourceToTargetConditions
- Pre-translate specific documents of a server project;
  - IServerProjectService.PretranslateDocuments
  - PretranslateOptions
  - PretranslateLookupBehavior
  - PretranslateExpectedFinalTranslationState
  - PretranslateStateToConfirmAndLock
  - PretranslateCopySourceToTargetBehavior
  - PretranslateCopySourceToTargetConditions
  - CustomPreTranslateParameter

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
```

```
    ...
    /// <summary>
    /// Pre-translates the whole server project (includes all documents for
    /// the specified target languages).
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="targetLangCodes">The three+(two) letter codes of target
    /// languages for which project level pretranslate is to be performed.
    /// If null, the pretranslate is performed for all target languages.</param>
    /// <param name="options">The options of the pre-translate process.</param>
    /// <returns>
    /// The ResultInfo object providing information about the success
    /// or failure of the operation.
    /// </returns>
    [OperationContract]
    ResultInfo PretranslateProject(Guid serverProjectGuid,
       string[] targetLangCodes, PretranslateOptions options);
    /// <summary>
    /// Pre-translates the specified documents of the server project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="translationDocGuids">The Guids of the documents
    /// to be pretranslated.</param>
    /// <param name="options">The options of the pre-translate process.</param>
    /// <returns>
    /// The ResultInfo object providing information about the success
    /// or failure of the operation.
    /// </returns>
    [OperationContract]
    ResultInfo PretranslateDocuments(Guid serverProjectGuid, Guid[]
            translationDocGuids, PretranslateOptions options);
    ...
}
```

PretranslateLookupBehavior enumeration

```
/// <summary>
/// Represents pre-tanslate lookup behavior.
/// </summary>
public enum PretranslateLookupBehavior
{
    /// <summary>
    /// Indicates to pre-translate only those segments for which 101%
    /// or 100% match is found.
    /// </summary>
    ExactMatchWithContext,
    /// <summary>
    /// Indicates to pre-translate only those segments for which 100%
    /// match is found.
    /// </summary>
    ExactMatch,
    /// <summary>
    /// Indicates to pre-translate only those segments for which a match
    /// above the "good match threshold.
    /// </summary>
    GoodMatch,
    /// <summary>
    ///  Indicates to pre-translate all segments.
    /// </summary>
    AnyMatch
}
```

PretranslateExpectedFinalTranslationState enumeration

```
/// <summary>
```

```
/// Represents the statuses of segments after pretranslation.
/// </summary>
public enum PretranslateExpectedFinalTranslationState
{
    /// <summary>
    /// The status of the segment is not altered.
    /// </summary>
    NoChange,
    /// <summary>
    /// The status of the segments is translator confirmed after
    /// pre-translation.
    /// </summary>
    Confirmed,
    /// <summary>
    /// The status of the segments is reviewer 2 confirmed (proofread)
    /// after pre-translation.
    /// </summary>
    Proofread,
    /// <summary>
    /// The status of the segments is pretranslated after pre-translation.
    /// </summary>
    Pretranslated,
    /// <summary>
    /// The status of the segments is reviewer 1 confirmed.
    /// </summary>
    Reviewer1Confirmed
}
```

PretranslateStateToConfirmAndLock enumeration

```
/// <summary>
/// Represents the state of segments that should be conformed and locked
/// after pre-translation.
/// </summary>
public enum PretranslateStateToConfirmAndLock
{
    /// <summary>
    /// No segment is locked and confirmed after pre-translation.
    /// </summary>
    None,
    /// <summary>
    /// Segments for which the pre-translate found an exact match is
    /// locked and confirmed after pre-translation.
    /// </summary>
    ExactMatch,
    /// <summary>
    /// Segments for which the pre-translate found an exact match with
    /// context is locked and confirmed after pre-translation.
    /// </summary>
    ExactMatchWithContext,
    /// <summary>
    /// Segments for which the pre-translate found an exact match with
    /// IceSpice context is locked and confirmed after pre-translation.
    /// </summary>
    IceSpiceMatch
}
```

PretranslateOptions class

```
/// <summary>
/// Encapsulates options for performing pre-translate on server project
/// translation documents.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class PretranslateOptions
{
```

```
    /// <summary>
    /// Represents pre-translate overwrite behavior. See
    /// PretranslateLookupBehavior for details.
    /// </summary>
    [DataMember]
    public PretranslateLookupBehavior PretranslateLookupBehavior;
    /// <summary>
    /// Indicates to pre-translate only those segments for which only one
    /// 101% match or 100% match is found, and to leave everything else
    /// unchanged. This option shall be used only if PretranslateLookupBehavior
    /// is ExactMatchWithContext or ExactMatch.
    /// </summary>
    [DataMember]
    public bool OnlyUnambiguousMatches;
    /// <summary>
    /// The level of the goodmatch TM rate. 80% is a tipical value.
    /// </summary>
    [DataMember]
    public int GoodMatchRate;
    /// <summary>
    /// If true, Machine Translation is also used during pretranslation.
    /// </summary>
    [DataMember]
    public bool UseMT;
    /// <summary>
    /// Confirm and lock segments after pre-translation. If false,
    /// the segments are not confirmed and locked.
    /// </summary>
    [DataMember]
    public bool LockPretranslated;
    /// <summary>
    /// Specifies which segments should be locked and confirmed after
    /// pre-translation. To confirm and lock the segments requires
    /// LockPretranslated flag to be set true too.
    /// </summary>
    [DataMember]
    public PretranslateStateToConfirmAndLock ConfirmLockPretranslated;
    /// <summary>
    /// Confirms/lockes segmentw after pre-translation with unambiguous
    /// matches only. To confirm and lock the segments requires
    /// LockPretranslated flag to be set true too.
    /// </summary>
    [DataMember]
    public bool ConfirmLockUnambiguousMatchesOnly;
    /// <summary>
    /// The status of the segments to set after pre-translation. If a
    /// segment is pre-translated this status will be set. Segments
    /// which no appropriate match is found are not changed.
    /// </summary>
    [DataMember]
    public PretranslateExpectedFinalTranslationState FinalTranslationState;
    /// <summary>
    /// Instead of pre-translation, copy the source to target in some
    /// cases. Describes the options when to copy the source to target.
    /// If this member is null all segments are lookup up in the TMs
    /// during pre-translation. If this member is not null, the rows
    /// matching the criteria are copied to the target side and not
    /// looked up in the TMs.
    /// </summary>
    [DataMember]
    public PretranslateCopySourceToTargetBehavior CopySourceToTarget;
    /// <summary>
    /// The custom execution parameters of the pre-translate. For more
    /// details see <see cref="CustomPreTranslateExecutionParameter"/>.
    /// </summary>
    [DataMember]
    public CustomPreTranslateParameter[] CustomParameters;
}
```

PretranslateCopySourceToTargetBehavior class

```
/// <summary>
/// Encapsulates options of copy source to target behavior during a
/// pre-translation operation on server project translation documents.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class PretranslateCopySourceToTargetBehavior
{
    /// <summary>
    /// Describes the type of rows to copy instead of pre-translation.
    /// </summary>
    [DataMember]
    public PretranslateCopySourceToTargetConditions Condition;
    /// <summary>
    /// Lock the row after copy source to target.
    /// </summary>
    [DataMember]
    public bool Lock;
}
```

PretranslateCopySourceToTargetConditions enumeration

```
/// <summary>
/// Describes the type of rows where copy source to target is used
/// during pre-translation.
/// <summary>
public enum PretranslateCopySourceToTargetConditions
{
    SegmentsWithOnlyTagsAndWhitespace,
    SegmentsWithOnlyTagsWhitespaceAndPunctuation
}
```

CustomPreTranslateParameter class

```
/// <summary>
/// Encapsulates a custom pre-translate parameter.
/// </summary>
public partial class CustomPreTranslateParameter
{
    /// <summary>
    /// The key of the parameter. The currently accepted values are:
    /// - MicrosoftMTCategoryId: will be taken into consideration only
    ///   if the server's active MT engine is the Microsoft MT plugin.
    ///   The server will use the configured category ID from the MT
    ///   configuration file if the value is null, will not use any
    ///   category ID if the value is empty and will use the specified
    ///   value otherwise.
    /// </summary>
    [DataMember]
    public string Key;
    /// <summary>
    /// The value of the parameter.
    /// </summary>
    [DataMember]
    public string Value;
}
```

3.7.13 Confirm and update

The **IServerProjectService** operations for confirm and update and related entity classes are discussed in this chapter.

The operations and entity classes involved are the following:

- Confirm and update all or specific documents of a server project;
  - IServerProjectService.ConfirmAndUpdate
  - IServerProjectService.ConfirmAndUpdate2
  - Statuses
  - UserNameBehaviors
  - ConfirmAndUpdateOptions
  - ConfirmAndUpdateDocError
  - ConfirmAndUpdateResultInfo

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    ...
    /// <summary>
    /// Confirms and updates rows of the specified documents in the specified
    /// server project.
    /// </summary>
    /// <param name="sessionId">The session id belonging to the user
    /// who performs the confirm operation. This user's user name is used when
    /// the server updates the stauses of the segments.
    /// If the UserNameBehavior is UseProjectDefault in the options parameter,
    /// the server uses this user's name as creator/modifier when updating
    /// the translation memories.</param>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="translationDocGuids">The guids of the documents to confirm
    ///  and update.</param>
    /// <param name="options">The options of the confirm and update process.
    /// </param>
    /// <returns>
    /// The ResultInfo object providing information about the success
    /// or failure of the operation.
    /// </returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    ConfirmAndUpdateResultInfo ConfirmAndUpdate(string sessionId,
        Guid serverProjectGuid, Guid[] translationDocGuids,
        ConfirmAndUpdateOptions options);

    /// <summary>
    /// Confirms and updates rows of the specified documents in the specified
    /// server project.
    /// </summary>
    /// <param name="userGuid">The guid of to the user who performs the confirm
    /// and update operation. This user's user name is used when the server
    /// updates the statuses of the segments.
    /// If the UserNameBehavior is UseProjectDefault in the options parameter,
    /// the server uses this user's name as creator/modifier when updating the
    /// translation memories.</param>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="translationDocGuids">The guids of the documents to confirm
    ///  and update.</param>
    /// <param name="options">The options of the confirm and update process.
    /// </param>
    /// <returns>
    /// The ResultInfo object providing information about the success
```

```
    /// or failure of the operation.
    /// </returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    ConfirmAndUpdateResultInfo ConfirmAndUpdate2(Guid userGuid,
        Guid serverProjectGuid, Guid[] translationDocGuids,
        ConfirmAndUpdateOptions options);
    ...
}
```

ConfirmAndUpdateSegmentStatuses enumeration

```
/// <summary>
/// Represents the status of segments that should be confirmed.
/// </summary>
[Flags]
public enum ConfirmAndUpdateSegmentStatuses
{
    /// <summary>
    /// The status of the segment is edited.
    /// </summary>
    Edited = 1,
    /// <summary>
    /// The status of the segment is translator confirmed.
    /// </summary>
    Confirmed = 2,
    /// <summary>
    /// The status of the segment is reviewer 2 confirmed
    /// (proofread).
    /// </summary>
    Proofread = 4,
    /// <summary>
    /// The status of the segment is pre-translated.
    /// </summary>
    PreTranslated = 8,
    /// <summary>
    /// The segment is locked.
    /// </summary>
    Locked = 16,
    /// <summary>
    /// All possible segment statuses.
    /// </summary>
    All = 32,
    /// <summary>
    /// The status of the segment is reviewer 1 confirmed.
    /// </summary>
    Reviewer1Confirmed = 64,
}
```

ConfirmAndUpdateUserNameBehaviors enumeration

```
/// <summary>
/// Represents the user name behavior for confirming.
/// </summary>
public enum ConfirmAndUpdateUserNameBehaviors
{
    /// <summary>
    /// The server will use the project's default user name.
    /// For more details see the description of the ConfirmAndUpdateOptions class.
    /// </summary>
    UseProjectDefault,
    /// <summary>
    /// The server will use the user names stored with the rows.
    /// For more details see the description of the ConfirmAndUpdateOptions class.
    /// </summary>
    UseNameStoredWithRow
}
```

ConfirmAndUpdateOptions class

```
/// <summary>
/// Encapsulates options for confirm and update rows on server project translation
/// documents.
/// </summary>
/// <remarks>
/// Based on the value of the UserNameBehavior field there are two different ways
/// to confirm and update the segments of a document:
/// 1)  The value of the UserNameBehavior is UseProjectDefault: in this case the
///     server will use the name of the user, who performs the confirm and update
///     operation, as creator/modifier when updating the translation memories, and
///     updateing the statuses of the document segments.
///     You can specify this user with the other parameters of the confirm and
///     update WS-API functions. If you use the ConfrimAndUpdate function, you have
///     to provide the sessionId belonging to the user who performs the confirm and
///     update. If you use the ConfirmAndUpdate2 fuction, you have
///     to provide the guid of the user who performs the confirm and update
///     operation.
/// 2)  The value of the UserNameBehavior is UseNameStoredWithRow: in this case
///     the server uses the user names stored with the document rows, when updating
///     the translation memories during the confirm and update process. You should
///     use the DocumentRoleToUse field, to specify from wich role would you like
///     to use the user name. If there is no stored user name
///     in this role for the document, the server will use the value of the
///     DefaultUserName field.
/// </remarks>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ConfirmAndUpdateOptions
{
    /// <summary>
    /// The statuses of segments to confirm. The segments with other
    /// statuses will be skipped during the confirm and update operation.
    /// </summary>
    [DataMember]
    public ConfirmAndUpdateSegmentStatuses Status;
    /// <summary>
    /// The use name behavior to use during the confirm and update process.
    /// </summary>
    [DataMember]
    public ConfirmAndUpdateUserNameBehaviors UserNameBehavior;
    /// <summary>
    /// If UserNameBehavior is UseNameStoredWithRowuser then you can specify
    /// which document assignment role to use to determine the user name
    /// during the update of the translation memories.
    /// The currently accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    /// <remarks>
    /// The server will use this field only when the UserNameBehavior is
    /// UseNameStoredWithRow.
    /// </remarks>
    [DataMember]
    public byte DocumentRoleToUse;
    /// <summary>
    /// memoQ will use this default user name during the update of the translation
    /// memories if the user name for the selected document assignment role (see
    /// DocumentRoleToUse) is not defined.
    /// </summary>
    /// <remarks>
    /// The server will use this field only when the UserNameBehavior is
    /// UseNameStoredWithRow.
    /// </remarks>
    [DataMember]
    public string DefaultUserName;
    /// <summary>
```

```
    /// The custom meta definitions and values as multi-line string.
    /// Null and as well as an empty string both mean "no custom metas".
    /// Each line has three or more columns, separated by tabs:
    ///     metaName   metaType   value   [value2...valueN]
    /// Possible metaType values:
    ///     FreeText
    ///     Number
    ///     DateTime
    ///     PickListSingle
    ///     PickListMultiple
    /// </summary>
    [DataMember]
    public string CustomMetas;
    /// <summary>
    /// Also update the TM repository during confirm and update operation.
    /// </summary>
    [DataMember]
    public bool UpdateTMRepository = false;
    /// <summary>
    /// Indicates whether make rows proofread or only confirmed.
    /// It does not modify the status of the segment if the segment's
    /// status is already proofread and the parameter value is false.
    /// </summary>
    /// <remarks>
    /// OBSOLETE - FROM SERVER VERSION 7.5 USE RoleForConfirmation PROPERTY
    /// TO SET THE DOCUMENT ASSIGNMENT ROLE TO CONFIRM SEGMENTS.
    /// If UseRoleForConfirmation is true the specified role will be used,
    /// else the one defined by this property.</remarks>
    [DataMember]
    [Obsolete("From server version 7.5 use RoleForConfirmation property instead")]
    public bool MakeRowsProofread = true;
    /// <summary>
    /// If UseRoleForConfirmation is true then you can specify
    /// which document assignment role to use to confirm segments.
    /// The currently accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// 3: Unchanged - Translation state wont be changed
    /// </summary>
    /// <remarks>
    /// The server will use this field only when the UseRoleForConfirmation is true
    /// </remarks>
    [DataMember]
    public byte RoleForConfirmation;
    /// <summary>
    /// Indicates whether use the given role for confirmation.
    /// If the parameter value is false the given confirmation role will not be
    /// observed, then it will be defined by MakeRowsProofread property.
    /// </summary>
    [DataMember]
    public bool UseRoleForConfirmation = true;
}
```

ConfirmAndUpdateDocError class

```
/// <summary>
/// Encapsulates the result of a confirm and update
/// operation for a single document.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ConfirmAndUpdateDocError
{
    /// <summary>
    /// The Guid of the document.
```

```
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The error code. Possible values:
    ///     TMBusy
    ///     TMReadonly
    ///     TMCorrupt
    ///     NoPrimaryTM
    ///     TMDoesNotExist
    /// </summary>
    [DataMember]
    public string ErrorCode;
}
```

ConfirmAndUpdateResultInfo class

```
/// <summary>
/// Encapsulates the result of a confirm and update operation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ConfirmAndUpdateResultInfo
{
    /// <summary>
    /// Contains error informations about the confirmable documents.
    /// It is null if no error occured.
    /// </summary>
    [DataMember]
    public ConfirmAndUpdateDocError[] DocErrors;
}
```

3.7.14 X-translation

The **IServerProjectService** operation for X-translation and related entity classes are discussed in this chapter.

The operation and entity classes involved are the following:

- X-translate specific documents of a server project;
  - IServerProjectService.XTranslate
  - XTranslateDocInfo
  - XTranslateScenario
  - ExpectedSourceStateBeforeXTranslate
  - ExpectedFinalStateAfterXTranslate
  - NewRevisionScenarioOptions
  - XTranslateOptions
  - XTranslateDocumentResult
  - XTranslateResultInfo

The involved **IServerProjectService** operation and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
```

```
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    ...
    /// <summary>
    /// X-translates the rows of the specified documents of the
    /// specified server project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="options">The X-translate options.</param>
    /// <returns>The XTranslateResultInfo object providing information
    /// about the candidates and about the X-translated rows.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    XTranslateResultInfo XTranslate(Guid serverProjectGuid,
        XtranslateOptions options);
    ...
}
```

XTranslateDocInfo class

```
/// <summary>
/// Encapsulates the document identifier and the document's
/// source major version to use for for X-translation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class XTranslateDocInfo
{
    /// <summary>
    /// The guid of the document.
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The source major version to use for X-translation.
    /// The -1 means the latest major version.
    /// </summary>
    [DataMember]
    public int SourceMajorVersion;

}
```

XTranslateScenario enumeration

```
/// <summary>
/// Represents the X-translate scenarios of the X-translate logic.
/// </summary>
public enum XTranslateScenario
{
    /// <summary>
    /// If you use this option memoQ will populate the document with
    /// previous approved translations, and mark rows with the "XLT"
    /// match rate. Translator's name is not preserved.
    /// </summary>
    NewRevision,
    /// <summary>
    /// If you use this option memoQ will populate rows with their
    /// previous target segment, preserving their status. The name
    /// of the translator is preserved.
    /// </summary>
    MidProjectUpdate
}
```

ExpectedSourceStateBeforeXTranslate enumeration

```
/// <summary>
/// Represents the statuses of the segments which
/// segments should be applied during the X-translation.
/// </summary>
public enum ExpectedSourceStateBeforeXTranslate
{
    /// <summary>
    /// Rows with a proofread status should be applied.
    /// </summary>
    ProofreadOnly,
    /// <summary>
    /// Rows with a translator confirmed, reviewer 1 confirmed or reviewer 2
    /// confirmed (prooppread) status should be applied.
    /// </summary>
    ProofreadOrConfirmed,
    /// <summary>
    /// All rows should be applied.
    /// </summary>
    AllTarget
}
```

ExpectedFinalStateAfterXTranslate enumeration

```
/// <summary>
/// Represents the expected translation
/// states of the X-translated segmenst.
/// </summary>
public enum ExpectedFinalStateAfterXTranslate
{
    /// <summary>
    /// The final state will be the same as in the previous version.
    /// </summary>
    SameAsPrevious,
    /// <summary>
    /// The final state will be pre-translated.
    /// </summary>
    Pretranslated,
    /// <summary>
    /// The final state will be translator confirmed.
    /// </summary>
    Confirmed,
    /// <summary>
    /// The final state will be reviewer 2 confirmed (proofread).
    /// </summary>
    Proofread,
    /// <summary>
    /// The final state will be reviewer 1 confirmed.
    /// </summary>
    Reviewer1Confirmed
}
```

NewRevisionScenarioOptions class

```
/// <summary>
/// Encapsulates the options for X-translation,
/// if the X-translate scenario is NewRevision.
/// </summary>
[DataContract]
public partial class NewRevisionScenarioOptions
{
```

```
    /// <summary>
    /// The source filter to use for X-translation.
    /// </summary>
    [DataMember]
    public ExpectedSourceStateBeforeXTranslate SourceFilter;
    /// <summary>
    /// If this member is true, segments with empty translation
    /// will also be applied during the X-translation. This option
    /// has effect only if the SourceFilter is AllRows.
    /// </summary>
    [DataMember]
    public bool InsertEmptyTranslations;
    /// <summary>
    /// The expected final translation state afte the X-translation.
    /// </summary>
    [DataMember]
    public ExpectedFinalStateAfterXTranslate ExpectedFinalState;
    /// <summary>
    /// If this member if true, the X-translated segments
    /// will be locked after the X-translation. This option
    /// has effect only if the ExpectedFinalState is not
    /// SameAsPrevious.
    /// </summary>
    [DataMember]
    public bool LockXTranslatedRows;

}
```

XTranslateOptions class

```
/// <summary>
/// Encapsulates options for X-translation on server project translation documents.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class XTranslateOptions
{
    /// <summary>
    /// The array of document infos which represent the
    /// documents to X-translate.
    /// </summary>
    /// <remarks>
    /// All of the documents should have version history
    /// and previous major version.
    /// </remarks>
    [DataMember]
    public XTranslateDocInfo[] DocInfos;
    /// <summary>
    /// The X-translate scenario to use for X-translation.
    /// </summary>
    [DataMember]
    public XTranslateScenario XTranslateScenario;
    /// <summary>
    /// If true memoQ will consider a current row as identical to
    /// a past row if if they both have the same context ID and
    /// the source text is the same.
    /// </summary>
    [DataMember]
    public bool WorkWithContextIds;
    /// <summary>
    /// Contains the options for the new revision scenario.
    /// This member should be null is the XTranslateScenario
    /// member if MidProjectUpdate.
    /// </summary>
```

```
    [DataMember]
    public NewRevisionScenarioOptions NewRevisionOptions;

}
```

XTranslateDocumentResult class

```
/// <summary>
/// Encapsulates the result of the X-translation
/// for a document.
/// </summary>
[DataContract]
public partial class XTranslateDocumentResult
{
    /// <summary>
    /// The guid of the document.
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The number of the candidates (rows identical in the document's new version).
    /// </summary>
    [DataMember]
    public int NumberOfCandidates;
    /// <summary>
    /// The number of the X-translated rows.
    /// </summary>
    [DataMember]
    public int NumberOfXTranslatedRows;
    /// <summary>
    /// The number of the rows of the document.
    /// </summary>
    [DataMember]
    public int NumberOfRows;

}
```

XTranslateResultInfo class

```
/// <summary>
/// Encapsulates the result of the X-translation.
/// </summary>
[DataContract]
public partial class XTranslateResultInfo
{
    /// <summary>
    /// The X-translation results of the documents.
    /// </summary>
    [DataMember]
    public XTranslateDocumentResult[] DocumentResults;

}
```

3.7.15 Asia Online

The **IServerProjectService** supports using Asia Online service for pre-translation. Asia Online support asynchronous translation. MemoQ server is able to upload documents to Asia Online services, wait for the translation process, and then download and update the document in memoQ server.

Asia Online requires configuration (such as a user name and password used to connect to Asia Online). This configuration requires memoQ client, and is available in the Server Administrator dialog.

Workflow of Asia Online translation

1. Configure Asia Online service in memoQ server using memoQ desktop client. Not supported through WS-API. (To be performed only once).
2. Get the identifiers of the documents to translate with Asia Online. Find the source and target languages of these documents.
3. Get the language pair codes (AsiaOnlineGetLanguagePairCode) for these documents. (Can be cached, does not change over time.)
4. List the supported domain combinations for the language pairs (AsiaOnlineGetDomainCombinations).
5. Start the translation (AsiaOnlineBeginTranslation).
6. Query the status of the translation periodically (AsiaOnlineGetTranslationStatus).

The operations and entity classes involved are the following:

- Getting the AsiaOnline language pair code for the specified source and target language pair
  - IServerProjectService.AsiaOnlineGetLanguagePairCode
  - AsiaOnlineGetLanguagePairCodeResultInfo
- Getting the supported domain combinations of the specified AsiaOnline language pair code
  - IServerProjectService.AsiaOnlineGetDomainCombinations
  - AsiaOnlineDomainCombination
  - AsiaOnlineGetDomainCombinationsResultInfo
- Submitting the specified documents of the server project to AsiaOnline services for asynchronous translation
  - IServerProjectService.AsiaOnlineBeginTranslation
  - AsiaOnlineTranslateOptions
  - AsiaOnlineBeginTranslationResultInfo
- Getting the status of Asia Online translation operations for a specified server project
  - IServerProjectService.AsiaOnlineGetTranslationStatus
  - AsiaOnlineTranslationStatus
  - AsiaOnlineTranslationResultInfo
- Getting the project IDs for the current user
  - IServerProjectService.AsiaOnlineGetProjectIds
  - AsiaOnlineGetProjectIdsResultInfo

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
```

```
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    /// <summary>
    /// Gets the language pair code, as used by Asia Online, for the specified
    /// source and target language pair.
    /// </summary>
    /// <param name="memoQSourceLangCode">The source language code, as language
    /// codes used by memoQ. The same as the source language of the project.
    /// </param>
    /// <param name="memoQTargetLangCode">The target language code, as language
    /// codes used by memoQ. The same as the target language of the project.
    ///</param>
    /// <returns>
    /// The AsiaOnlineGetLanguagePairCodeResultInfo object providing information
    /// about the operation. If the operation succeeded, contains the Asia Online
    /// language pair code.
    /// </returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    AsiaOnlineGetLanguagePairCodeResultInfo AsiaOnlineGetLanguagePairCode(
        string memoQSourceLangCode, string memoQTargetLangCode);
    /// <summary>
    /// Gets the supported domain combinations of the specified Asia Online
    /// language pair code.
    /// </summary>
    /// <param name="languagePairCode">The Asia Online language pair code.</param>
    /// <returns>
    /// The AsiaOnlineGetDomainCombinationsResultInfo object providing information
    /// about the operation. If the operation succeeded, contains the list of
    /// supported domain combinations.
    /// </returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    AsiaOnlineGetDomainCombinationsResultInfo AsiaOnlineGetDomainCombinations(
        int languagePairCode);
    /// <summary>
    /// Submits the specified documents of the server project to Asia Online
    /// translation. The translation is an asynchronous operation, this
    /// operation submits the documents to Asia Online services, but does not
    /// wait for the process to complete. Use AsiaOnlineGetTranslationStatus
    /// to query the status of the operation.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="documentGuids">The Guids of the documents
    /// to be pretranslated.</param>
    /// <param name="options">The options of the AO translate process.</param>
    /// <returns>
    /// The AsiaOnlineBeginTranslationResultInfo for each document, providing
    /// information about the success or failure of the operation for the
    /// particular document.
    /// </returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    AsiaOnlineBeginTranslationResultInfo[] AsiaOnlineBeginTranslation(
        Guid serverProjectGuid,
        Guid[] documentGuids, AsiaOnlineTranslateOptions options);
    /// <summary>
```

```
    /// Gets the status of Asia Online translation operations for a specific
    /// server project. Lists the status for each document that was submitted
    /// for translation to Asia Online in the specified server project.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <returns>
    /// An array or AsiaOnlineTranslationResultInfo objects, for each document
    /// in the server project that was submitted for translation to Asia Online.
    /// </returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    AsiaOnlineTranslationResultInfo[] AsiaOnlineGetTranslationStatus(
        Guid serverProjectGuid);
     /// <summary>
    /// Gets the project IDs belonging to the current user.
    /// </summary>
    /// <returns>
    /// An array of integers contains the project IDs belonging to a user.
    /// </returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    AsiaOnlineGetProjectIdsResultInfo AsiaOnlineGetProjectIds();
}
```

AsiaOnlineGetLanguagePairCodeResultInfo

```
/// <summary>
/// Encapsulates the result of AsiaOnlineGetLanguagePairCode operation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class AsiaOnlineGetLanguagePairCodeResultInfo : ResultInfo
{
    /// <summary>
    /// The identifier of the language pair, as known by Asia Online.
    /// </summary>
    [DataMember]
    public int AOLanguagePairId;
}
```

AsiaOnlineDomainCombination

```
/// <summary>
/// Describes an Asia Online domain combination.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class AsiaOnlineDomainCombination
{
    /// <summary>
    /// The code of the Asia Online domain combination.
    /// </summary>
    [DataMember]
    public long DomainCombinationCode;
    /// <summary>
    /// The name of the Asia Online domain combination.
    /// </summary>
    [DataMember]
    public string DomainCombinationName;
}
```

AsiaOnlineGetDomainCombinationsResultInfo

```
/// <summary>
/// Encapsulates the result of AsiaOnlineGetDomainCombinations operation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class AsiaOnlineGetDomainCombinationsResultInfo : ResultInfo
{
    /// <summary>
    /// The Asia Online language pair code.
    /// </summary>
    [DataMember]
    public int LanguagePairCode;
    /// <summary>
    /// The Asia Online domain combinations matching the queries language
    /// pair code.
    /// </summary>
    [DataMember]
    public AsiaOnlineDomainCombination[] DomainCombinations;
}
```

AsiaOnlineTranslateOptions

```
/// <summary>
/// Encapsulates options for performing Asia Online translate on server
/// project translation documents.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class AsiaOnlineTranslateOptions
{
    /// <summary>
    /// The domain combination code to use for translation.
    /// </summary>
    [DataMember]
    public long DomainCombinationCode;
    /// <summary>
    /// True to submit rows for translation by Asia Online, which are
    /// in edited status.
    /// </summary>
    [DataMember]
    public bool SubmitEditedRows;
    /// <summary>
    /// True to submit rows for translation by Asia Online, which are
    /// in not started or fragment assembled status.
    /// </summary>
    [DataMember]
    public bool SubmitNotStartedAndFragmentAssembledRows;
    /// <summary>
    /// True to submit rows for translation by Asia Online, which are
    /// in pre-translate status. Must specify pre-translation match
    /// threshold.
    /// </summary>
    [DataMember]
    public bool SubmitPreTranslatedRows;
    /// <summary>
    /// When submitting pre-translated rows for translation, specifies
    /// the match rate threshold below which rows are submitted for
    /// translation.
    /// </summary>
    [DataMember]
```

```
    public int SubmitPreTranslatedRowsMatchThreshold;
    /// <summary>
    /// The timeout (in hours) for the translation. The operation is
    /// cancelled if it does not finish within this time frame.
    /// </summary>
    [DataMember]
    public int JobTimeoutHours;
    /// <summary>
    /// The project number to use for translation.
    /// </summary>
    [DataMember]
    public int ProjectId;
}
```

AsiaOnlineBeginTranslationResultInfo

```
/// <summary>
/// Encapsulates the result of AsiaOnlineBeginTranslation operation
/// for each document.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class AsiaOnlineBeginTranslationResultInfo : ResultInfo
{
    // Class is empty by design.
}
```

AsiaOnlineTranslationStatus

```
/// <summary>
/// Statuses of an Asia Online translation operations for a single
/// document.
/// </summary>
public enum AsiaOnlineTranslationStatus
{
    /// <summary>
    /// The translation operation has not bee started yet.
    /// </summary>
    S00_NotStarted,
    /// <summary>
    /// Document has been exported by memoQ server and is waiting
    /// for upload to Asia Online.
    /// </summary>
    S01_Exported,
    /// <summary>
    /// Document has been uploaded to Asia Online and is queue
    /// for processing by AO.
    /// </summary>
    S02_Uploaded,
    /// <summary>
    /// Document has been translated by Asia Online and is queue
    /// for download.
    /// </summary>
    S03_MTranslated,
    /// <summary>
    /// Translated document has been downloaded from Asia Online
    /// and is in queue to update the document in memoQ server.
    /// </summary>
    S04_ResultDownloaded,
    /// <summary>
```

```
    /// Document has been udpated in memoQ server with the
    /// translations by Asia Online.
    /// </summary>
    S10_UpdatedInProject,
    /// <summary>
    /// Error status. Export of document has failed.
    /// </summary>
    S11_ExportFailed,
    /// <summary>
    /// Error status. Upload to Asia Online has failed.
    /// </summary>
    S12_UploadFailed,
    /// <summary>
    /// Error status. Translation has failed.
    /// </summary>
    S13_TranslationFailed,
    /// <summary>
    /// Error status. Download of translated result has failed.
    /// </summary>
    S14_DownloadFailed,
    /// <summary>
    /// Error status. Update of document in memoQ server has
    /// failed.
    /// </summary>
    S15_UpdateFailed,
    /// <summary>
    /// Error status. Translation process has not finished within
    /// expected time frame.
    /// </summary>
    S16_StoppedWaiting
};
```

AsiaOnlineTranslationResultInfo

```
/// <summary>
/// Encapsulates the status of an Asia Online transation operation
/// for a single document (known as a job in Asia Online).
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class AsiaOnlineTranslationResultInfo : ResultInfo
{
    /// <summary>
    /// The deadline of the translation process, or the date when the
    /// operation has completed.
    /// </summary>
    [DataMember]
    public DateTime DeadlineOrFinished;
    /// <summary>
    /// The name of the document.
    /// </summary>
    [DataMember]
    public string DocumentName;
    /// <summary>
    /// The target language of the document.
    /// </summary>
    [DataMember]
    public string TargetLang;
    /// <summary>
    /// The identifier of the document.
```

```
    /// </summary>
    [DataMember]
    public Guid DocumentId;
    /// <summary>
    /// If there has been an error reported by Asia Online,
    /// contains the message or the error.
    /// </summary>
    [DataMember]
    public string ErrorMessage;
    /// <summary>
    /// The status of the translation operation.
    /// </summary>
    [DataMember]
    public AsiaOnlineTranslationStatus Status;
}
```

AsiaOnlineGetProjectIdsResultInfo

```
/// <summary>
/// Encapsulates the result of AsiaOnlineGetProjectIds operation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class AsiaOnlineGetProjectIdsResultInfo : ResultInfo
{
    /// <summary>
    /// The project IDs belonging to the Asia Online account.
    /// </summary>
    [DataMember]
    public int[] AOProjectIds;
}
```

3.7.16 History

The **IServerProjectService** interface provides a function to be able to get the document history. The function and related entity classes are discussed in this chapter.

The operations and entity classes involved are the following:

- IServerProjectService.GetDocumentHistory
- DocumentHistoryRequest
- DocumentHistoryItemType
- DocumentHistoryItemInfo
- AssignmentChangeHistoryItemInfo
- DeadlineChangeHistoryItemInfo
- DocumentBilingualImportHistoryItemInfo
- DocumentDeliverHistoryItemInfo
- DocumentImportHistoryItemInfo
- DocumentReturnHistoryItemInfo
- DocumentSlicingHistoryItemInfo
- FirstAcceptAssignHistoryItemInfo
- FirstAcceptDeclineHistoryItemInfo

- FirstAcceptFailedHistoryItemInfo
- GroupSourcingAssignHistoryItemInfo
- GroupSourcingDocumentDeliverHistoryItemInfo
- SubvendorAssignDeadlineChangeHistoryItemInfo
- SubvendorAssignHistoryItemInfo
- WorkflowStatusChangeHistoryItemInfo
- DocumentXTranslationHistoryItemInfo
- DocumentRowsLockedHistoryItemInfo
- DocumentSnapshotCreatedHistoryItemInfo
- WorkingTMsDeletedHistoryItemInfo
- ProjectLaunchedHistoryItemInfo
- AutomatedActionStartedHistoryItemInfo

The involved **IServerProjectService** operations and related entity classes are described in details next: the C# code for the interfaces/classes with their description is listed.

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    /// <summary>
    /// Returns the document history items related to the documents
    /// or divisions in the specified project. The function returns
    /// the history information for all of the project documents or
    /// divisions if the DocumentGuids parameter of the request
    /// parameter is null. Otherwise the function gives back only the
    /// history items related to the specified documents or divisions.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="request">The document history details.</param>
    /// <returns>The history items related to the specified documents
    /// or divisions.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [ServiceKnownType(typeof(AssignmentChangeHistoryItemInfo))]
    [ServiceKnownType(typeof(DeadlineChangeHistoryItemInfo))]
    [ServiceKnownType(typeof(DocumentBilingualImportHistoryItemInfo))]
    [ServiceKnownType(typeof(DocumentDeliverHistoryItemInfo))]
    [ServiceKnownType(typeof(DocumentImportHistoryItemInfo))]
    [ServiceKnownType(typeof(DocumentReturnHistoryItemInfo))]
    [ServiceKnownType(typeof(DocumentSlicingHistoryItemInfo))]
    [ServiceKnownType(typeof(FirstAcceptAssignHistoryItemInfo))]
    [ServiceKnownType(typeof(FirstAcceptAcceptHistoryItemInfo))]
    [ServiceKnownType(typeof(FirstAcceptDeclineHistoryItemInfo))]
    [ServiceKnownType(typeof(FirstAcceptFailedHistoryItemInfo))]
    [ServiceKnownType(typeof(GroupSourcingAssignHistoryItemInfo))]
    [ServiceKnownType(typeof(GroupSourcingDocumentDeliverHistoryItemInfo))]
    [ServiceKnownType(typeof(SubvendorAssignDeadlineChangeHistoryItemInfo))]
    [ServiceKnownType(typeof(SubvendorAssignHistoryItemInfo))]
    [ServiceKnownType(typeof(WorkflowStatusChangeHistoryItemInfo))]
    [ServiceKnownType(typeof(DocumentXTranslationHistoryItemInfo))]
```

```
    [ServiceKnownType(typeof(DocumentRowsLockedHistoryItemInfo))]
    [ServiceKnownType(typeof(DocumentSnapshotCreatedHistoryItemInfo))]
    [ServiceKnownType(typeof(WorkingTMsDeletedHistoryItemInfo))]
    [ServiceKnownType(typeof(ProjectLaunchedHistoryItemInfo))]
    [ServiceKnownType(typeof(AutomatedActionStartedHistoryItemInfo))]
    DocumentHistoryItemInfo[] GetDocumentHistory(Guid serverProjectGuid,
          DocumentHistoryRequest request);
}
```

DocumentHistoryRequest class

```
/// <summary>
/// Represents a document history request.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class DocumentHistoryRequest
{
    /// <summary>
    /// The guids of the documents or divisions to which
    /// the history items should be returned. The null
    /// value means that all of the history items should
    /// be returned.
    /// </summary>
    [DataMember]
    public Guid[] DocumentGuids;
}
```

DocumentHistoryItemType enum

```
/// <summary>
/// Represents the possible document history item types.
/// </summary>
public enum DocumentHistoryItemType
{
    /// <summary>
    /// Assignment change document history item type.
    /// Used when the assigned users had been changed.
    /// </summary>
    AssignmentChange,
    /// <summary>
    /// Deadline change document history item type.
    /// Used when the document deadline had been changed.
    /// </summary>
    DeadlineChange,
    /// <summary>
    /// Document deliver history item type.
    /// Used when the user had delivered the document.
    /// </summary>
    DocumentDeliver,
    /// <summary>
    /// Document return history item type.
    /// Used when the user had sent back the document
    /// to the previous user.
    /// </summary>
    DocumentReturn,
    /// <summary>
    /// Workflow status change document history item type.
    /// Used when the status of the document had been changed.
    /// </summary>
    WorkflowStatusChange,
    /// <summary>
    /// FirstAccept assign document history item type.
```

```
/// Used when the document had been assigned with
/// FirstAccept to a group of the users.
/// </summary>
FirstAcceptAssign,
/// <summary>
/// FirstAccept accept document history item type.
/// Used when the document had been accepted by somebody.
/// </summary>
FirstAcceptAccept,
/// <summary>
/// FirstAccept decline document history item type.
/// Used when the document had been declined by somebody.
/// </summary>
FirstAcceptDecline,
/// <summary>
/// FirstAccept failed document history item type.
/// Used when the FirstAccept had failed.
/// </summary>
FirstAcceptFailed,
/// <summary>
/// GroupSourcing assign document history item type.
/// Used when the document had been assigned to a
/// group of the users with GroupSourcing.
/// </summary>
GroupSourcingAssign,
/// <summary>
/// GroupSourcing document deliver history item type.
/// Used when somebody had delivered the document
/// from the group.
/// </summary>
GroupSourcingDocumentDeliver,
/// <summary>
/// Subvendor assign document history item type.
/// Used when the document had been assigned to
/// a subvendor group.
/// </summary>
SubvendorAssign,
/// <summary>
/// Subvendor group deadline change history item type.
/// Used when the deadline for the assigned subvendor
/// gropu had been changed.
/// </summary>
SubvendorAssignDeadlineChange,
/// <summary>
/// Document slicing history item type.
/// Used when the document had been sliced.
/// </summary>
DocumentSlicing,
/// <summary>
/// Document reconsolidation history item type.
/// Used when the document had been reconsolidated.
/// </summary>
DocumentReconsolidation,
/// <summary>
/// Document import history item type.
/// Used when the document had been imported.
/// </summary>
DocumentImport,
/// <summary>
/// Document remove history item type.
/// Used when the document had been removed.
/// </summary>
```

```
    DocumentRemove,
    /// <summary>
    /// Document pre-translation history item type.
    /// Used when the document had been pre-translated.
    /// </summary>
    DocumentPreTranslation,
    /// <summary>
    /// Document bilingual import history item type.
    /// Used when the bilingual document had been imported.
    /// </summary>
    DocumentBilingualImport,
    /// <summary>
    /// Document to Asia Online history item type.
    /// Used when the document had been sent to
    /// the Asia Online.
    /// </summary>
    DocumentToAsiaOnline,
    /// <summary>
    /// Document from Asia Online history item type.
    /// Used when the document had been received from
    /// the Asia Online.
    /// </summary>
    DocumentFromAsiaOnline,
    /// <summary>
    /// Work started on project history item info.
    /// Used when a user had started his/her work
    /// on the project.
    /// </summary>
    WorkStartedOnProject
    /// <summary>
    /// Document analysis report created history
    /// item info. Used when an analysis report
    /// has been calculated.
    /// </summary>
    DocumentAnalysisReport,
    /// <summary>
    /// Document progress report created history
    /// item info. Used when a progress report
    /// has been calculated.
    /// </summary>
    DocumentProgressReport,
    /// <summary>
    /// Document post-translation analysis history
    /// item info. Used when a post-translation
    /// analysis has been calculated.
    /// </summary>
    DocumentPTA,
    /// <summary>
    /// Copy source to target history item info.
    /// Used when the source segments have been
    /// copied from the source to the target.
    /// </summary>
    DocumentCopySourceToTarget,
    /// <summary>
    /// X-translation history item info. Used
    /// when an X-tanslation has been performed.
    /// </summary>
    DocumentXTranslation,
    /// <summary>
    /// Edit distance statistic history item info.
    /// Used when an edit distance statistic has
    /// been calculated.
```

```
    /// </summary>
    DocumentEditDistanceStat,
    /// <summary>
    /// Reviewer edit distance statistic history
    /// item info. Used when an edit distance
    /// statistic has been calculated.
    /// </summary>
    DocumentReviewerEditDistanceStat,
    /// <summary>
    /// Confirm and update history item info.
    /// Used when a copy sorce to target
    /// operation has been performed.
    /// </summary>
    DocumentConfirmAndUpdateRows,
    /// <summary>
    /// Rows locked history item info. Used when
    /// the document's rows have been locked or
    /// unlocked.
    /// </summary>
    DocumentRowsLocked,
    /// <summary>
    /// Create snapshot history item info. Used
    /// when a snapshot has been created.
    /// </summary>
    DocumentSnapshotCreated,
    /// <summary>
    /// Delete working TMs history item info.
    /// Used when the working TMs has been
    /// deleted at project wrap up.
    /// </summary>
    WorkingTMsDeleted,
    /// <summary>
    /// Project launched history item info.
    /// Used when the project has been launched.
    /// </summary>
    ProjectLaunched,
    /// <summary>
    /// Project unlocked history item info. Used
    /// when the project has been unlocked for
    /// autopilot.
    /// </summary>
    ProjectUnlocked
    /// <summary>
    /// Document returned to source history item info.
    /// Used when a document has been returned to the
    /// content connector.
    /// </summary>
    DocumentReturnedToSource,
    /// <summary>
    /// Automated action started history item info.
    /// Used when an automated action has been started.
    /// </summary>
    AutomatedActionStarted
}
```

DocumentHistoryItemInfo class

```
/// <summary>
/// Encapsulates document history item information for a document.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
[KnownType(typeof(AssignmentChangeHistoryItemInfo))]
```

```
[KnownType(typeof(DeadlineChangeHistoryItemInfo))]
[KnownType(typeof(DocumentBilingualImportHistoryItemInfo))]
[KnownType(typeof(DocumentDeliverHistoryItemInfo))]
[KnownType(typeof(DocumentImportHistoryItemInfo))]
[KnownType(typeof(DocumentReturnHistoryItemInfo))]
[KnownType(typeof(DocumentSlicingHistoryItemInfo))]
[KnownType(typeof(FirstAcceptAssignHistoryItemInfo))]
[KnownType(typeof(FirstAcceptAcceptHistoryItemInfo))]
[KnownType(typeof(FirstAcceptDeclineHistoryItemInfo))]
[KnownType(typeof(FirstAcceptFailedHistoryItemInfo))]
[KnownType(typeof(GroupSourcingAssignHistoryItemInfo))]
[KnownType(typeof(GroupSourcingDocumentDeliverHistoryItemInfo))]
[KnownType(typeof(SubvendorAssignDeadlineChangeHistoryItemInfo))]
[KnownType(typeof(SubvendorAssignHistoryItemInfo))]
[KnownType(typeof(WorkflowStatusChangeHistoryItemInfo))]
[KnownType(typeof(DocumentXTranslationHistoryItemInfo))]
[KnownType(typeof(DocumentRowsLockedHistoryItemInfo))]
[KnownType(typeof(DocumentSnapshotCreatedHistoryItemInfo))]
[KnownType(typeof(WorkingTMsDeletedHistoryItemInfo))]
[KnownType(typeof(ProjectLaunchedHistoryItemInfo))]
[KnownType(typeof(AutomatedActionStartedHistoryItemInfo))]
public partial class DocumentHistoryItemInfo
{
    /// <summary>
    /// The guid of the user who did the operation. It can be an empty
    /// guid if the operation was not initiated by a single user.
    /// </summary>
    [DataMember]
    public Guid ModifierUserGuid;
    /// <summary>
    /// Gets the guid of the subvendor group of the modifier user
    /// if she is member of a subvendor group, otherwise it is an
    /// empty guid.
    /// </summary>
    [DataMember]
    public Guid ModifierSVGroupId;
    /// <summary>
    /// The time of the operation.
    /// </summary>
    [DataMember]
    public DateTime TimeStamp;
    /// <summary>
    /// The guid of the document or the division. It can be an
    /// empty guid if the history item is not belonging to a
    /// document.
    /// </summary>
    [DataMember]
    public Guid DocumentOrDivisionGuid;
    /// <summary>
    /// The name of the document or the division. It can be
    /// an empty string if the history item is not belonging
    /// to a document.
    /// </summary>
    [DataMember]
    public string DocumentOrDivisionName;
    /// <summary>
    /// The type of the document history item.
    /// </summary>
    [DataMember]
    public DocumentHistoryItemType HistoryItemType;
}
```

AssignmentChangeHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the document
/// assignment changes.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class AssignmentChangeHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The document assignment role.
    /// The currently accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
    /// <summary>
    /// The guid of the assigned user. It will be an empty
    /// guid if the document had been assigned to no one.
    /// </summary>
    [DataMember]
    public Guid NewUserGuid;
    /// <summary>
    /// The deadline for the assigned user. It will be
    /// the minimum value of the DateTime if the document
    /// had been assigned to no one.
    /// </summary>
    [DataMember]
    public DateTime Deadline;
}
```

DeadlineChangeHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the document
/// deadline changes.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class DeadlineChangeHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The document assignment role.
    /// The currently accepted values are the followings:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
    /// <summary>
    /// The new deadline for the assigned user.
    /// </summary>
    [DataMember]
    public DateTime NewDeadline;
}
```

DocumentBilingualImportHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
```

```
/// bilingual document imports.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class DocumentBilingualImportHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The name of the bilingual document format.
    /// </summary>
    [DataMember]
    public string BilingualFormat;
}
```

DocumentDeliverHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// document deliveries.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class DocumentDeliverHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The new workflow status of the document.
    /// </summary>
    [DataMember]
    public WorkflowStatus NewWorkflowStatus;
}
```

DocumentImportHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// document imports.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class DocumentImportHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// Indicate whether the document had been reimported.
    /// </summary>
    [DataMember]
    public bool Reimport;
}
```

DocumentReturnHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// document deliveries to the previous users.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class DocumentReturnHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The new workflow status of the document.
    /// </summary>
    [DataMember]
    public WorkflowStatus NewWorkflowStatus;
}
```

DocumentSlicingHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// document slicing operations.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class DocumentSlicingHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The number of the document parts.
    /// </summary>
    [DataMember]
    public int NumberOfParts;
}
```

FirstAcceptAssignHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// document assignments with FirstAccept.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class FirstAcceptAssignHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The document assignment role.
    /// The currently accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
    /// <summary>
    /// The guids of the assigned users.
    /// </summary>
    [DataMember]
    public Guid[] UserGuids;
}
```

FirstAcceptAcceptHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// FirstAccept accept operations.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class FirstAcceptAcceptHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The document assignment role.
    /// The currently accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
}
```

FirstAcceptDeclineHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// FirstAccept decline operations.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class FirstAcceptDeclineHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The document assignment role.
    /// The currently accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
    /// <summary>
    /// Indicates whether the FirstAccept had failed (all
    /// of the other assigned user had declined the
    /// document as well)
    /// </summary>
    [DataMember]
    public bool FirstAcceptFailed;
}
```

FirstAcceptFailedHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information which is will be created
/// when a FirstAccept had failed.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class FirstAcceptFailedHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The document assignment role.
    /// The currently accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
    /// <summary>
    /// The deadline for the FirstAccept.
    /// </summary>
    [DataMember]
    public DateTime Deadline;
}
```

GroupSourcingAssignHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// document assignments with GroupSourcing.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class GroupSourcingAssignHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The document assignment role.
    /// The currently accepted values are the following:
```

```
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
    /// <summary>
    /// The guids of the assigned users.
    /// </summary>
    [DataMember]
    public Guid[] UserGuids;
}
```

GroupSourcingDocumentDeliverHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// document delivery operation when the assignment
/// type is GroupSourcing.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class GroupSourcingDocumentDeliverHistoryItemInfo :
DocumentHistoryItemInfo
{
    /// <summary>
    /// The document assignment role.
    /// The currently accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
    /// <summary>
    /// Indicates whether all of the document's segments are
    /// confirmed/proofread.
    /// </summary>
    [DataMember]
    public bool DocumentComplete;
}
```

SubvendorAssignDeadlineChangeHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// deadline changes when the document had been assigned
/// to a subvendor group.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class SubvendorAssignDeadlineChangeHistoryItemInfo :
DocumentHistoryItemInfo
{
    /// <summary>
    /// The document assignment role.
    /// The currently accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
    /// <summary>
```

```
    /// The deadline for the subvendor group.
    /// </summary>
    [DataMember]
    public DateTime Deadline;
}
```

SubvendorAssignHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// subvendor assignment changes.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class SubvendorAssignHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The document assignment role.
    /// The currently accepted values are the following:
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
    /// <summary>
    /// The guid of the assigned subvendor group.
    /// </summary>
    [DataMember]
    public Guid SubvendorGroupId;
    /// <summary>
    /// The deadline for the subvendor group.
    /// </summary>
    [DataMember]
    public DateTime Deadline;
}
```

WorkflowStatusChangeHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// workflow change operations.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class WorkflowStatusChangeHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// The old workflow status of the document.
    /// </summary>
    [DataMember]
    public WorkflowStatus OldWorkflowStatus;
    /// <summary>
    /// The new workflow status of the document.
    /// </summary>
    [DataMember]
    public WorkflowStatus NewWorkflowStatus;
}
```

DocumentXTranslationHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// X-translation operations.
```

```
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class DocumentXTranslationHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// Indicated whether the x-translate scenario
    /// is mid-project update or not.
    /// </summary>
    [DataMember]
    public bool MidProjectUpdate;
}
```

DocumentRowsLockedHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// lock document rows operations.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class DocumentRowsLockedHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// Describes the possible lock modes.
    /// </summary>
    public enum LockModes
    {
        /// <summary>
        /// Lock all rows.
        /// </summary>
        AllRows,
        /// <summary>
        /// Lock only specific rows.
        /// </summary>
        SpecificRows,
        /// <summary>
        /// Lock rows in differen language.
        /// </summary>
        RowInDifferentLanguage
    }

    /// <summary>
    /// Indicates wheter the segments have been
    /// locked or unlocked.
    /// </summary>
    [DataMember]
    public bool LockSegments;

    /// <summary>
    /// Gets the lock mode.
    /// </summary>
    [DataMember]
    public LockModes LockMode;

    /// <summary>
    /// Indicates whether lock translator confirmed segments.
    /// It contains valid data only if the LockMode is
    /// LockMode.SpecificRows.
    /// </summary>
    [DataMember]
    public bool LockTranslatorConfirmedSegments;
```

```
    /// <summary>
    /// Indicates whether lock reviewer 1 confirmed segments.
    /// It contains valid data only if the LockMode is
    /// LockMode.SpecificRows.
    /// </summary>
    [DataMember]
    public bool LockReviewer1ConfirmedSegments;

    /// <summary>
    /// Indicates whether lock reviewer 2 confirmed segments.
    /// It contains valid data only if the LockMode is
    /// LockMode.SpecificRows.
    /// </summary>
    [DataMember]
    public bool LockReviewer2ConfirmedSegments;

    /// <summary>
    /// Indicates whether lock repetitions.
    /// It contains valid data only if the LockMode is
    /// LockMode.SpecificRows.
    /// </summary>
    [DataMember]
    public bool LockRepetitions;

    /// <summary>
    /// Indicates whether lock non-repetitions.
    /// It contains valid data only if the LockMode is
    /// LockMode.SpecificRows.
    /// </summary>
    [DataMember]
    public bool LockNonRepetitions;

    /// <summary>
    /// Indicates whether lock pre-translated segments.
    /// It contains valid data only if the LockMode is
    /// LockMode.SpecificRows.
    /// </summary>
    [DataMember]
    public bool LockPreTranslatedSegments;
}
```

DocumentSnapshotCreatedHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// create document snapshot operations.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class DocumentSnapshotCreatedHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// Gets the number of the created minor version.
    /// </summary>
    [DataMember]
    public int MinorVersion;
}
```

WorkingTMsDeletedHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// delete working TM operations.
```

```
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class WorkingTMsDeletedHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// Gets the names of the deleted TMs.
    /// </summary>
    public string[] DeletedTMNames;
}
```

ProjectLaunchedHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// project launch operations.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ProjectLaunchedHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// Indicates whether the distribution file
    /// has been attached.
    /// </summary>
    public bool DistributionFileAttached;
}
```

AutomatedActionStartedHistoryItemInfo class

```
/// <summary>
/// Encapsulates history information related to the
/// automated actions.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class AutomatedActionStartedHistoryItemInfo : DocumentHistoryItemInfo
{
    /// <summary>
    /// Describes the possible automated action types.
    /// </summary>
    public enum AutomatedActionTypes
    {
        /// <summary>
        /// Create analysis report action.
        /// </summary>
        CreateAnalysisReport,
        /// <summary>
        /// Pre-translate action.
        /// </summary>
        PreTranslate,
        /// <summary>
        /// Lock segments action.
        /// </summary>
        LockSegments,
        /// <summary>
        /// Create snapshot action.
        /// </summary>
        CreateSnapshot,
        /// <summary>
        /// Copy source to target action.
        /// </summary>
        CopySourceToTargetWhereEmpty,
        /// <summary>
        /// Run post-translation analysis action.
```

```
        /// </summary>
        RunPostTranslationAnalysis,
        /// <summary>
        /// X-translate action.
        /// </summary>
        XTranslate,
        /// <summary>
        /// Confirm and updte action.
        /// </summary>
        ConfirmAndUpdate,
        /// <summary>
        /// Delete working TMs action.
        /// </summary>
        DeleteWorkingTMs,
        /// <summary>
        /// Create progress report action.
        /// </summary>
        CreateProgressReport,
        /// <summary>
        /// Auto-assign users action.
        /// </summary>
        AutoAssignUsersToDocuments,
        /// <summary>
        /// Launch project action.
        /// </summary>
        LaunchProject,
        /// <summary>
        /// Return to source action.
        /// </summary>
        ReturnToSource,
        /// <summary>
        /// Put up for FirstAccept action.
        /// </summary>
        PutUpForFirstAccept,
        /// <summary>
        /// Calculate edit distance action.
        /// </summary>
        CalculateEditDistance,
        /// <summary>
        /// Calculate reviewer edit distance action.
        /// </summary>
        CalculateReviewerEditDistance,
        /// <summary>
        /// Send to Asia Online action.
        /// </summary>
        SendToAO
    }

    /// <summary>
    /// Gets the type of the automated action.
    /// </summary>
    public AutomatedActionTypes ActionType;
}
```

3.7.17 Package management and delivering packages

The **IServerProjectService** operation for packages, listing, creating, importing, delivering packages, and related entity classes are discussed in this chapter.

There are two types of packages: handoff packages and delivery packages. The first package contains the work sent to translators, while deliveries are the result received from the translators. Handoff packages are created automatically for projects that allow the package workflow. These packages can be listed and downloaded. Delivery packages can be imported into these projects receiving the translation.

Handoff packages can also be imported into the server and a new server project created from the package. Projects can be updates from update packages. Such projects behave as regular projects, and allow creating delivery packages from sending the work back.

The operation and entity classes involved are the following:

- Managing packages and deliveries
  - IServerProjectService.ListPackagesForProject
  - IServerProjectService.ListPackagesForProjectAndUser
  - PackageInfo
  - IServerProjectService.PreparePackageForDownload
  - PreparePackageResultInfo
  - IServerProjectManager.ListContentOfPackage
  - PackageContentInfo
  - IServerProjectManager.DeliverPackage
  - PackageDeliveryResultInfo
  - IServerProjectManager.CreateProjectFromPackage2
  - IServerProjectManager.UpdateProjectFromPackage
  - CreateDeliveryResult
  - IServerProjectManager.CreateDeliveryPackage

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    ...
    /// <summary>
    /// Lists all packages available for download in the specified server project.
    /// The project must support package creation, otherwise an exception is
    /// thrown. Does not list any packages until the project has been launched.
    /// </summary>
    /// <param name="projectGuid">The identifier of the project.</param>
    /// <returns>The list of all packages in the project.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    PackageInfo[] ListPackagesForProject(Guid projectGuid);
    /// <summary>
    /// Lists all packages available for download in the specified server project
    /// created for the specified user.
    /// The project must support package creation, otherwise an exception is
    /// thrown. Does not list any packages until the project has been launched.
    /// </summary>
    /// <param name="projectGuid">The identifier of the project.</param>
    /// <param name="userGuid">The identifier of the user to list the packages
    /// for.</param>
    /// <returns>The list of packages in the project for the user.</returns>
    [OperationContract]
```

```
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    PackageInfo[] ListPackagesForProjectAndUser(Guid projectGuid, Guid userGuid);
    /// <summary>
    /// Prepares a package for download. The result contains the identifier of
    /// the package which can be downloaded using FileManagerService. The result
    /// of the preparation can be that no package is created (in case there has
    /// been some changes in the project that were undone in the meantime and
    /// there is no change for the user). In this case no file is available for
    /// download. The package should be downloaded separately if this operation
    /// succeeds and reports that the package is available.
    /// </summary>
    /// <param name="packageId">The identifier of the package.</param>
    /// <returns>A result info which signals if there is a package available
    /// for download, and if so, contains its identifier.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    PreparePackageResultInfo PreparePackageForDownload(Guid packageId);
    /// <summary>
    /// Gets the content of a package. Includes the list of documents (new or
    /// changed) and the deleted documents.
    /// </summary>
    /// <param name="packageIds">The list of package identifiers for which to
    /// get the content.</param>
    /// <returns>A a PackageContentInfo object for each package.</returns>
    [OperationContract]
    [FaultContract(typeof (UnexpectedFault)), FaultContract(typeof (GenericFault))]
    PackageContentInfo[] ListContentOfPackages(Guid[] packageIds);
    /// <summary>
    /// Delivers a package, processes its content, and reports the result of the
    /// delivery. The delivery package must be uploaded using FileManagerService
    /// and the identifier of the file returned by FileManagerService given as
    /// the fileGuid. The result of the delivery can be failure for various
    /// reasons globally, and also on a document level. Detailed report is returned
    /// by the operation.
    /// </summary>
    /// <param name="fileGuid">The identifier of the delivery file uploaded using
    /// FileManagerService.</param>
    /// <returns>A packageDeliveryResultInfo object which describes the result
    /// of the delivery in details.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    PackageDeliveryResultInfo DeliverPackage(Guid fileGuid);

    /// <summary>
    /// Creates a new project from a package. The documents from the package are
    /// imported into the newly created project.
    /// </summary>
    /// <param name="createInfo">The class encapsulating every information required
    /// to create a new server project.
    /// The user identified by the CreatorUser member of the spInfo parameter
    /// is automatically added to the project with the Project Manager role.
    /// </param>
    /// <param name="fileId">The file id of the package to import. Should be
    /// uploaded usign FileManagerService. The package must be an initial handoff
    /// package and cannot be an update package.</param>
    /// <returns>The guid of the newly created server project.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    Guid CreateProjectFromPackage2(ServerProjectDesktopDocsCreateInfo createInfo, G
uid fileId);
    /// <summary>
    /// Import an update package and updates an existing project.The update package
    /// must match the project (the project must have been created from a package
    /// for which this update refers to).
    /// </summary>
    /// <param name="projectGuid">The identifier of the project to update.</param>
    /// <param name="fileId">The file id of the update package to import. Should be
    /// uploaded usign FileManagerService.</param>
```

```
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    void UpdateProjectFromPackage(Guid projectGuid, Guid fileId);
    /// <summary>
    /// Creates a delivery package from a project that was created from a package.
    /// The specified documents are exported as a delivery package.
    /// </summary>
    /// <param name="projectGuid">The identifier of the project.</param>
    /// <param name="documentGuids">The identifiers of the documents to include in
    /// the delivery.</param>
    /// <param name="returnToPreviousActor">True to return the documents to the
    /// previous actor for review; false to deliver the documents and move them to
    /// the next workflow state.</param>
    /// <returns>An object describing the results of the operation.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    CreateDeliveryResult CreateDeliveryPackage(Guid projectGuid,
        Guid[] documentGuids, bool returnToPreviousActor);
    ...
}
```

PackageInfo class

```
/// <summary>
/// Describes a package in a project that allows creating packages.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PackageInfo
{
    /// <summary>
    /// The unique identifier of the package.
    /// </summary>
    [DataMember]
    public Guid PackageID;
    /// <summary>
    /// The identifier of the server project this package belongs to.
    /// </summary>
    [DataMember]
    public Guid ProjectGuid;
    /// <summary>
    /// The identifier of the user for whom this package is created.
    /// </summary>
    [DataMember]
    public Guid UserGuid;
    /// <summary>
    /// The version / sequence number of the package. This incremental
    /// number
    /// </summary>
    [DataMember]
    public int SequenceNumber;
    /// <summary>
    /// The date when this package was created. Can be null if the
    /// package has not been created yet (the project contains
    /// changes for the user, but the package has not been assembled
    /// yet).
    /// </summary>
    [DataMember]
    public DateTime? CreationTime;
    /// <summary>
    /// The date when this package was first downloaded. Null if the
    /// package has not been downloaded yet.
    /// </summary>
    [DataMember]
    public DateTime? DownloadedTime;
}
```

PreparePackageResultInfo class

```
/// <summary>
/// The result of a PreparePackageForDownload operation.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PreparePackageResultInfo
{
    /// <summary>
    /// True if the package is ready for download. False if the package
    /// contains no information and there is nothing for download.
    /// </summary>
    [DataMember]
    public bool PackageReady;
    /// <summary>
    /// The identifier of the package for download. Contains a valid
    /// identifier only if PackageReady is true.
    /// </summary>
    [DataMember]
    public Guid? ResultFileId;
}
```

PackageContentInfo class

```
/// <summary>
/// Describes a package in a project that allows creating packages.
/// Contains detailed information about the content of the package,
/// if available.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PackageContentInfo : PackageInfo
{
    /// <summary>
    /// Detailed information about new and updated documents.
    /// Can be null if the package has not been created yet (the
    /// project contains changes for the user, but the package has
    /// not been assembled yet).
    /// </summary>
    [DataMember]
    public PackageContentDocument[] Documents;
}
```

PackageContentDocument class

```
/// <summary>
/// Describes a document in a package.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PackageContentDocument
{
    /// <summary>
    /// The identifier of the document.
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The name of the document.
    /// </summary>
    [DataMember]
    public string DocumentName;
    /// <summary>
    /// The target language fo the document.
    /// </summary>
    [DataMember]
    public string TargetLang;
    /// <summary>
    /// The deadline for the user.
```

```
    /// </summary>
    [DataMember]
    public DateTime Deadline;
    /// <summary>
    /// The assignment role of the user.
    /// 0: Translator
    /// 1: Reviewer1
    /// 2: Reviewer2
    /// </summary>
    [DataMember]
    public int Role;
}
```

PackageDeliveryResultInfo enum

```
/// <summary>
/// Describes the result of a delivery.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PackageDeliveryResultInfo
{
    /// <summary>
    /// Detailed reports for each document in the delivery package. Filled only
    /// when the DeliveryResult is accepted.
    /// </summary>
    [DataMember]
    public DocDeliveryResultInfo[] DocumentResults;
    /// <summary>
    /// The result of the delivery.
    /// </summary>
    [DataMember]
    public PackageDeliveryResult DeliveryResult;
}
```

PackageDeliveryResult class

```
/// <summary>
/// Describes the result of the delivery of an entire package.
/// </summary>
public enum PackageDeliveryResult
{
    /// <summary>
    /// The delivery was accepted; some problems and rejections are still
    /// possible.
    /// </summary>
    Accepted,
    /// <summary>
    /// Package was invalid and was rejected.
    /// </summary>
    InvalidPackageRejected,
    /// <summary>
    /// The project does not allow partial delivery but the package did
    /// not contain all necessary document; the entire delivery was
    /// rejected.
    /// </summary>
    PartialDeliveryRejected
}
```

DocDeliveryResultInfo class

```
/// <summary>
/// The result of the delivery of a single document in a delivery
/// package.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class DocDeliveryResultInfo
{
```

```
    /// <summary>
    /// The identifier of the document.
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The name of the document.
    /// </summary>
    [DataMember]
    public string DocumentName;
    /// <summary>
    /// The target language of the document.
    /// </summary>
    [DataMember]
    public string TargetLang;
    /// <summary>
    /// The result of the delivery.
    /// </summary>
    [DataMember]
    public DocDeliveryResult DeliveryResult;
}
```

DocDeliveryResult enum

```
/// <summary>
/// Describes the result of the delivery for a single document
/// in a delivery package.
/// </summary>
public enum DocDeliveryResult
{
    /// <summary>
    /// Delivery was succesfully.
    /// </summary>
    Success,
    /// <summary>
    /// Delivery failed because the mqxliff content was invalid.
    /// </summary>
    InvalidXliff,
    /// <summary>
    /// Delivery failed because the document is no longer assigned
    /// to the user (possibly removed from the project), or the
    /// worfklow state is different that the role of the current
    /// user (hence someone else is working on the document).
    /// </summary>
    RejectedNotInWorkflow
}
```

CreateDeliveryResult class

```
/// <summary>
/// Describes the result of a CreateDeliveryPackage operation.
/// </summary>
public class CreateDeliveryResult : ResultInfo
{
    /// <summary>
    /// Guids of the documents that cannot be delivered due to
    /// not being finished.
    /// </summary>
    public Guid[] DocumentsNotFinished;
    /// <summary>
    /// The id of the created delivery package. Not filled if
    /// the result is not success.
    /// </summary>
    public Guid FileId;
}
```

3.7.18 Running QA

The **IServerProjectService** operation for running QA checks and retrieving the report, and related entity classes are discussed in this chapter.

The operation and entity classes involved are the following:

- Run QA and get QA report
  - IServerProjectService.RunQAGetReport
  - RunQAGetReportOptions
  - QAReport
  - QAReportForDocument

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    ...
    /// <summary>
    /// Executed QA checks on the specified scope, collects errors and warnings,
    /// and returns a report of the errors and warnings present in the specified
    /// scope.
    /// </summary>
    /// <param name="serverProjectGuid">The unique identifier of the project.
    /// </param>
    /// <param name="options">Detailed options of the operation.</param>
    /// <returns>An object that contains an overview report for each document
    /// in the scope, and optionally a detailed report in the requested format
    /// (e.g. HTML report).</returns>
    [OperationContract]
    QAReport RunQAGetReport(Guid serverProjectGuid, RunQAGetReportOptions options);
    ...
}
```

QAReportTypes class

```
public enum QAReportTypes
{
    /// <summary>
    /// No report is requested, only run QA.
    /// </summary>
    None,
    /// <summary>
    /// An HTML report is requested. The retured result is a fully
    /// formatted HTML document. The report contains all the QA
    /// problems in the scope.
    /// </summary>
    Html
}
```

RunQAGetReportOptions class

```
/// <summary>
/// Options of running QA and getting a report.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class RunQAGetReportOptions
{
    /// <summary>
```

```
    /// The guids of the documents or slices for which QA should be
    /// executed. Null value means to run QA on the entire project.
    /// The report will only contain data for these documents.
    /// </summary>
    [DataMember]
    public Guid[] DocumentGuids;
    /// <summary>
    /// True to inclde locked rows; if false, all locked rows are
    /// skipped and not checked.
    /// </summary>
    [DataMember]
    public bool IncludeLockedRows;
    /// <summary>
    /// The type of report to return after the QA process.
    /// </summary>
    [DataMember]
    public QAReportTypes ReportType;
}
```

QAReport class

```
/// <summary>
/// The report after running QA.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class QAReport
{
    /// <summary>
    /// Overview report for each document (for all documents the
    /// operation is executed on).
    /// </summary>
    [DataMember]
    public QAReportForDocument[] ReportsPerDocument;
    /// <summary>
    /// Detailed report in the requested format; null if no
    /// report is requested..
    /// </summary>
    [DataMember]
    public string DetailedReport;
}
```

QAReportForDocument class

```
/// <summary>
/// The report after running QA.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class QAReportForDocument
{
    /// <summary>
    /// The name of the document.
    /// </summary>
    [DataMember]
    public string DocumentName;
    /// <summary>
    /// The target language of the document.
    /// </summary>
    [DataMember]
    public string TargetLanguage;
    /// <summary>
    /// The unique id of the document.
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
```

```
    /// The number of errors in the document.
    /// </summary>
    [DataMember]
    public int NumberOfErrors;
    /// <summary>
    /// The number of warnings in the document
    /// (including ignored ones).
    /// </summary>
    [DataMember]
    public int NumberOfWarnings;
    /// <summary>
    /// The number of warnings in the document
    /// (excluding ignored ones).
    /// </summary>
    [DataMember]
    public int NumberOfUnignoredWarnings;
}
```

3.7.19 Running post-translation analysis

The **IServerProjectService** operation for post-translation analysis and related entity classes are discussed in this chapter.

Post-translation analysis are reports stored in the project. Reports can be created on demand and existing reports can be listed and downloaded as entities as well as CSV exports.

The operation and entity classes involved are the following:

- Run, and retrieve the result of post-translation analysis
  - IServerProjectService.ListPostTranslationAnalysisReports
  - IServerProjectService.GetPostTranslationAnalysisReportData
  - IServerProjectService.GetPostTranslationAnalysisReportAsCSV
  - IServerProjectService.RunPostTranslationAnalysis
  - PostTranslationAnalysisReportInfo
  - PostTranslationResultForLang
  - PostTransAnalysisReportItem
  - PostTransAnalysisReportForUser
  - PostTransAnalysisReportForDocument
  - PostTranslationReportCounts
  - PostTranslationAnalysisOptions
  - PostTranslationAnalysisAsCSVResultForLang

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    ...
    /// <summary>
    /// Lists post translation analysis reports created earlier.
```

```
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <returns>An array of PostTranslationAnalysisReportInfo objects where
    /// each item describes a separate report and its metadata, including
    /// the unique identifier of the report which can be used to idenrify
    /// the report (e.g. for downloading). If the project contains no
    /// post translation analysis reports, an empty array is returned.</returns>
    [OperationContract]
    [FaultContract(typeof (UnexpectedFault)), FaultContract(typeof (GenericFault))]
    PostTranslationAnalysisReportInfo[] ListPostTranslationAnalysisReports(
        Guid serverProjectGuid);
    /// <summary>
    /// Gets a post-translation analysis report.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="reportId">The unique identifier of the report.</param>
    /// <returns>An object that encapsulates the report itself. The report is
    /// decomposed into per-language data, for each language the report was
    /// created for.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    PostTranslationAnalysisResultInfo GetPostTranslationAnalysisReportData(
        Guid serverProjectGuid, Guid reportId);
    /// <summary>
    /// Exports a post translation analysis report as a CSV and gets the
    /// results.
    /// </summary>
    /// <param name="serverProjectGuid"><The guid of the server project.</param>
    /// <param name="reportId">The unique identifier of the report.</param>
    /// <returns>An object that contains the exported report data for each
    /// language the report contains.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    PostTranslationAnalysisAsCSVResult GetPostTranslationAnalysisReportAsCSV(
        Guid serverProjectGuid, Guid reportId);
    /// <summary>
    /// Runs post-translation analysis on a project and get the resulting report.
    /// </summary>
    /// <param name="serverProjectGuid">The guid of the server project.</param>
    /// <param name="options">An object that describes the options of creating
    /// the report.</param>
    /// <returns>An object that describes the result of the operation and
    /// contains the report data.</returns>
    [OperationContract]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    PostTranslationAnalysisResultInfo RunPostTranslationAnalysis(Guid serverProjectGuid,
        PostTranslationAnalysisOptions options);
    ...
}
```

PostTranslationAnalysisReportInfo class

```
/// <summary>
/// Describes meta information of a post-translation analysis report.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PostTranslationAnalysisReportInfo
{
    /// <summary>
    /// The unique identifier of this report.
    /// </summary>
    [DataMember]
    public Guid ID;
    /// <summary>
    /// The data and time when this report was created.
    /// </summary>
```

```
    [DataMember]
    public DateTime Created;
    /// <summary>
    /// The creator user of the report.
    /// </summary>
    [DataMember]
    public string CreatedBy;
    /// <summary>
    /// The note submitted when creating the report.
    /// </summary>
    [DataMember]
    public string Note;
    /// <summary>
    /// The languages the report is calculated for.
    /// </summary>
    [DataMember]
    public string[] Languages;
}
```

PostTranslationAnalysisOptions class

```
/// <summary>
/// Options of creating a post-translation analysis report.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PostTranslationAnalysisOptions
{
    /// <summary>
    /// The language codes of the project to create report for. When specified,
    /// the project documents are filtered for these languages and only the
    /// documents for this language are included in the report. If not specified
    /// all languages of the project are used. Empty array should not be used.
    /// </summary>
    [DataMember]
    public string[] LanguageCodes;
    /// <summary>
    /// The guids of the documents to create report for. When specified,
    /// only these documents are included in the report. If not specified
    /// all documents of the project are used.  Empty array should not be used.
    /// </summary>
    [DataMember]
    public Guid[] DocumentGuids;
    /// <summary>
    /// True to save the report in the project after creation. This report can
    /// later be retrieved. False creates and returns the report, but does not
    /// store it with the project,
    /// </summary>
    [DataMember]
    public bool StoreReportInProject;
    /// <summary>
    /// A note stored with the report; when the report is saved in the project.
    /// </summary>
    [DataMember]
    public string Note;
    /// <summary>
    /// The weight of one tag in chars. Added to the number of chars.
    /// </summary>
    [DataMember]
    public double TagWeightChar;
    /// <summary>
    /// The weight of one tag in chars. Added to the number of chars.
    /// </summary>
    [DataMember]
    public double TagWeightWord;
    /// <summary>
    /// If true, repetitive 100% matches are counted as repetitions; if
```

```
    /// false, they are counted as 100% matches. Default behavior is true.
    public bool RepetitionPreferenceOver100 = true;
}
```

PostTranslationAnalysisAsCSVResult class

```
/// <summary>
/// The CSV export of a post translation analysis. Contains the reports
/// for all languages the report was created for.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PostTranslationAnalysisAsCSVResult
{
    /// <summary>
    /// The array of post translation analysis results: the array has
    /// one element for each target language.
    /// </summary>
    [DataMember]
    public PostTranslationAnalysisAsCSVResultForLang[] DataForTargetLangs;
}
```

PostTranslationAnalysisAsCSVResultForLang class

```
/// <summary>
/// The CSV export of a post translation analysis for a single target
/// language.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PostTranslationAnalysisAsCSVResultForLang
{
    /// <summary>
    /// The language this report is for.
    /// </summary>
    [DataMember]
    public string LanguageCode;
    /// <summary>
    /// The content of the exported CSV in Unicode encoding.
    /// </summary>
    [DataMember]
    public byte[] ExportedContent;
}
```

PostTranslationAnalysisResultInfo class

```
/// <summary>
/// The result of a post-translation analysis query. Contains the report data.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PostTranslationAnalysisResultInfo : ResultInfo
{
    /// <summary>
    /// The array of post translation analysis results: the array has
    /// one element for each target language. If the operations finished
    /// with errors, the value is not defined.
    /// </summary>
    [DataMember]
    public PostTranslationResultForLang[] ResultsForTargetLangs;
}
```

PostTranslationResultForLang class

```
/// <summary>
/// Encapsulates the result of a post translation analysis operation
/// for a single target language.
/// </summary>
```

```
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PostTranslationResultForLang
{
    /// <summary>
    /// The target language this report is for.
    /// </summary>
    [DataMember]
    public string TargetLangCode;
    /// <summary>
    /// Summary of the counts in this language.
    /// </summary>
    [DataMember]
    public PostTransAnalysisReportItem Summary;
    /// <summary>
    /// The counts per user. The array holds a single element for
    /// each translator user in the project who has translated at
    /// least one row in this language.
    /// </summary>
    [DataMember]
    public PostTransAnalysisReportForUser[] ByUser;
    /// <summary>
    /// The counts per document, detailed by users. The array holds
    /// and element for each document.
    /// </summary>
    [DataMember]
    public PostTransAnalysisReportForDocument[] ByDocument;
}
```

PostTransAnalysisReportForUser class

```
/// <summary>
/// Encapsulates a post-translation analysis report for a single user.
/// Holds the counts the user has translated.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PostTransAnalysisReportForUser : PostTransAnalysisReportItem
{
    /// <summary>
    /// The name of the user.
    /// </summary>
    [DataMember]
    public string Username;
}
```

PostTransAnalysisReportForDocument class

```
/// <summary>
/// Encapsulates a post-translation analysis report for a single document,
/// and details the user's contributions one by one.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PostTransAnalysisReportForDocument
{
    /// <summary>
    /// The unique identifier of the document.
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// Summary of the counts in this document.
    /// </summary>
    [DataMember]
    public PostTransAnalysisReportItem Summary;
    /// <summary>
    /// The counts per user. The array holds a single element for
```

```
    /// each translator user in the project who has translated at
    /// least one row in this document.
    /// </summary>
    [DataMember]
    public PostTransAnalysisReportForUser[] ByUser;
}
```

PostTransAnalysisReportItem class

```
/// <summary>
/// Details segment counts in a post-translation analysis report
/// for a given scope (depends on the encapsulating class).
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PostTransAnalysisReportItem
{
    /// <summary>
    /// All rows.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts All;
    /// <summary>
    /// XTranslated rows.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts XTranslated;
    /// <summary>
    /// Auto-propagated rows.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts AutoPropagated;
    /// <summary>
    /// Rows with 101% hits.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts Hit101;
    /// <summary>
    /// Rows with 100% hits.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts Hit100;
    /// <summary>
    /// Rows with 95-99% hits.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts Hit95_99;
    /// <summary>
    /// Rows with 85-94% hits.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts Hit85_94;
    /// <summary>
    /// Rows with 75-84% hits.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts Hit75_84;
    /// <summary>
    /// Rows with 50-74% hits.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts Hit50_74;
    /// <summary>
    /// Rows with fragment hits.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts Fragments;
```

```
    /// <summary>
    /// Rows with no hits.
    /// </summary>
    [DataMember]
    public PostTranslationReportCounts NoMatch;
}
```

PostTranslationReportCounts class

```
/// <summary>
/// Encapsulates segment counts in a post-translation analysis report
/// for a specific category of rows.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class PostTranslationReportCounts
{
    /// <summary>
    /// Number of segments.
    /// </summary>
    [DataMember]
    public int SegmentCount;
    /// <summary>
    /// Number of characters on the source side, including tag weights.
    /// </summary>
    [DataMember]
    public int SourceCharCount;
    /// <summary>
    /// Number of words on the source side, including tag weights.
    /// </summary>
    [DataMember]
    public int SourceWordCount;
    /// <summary>
    /// Number of tags on the source side.
    /// </summary>
    [DataMember]
    public int SourceTagCount;
    /// <summary>
    /// Number of non-Asian words on the source side.
    /// </summary>
    [DataMember]
    public int SourceNonAsianWordCount;
    /// <summary>
    /// Number of Asian words on the source side.
    /// </summary>
    [DataMember]
    public int SourceAsianCharCount;
}
```

3.7.20 Image localization packages

The **IServerProjectService** operation for creating and image localization package and importing it back, and related entity classes are discussed in this chapter.

The operation and entity classes involved are the following:

- Create image localization package and import localization package
  - IServerProjectService. CreateImageLocalizationPack
  - IServerProjectService. ImportImageLocalizationPack
  - ImportImageLocalizationPackResultInfo

IServerProjectService interface

```
/// <summary>
/// This interface has operations for server project management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IServerProjectService
{
    ...
    /// <summary>
    /// Creates an image localization package from the image documents of the
    /// projects. The package contains the images for localization and the
    /// instructions for the localizer.
    /// The result of the operation is a downloadable zip file.
    /// The localization pack is bilingual, that it, it contains the images
    /// in the source language, and is translated to a single target
    /// language. Therefore the documents must be of the same target
    /// language and all documents must be images (see IsImage property of
    /// ServerProjectTranslationDocInfo).
    /// </summary>
    /// <param name="serverProjectGuid">The identifier of the project.</param>
    /// <param name="documentGuids">The documents of the project to create
    /// the localization pack from. Must be all images and must be of the
    /// same target language.</param>
    /// <returns>
    /// The FileResultInfo providing information about the result of the
    /// operation. In case of success, it holds guid of the resulting file.
    /// The file can be downloaded using FileManagerService.
    /// and related operations.
    /// </returns>
    [OperationContract]
    [FaultContract(typeof (UnexpectedFault)), FaultContract(typeof (GenericFault))]
    FileResultInfo CreateImageLocalizationPack(Guid serverProjectGuid,
        Guid[] documentGuids);
    /// <summary>
    /// Imports and image localization package and saves the localized images
    /// in the package into the server project.
    /// </summary>
    /// <param name="serverProjectGuid">The identifier of the project.</param>
    /// <param name="uploadedPackageFileId">The identifer of the package file
    /// that was uploaded using IFileManagerService before.</param>
    /// <returns>
    /// The ResultInfo providing information about the result of the operation
    /// including any errors and warnings that may have occured.
    /// </returns>
    [OperationContract]
    [FaultContract(typeof (UnexpectedFault)), FaultContract(typeof (GenericFault))]
    ImportImageLocalizationPackResultInfo ImportImageLocalizationPack(
        Guid serverProjectGuid, Guid uploadedPackageFileId);
    ...
}
```

ImportImageLocalizationPackResultInfo class

```
/// <summary>
/// Encapsulates the result of importing a localization package back
/// into a project.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ImportImageLocalizationPackResultInfo : FileResultInfo
{
    /// <summary>
    /// Contains the names of the images in the imported package that
    /// were not imported because no image document in the project
    /// was found with the same identifier.
    /// </summary>
    [DataMember]
```

```
    public string[] SkippedImages;
}
```

3.8 File management API

3.8.1 Overview

Server project operations involve the transfer of large files (translation documents and bilinguals). Therefore, a unified and reliable transfer method has been introduced, that can be used throughout the API, involving chunked transfer and an option to handle compressed data. The file transfer API provides the following functionality:

- Chunked file upload, optionally zipped. For example, if a translation document is to be imported to a server project, then it has to be uploaded first by using file transfer upload operations.
- Chunked file download, optionally zipped. For example, if a translation document is exported, then the resulting file is created and temporarily stored on the server. Then the file can be downloaded by using file transfer download operations.

3.8.2 Brief description of the interface

The file management related interfaces and entity classes of the memoQ Server WS API are the following:

IFileManagerService (possibly exposed as FileManagerService proxy class): This interface has operations for chunked file upload and download.

3.8.3 Detailed description of the interface

Below is the C# code for the interfaces/classes with their description. The source is annotated by .NET Framework WCF attributes (ServiceContract, OperationContract, DataContract, DataMember), which can be ignored.

IFileManagerService interface

```
/// <summary>
/// This interface has operations for file management (upload and download).
/// </summary>
///<remarks>
/// <para>Certain memoQ Server web service operations (such as
/// IServerProjectService.ImportTranslationDocument for document import)
/// require a file to be uploaded. IFileManagerInterface.BeginChunkedFileUpload
/// and its related operations can be used to upload the file first,
/// and then the file identifier returned by BeginChunkedFileUpload can be used
/// to identify the uploaded file to be imported when calling
/// IServerProjectServicedocument.ImportTranslationDocument.</para>
/// <para>There are other MemoQ Server web service operations (such as
/// IServerProjectServicedocument.ExportTranslationDocument for document export)
/// that store the result on the MemoQ Server in a file. The file identifier
/// returned by these operations can be used to download the file using
/// IFileManagerInterface.BeginChunkedFileDownload and its related
/// operations.</para>
/// <para>
/// Uploaded files and files created by web service operations (such as
```

```
/// IServerProjectServicedocument.ExportTranslationDocument are no longer
/// preserved than 24 hours (can be configured)on the MemoQ Server.
/// Therefore, initiate processing operations (such as document import)
/// immediately after file upload. It is also highly recommended to delete the
/// uploaded file by calling DeleteFile if a file is no longer needed.
/// For example after the processing (such as document import) has finished,
/// and also after having downloaded the resulting file of a server operation
/// (such as document export). This will preserve server resources.
/// </para>
/// </remarks>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IFileManagerService
{
    #region File upload management

    /// <summary>
    /// Starts a new chunked file upload session.
    /// </summary>
    /// <param name="fileName">Name or path of the file. This information
    /// is stored with the uploaded file. It can contain only characters
    /// that are valid on a Windows Operating System (e.g. '?', '\' are not
    /// allowed).</param>
    /// <param name="isZipped">If set to true, the file is assumed to be a
    /// zipped file, therefore it is unzipped on the server after upload.
    /// </param>
    /// <returns>
    /// The session id (guid) of the newly started chunked file upload session.
    /// It should be provided as first parameter for the AddNextFileChunk and
    /// EndChunkedFileUpload. This guid is also the file identifier, that can
    /// be used by other web service operations after the file upload
    /// session has been ended by calling EndChunkedFileUpload.
    /// </returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    Guid BeginChunkedFileUpload(string fileName, bool isZipped);
    /// <summary>
    /// Performs the upload of the next file chunk. The operation should be called
    /// in turns to upload the next chunk of the file.
    /// It is important that the interval between two AddNextFileChunk calls
    /// (and the interval between the call of BeginChunkedFileUpload and the first
    /// call of AddNextFileChunk) is less than a minute or two. If the interval
    /// is larger, then the upload session times out on the server and reserved
    /// resources (such as the file handle) are released, the upload can not
    /// continue.
    /// </summary>
    /// <param name="fileIdAndSessionId">The session id (guid) of the
    /// chunked file upload session created by BeginChunkedFileUpload.</param>
    /// <param name="fileData">The bytes of the file chunk.</param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void AddNextFileChunk(Guid fileIdAndSessionId, byte[] fileData);
    /// <summary>
    /// Ends the chunked file upload session. Call to indicate to the MemoQ Server
    /// that all chunks have been sent by calling AddNextFileChunk.
    /// It is very important to call this method as soon as possible after
    /// uploading the last chunk of data to release server resources (such as
    /// the file handle). If the uploaded file is zipped, then the unzipping is
    /// performed, and the function does not return until the unzipping of the
    /// file has finished.
    /// </summary>
    /// <param name="fileIdAndSessionId">The session id (guid) of the
    /// chunked file upload session created by BeginChunkedFileUpload.</param>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void EndChunkedFileUpload(Guid fileIdAndSessionId);

    #endregion
```

```
#region File download management
/// <summary>
/// Starts a new chunked file download session.
/// </summary>
/// <param name="fileGuid">The file identifier.</param>
/// <param name="zip">If set to true, then the file is zipped before download,
/// and the BeginChunkedFileDownload operation does not return until the
/// file is zipped.</param>
/// <param name="fileName">Name or path of the file (output parameter).
/// It's exact content depends on the operation that has created the file.
/// </param>
/// <param name="fileSize">Size of the file in bytes (output parameter).
/// If the file is to be zipped, the size of the zipped file is returned.
/// </param>
/// <returns>
/// The session id (guid) of the newly started chunked file download session.
/// It should be provided as first parameter for the GetNextFileChunk and
/// EndChunkedFileDownload operations.
/// </returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
Guid BeginChunkedFileDownload(Guid fileGuid, bool zip, out string fileName,
    out int fileSize);
/// <summary>
/// Performs the download of the next file chunk. The operation should be
/// called in turns to download the next chunk of the file.
/// It is important that the interval between two GetNextFileChunk calls
/// (and the interval between the call of BeginChunkedFileDownload and the
/// first call of GetNextFileChunk) is less than a minute or two. If the
/// interval is larger, then the download session times out on the server
/// and reserved resources (such as the file handle) are released, the
/// download can not continue.
/// </summary>
/// <param name="sessionId">The  session id (guid) of the
/// chunked file download session created by BeginChunkedFileDownload.</param>
/// <param name="byteCount">The number of bytes to be returned. The actual
/// number of bytes may be less then this if the amount of remaining file data
/// is less. Therefore alway check the number of bytes of the returned data.
/// </param>
/// <returns>The file data requested. The actual number of bytes may be less
/// then the requested amount if the amount of remaining file data is less.
/// </returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
byte[] GetNextFileChunk(Guid sessionId, int byteCount);
/// <summary>
/// Call to end the chunked file download session after downloading all file
/// bytes by GetNextFileChunk.
/// It is very important to call this method as soon as possible after
/// downloading the last chunk of data to release server resources (such as
/// the file handle).
/// </summary>
/// <param name="sessionId">The session id (guid) of the
/// chunked file download session created by BeginChunkedFileDownload.</param>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
void EndChunkedFileDownload(Guid sessionId);
#endregion

#region Other
/// <summary>
/// Deletes the file on the MemoQ Server. Call to delete uploaded and server
/// generated files as soon as possible if they are no longer needed.
/// </summary>
/// <param name="fileGuid">The identifier of the file to be deleted.</param>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
void DeleteFile(Guid fileGuid);
```

```
    #endregion
}
```

3.8.4 Demo code in C# for using the file manager service:

```
class FileManager
{
    /// <summary>
    /// Uploads a file given by its path.
    /// </summary>
    /// <param name="fmService">The IFileManagerService service proxy.</param>
    /// <param name="filePath">The file path.</param>
    /// <returns>The Guid of the uploaded file.</returns>
    public static Guid UploadFile(IFileManagerService fmService, string filePath)
    {
        FileStream fileStream = null;
        Guid fileGuid = Guid.Empty;
        const int chunkSize = 500000;
        byte[] chunkBytes = new byte[chunkSize];
        byte[] dataToUpload;
        int bytesRead;

        try
        {
            // Open souce file stream
            fileStream = File.OpenRead(filePath);

            // Begin chunked upload
            fileGuid = fmService.BeginChunkedFileUpload(filePath, false);

            // Upload chunks
            while ((bytesRead = fileStream.Read(chunkBytes, 0, chunkSize)) != 0)
            {
                if (bytesRead == chunkSize)
                    dataToUpload = chunkBytes;
                else
                {
                    // If we are at the end, we want to upload only
                    // the bytes read from the stream.
                    dataToUpload = new byte[bytesRead];
                    Array.Copy(chunkBytes, dataToUpload, bytesRead);
                }
                fmService.AddNextFileChunk(fileGuid, dataToUpload);
            }

            return fileGuid;
        }
        finally
        {
            if (fileStream != null)
                fileStream.Close();

            // End chunked upload
            if (fileGuid != Guid.Empty)
                fmService.EndChunkedFileUpload(fileGuid);

        }
    }

    /// <summary>
    /// Downloads a file given by its Guid.
    /// </summary>
    /// <param name="fmService">The IFileManagerService service proxy.</param>
    /// <param name="fileGuid">The file GUID.</param>
    /// <param name="fileName">Name of the file.</param>
    /// <param name="targetdir">The target directory.</param>
```

```
    public static void DownloadFile(IFileManagerService fmService, Guid fileGuid,
        out string fileName, string targetdir)
    {
        FileStream fileStream = null;
        Guid sessionGuid = Guid.Empty;
        const int chunkSize = 500000;
        byte[] chunkBytes;
        int fileSize;
        int fileBytesLeft;

        try
        {
            sessionGuid = fmService.BeginChunkedFileDownload(out fileName,
                out fileSize, fileGuid, false);
            fileStream = new FileStream(Path.Combine(targetdir, fileName),
                FileMode.Create);
            fileBytesLeft = fileSize;

            while (fileBytesLeft > 0)
            {
                chunkBytes = fmService.GetNextFileChunk(sessionGuid, chunkSize);
                fileStream.Write(chunkBytes, 0, chunkBytes.Length);
                fileBytesLeft -= chunkBytes.Length;
            }
        }
        finally
        {
            if (fileStream != null)
                fileStream.Close();

            try
            {
                if (sessionGuid != Guid.Empty)
                    fmService.EndChunkedFileDownload(sessionGuid);

                // We no longer need the file, delete it on the server.
                fmService.DeleteFile(fileGuid);
            }
            catch (Exception e)
            {
                Log.WriteVerbose(e.ToString());
                // We are not interested in these errors. And don't want the
                // original error to be hidden by these exceptions.
            }
        }
    }
}
```

3.9 ELM API

3.9.1 Overview

The ELM API provides the following functionality:

- Manage license assignments if the server is using ELM licensing
    - List license assignments
    - Assign a license to a user
    - Change the expiry of a license assignment
    - Revoke a license assignment

- Manage license permissions
  - List license permissions
  - Adding license permissions to a user or group
  - Change the expiry of a license permission
  - Revoke a license permission
- List project license permissions
- List ELM/CAL license pools

3.9.2 Brief description of the interface

The ELM/CAL related interfaces and entity classes of the memoQ Server WS API are the following:

ELMProduct: Represents an ELM/CAL product. Currently memoQ Translator Pro Edition, memoQ Project Manager Edition and memoQ Translation Light Edition are supported.

ELMAssignmentType: Represents an ELM assignment type. Can be manual, webtrans based, plugin, or permission based.

ELMAssignment: Represents an ELM license to user assignment.

ELMPermission: Represents an ELM/CAL permission.

ELMProjectPermission: Represents an ELM/CAL license pool.

ELMFault: Represents and ELM/CAL related fault.

IELMService (possibly exposed as ELMService proxy class): This interface has operations for ELM/CAL management. Some parts of the interface are only available if the server is using ELM licensing. These functions throw a fault if the server is using CAL licensing.

Please note that memoQ Translator Light licenses are special. These licenses are not assignable directly; memoQ server hands out these licenses for webTrans users and TM/TB plugin connections. This type is available for query only.

3.9.3 Detailed description of the interface

Below is the C# code for the interfaces/classes with their description. The source is annotated by .NET Framework WCF attributes (ServiceContract, OperationContract, DataContract, DataMember, FaultContract), which can be ignored.

IELMService interface

```
/// <summary>
/// This interface has operations for ELM management.
/// </summary>
[ServiceContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public interface IELMService
{
    #region License assignments

    /// <summary>
```

```
/// Returns the list of license assignments.
/// Does not throw if the product does not exist on the system,
/// an empty array is returned in this case. Available only if
/// the server is using ELM licensing, otherwise throws a fault.
/// </summary>
/// <param name="product">The product for which assignments are to be
/// returned.</param>
/// <param name="includeExpiredAndReturned">Indicates whether expired and
/// returned license assignments are also to be included in the list.</param>
/// <returns>An array of license assignments.</returns>
/// <exception cref="ELMFault">Thrown in the following cases:
///  - With error code "ELM+ServerUsingCALLicensing" if the server is using
///    CAL licensing.
/// </exception>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
ELMAssignment[] ListLicenseAssignments(ELMProduct product,
    bool includeExpiredAndReturned);

/// <summary>
/// Assigns a license to the specified user for the specified product.
/// Available only if the server is using ELM licensing,
/// otherwise throws a fault.
/// </summary>
/// <param name="product">The product for which license is to be allocated.
/// </param>
/// <param name="userGuid">The user to whom the license is to be assigned.
/// </param>
/// <param name="expiry">The expiry of the license (UTC time). Has to be a
/// date and time in the future.</param>
/// <returns>The Guid of the new license assignment.</returns>
/// <exception cref="ELMFault">Thrown in the following cases:
///  - With error code "ELM+ServerUsingCALLicensing" if the server is using
///    CAL licensing.
///  - With error code "ELM+ExpiryIsBackInTime" if expiry is back in time.
///  - With error code "ELM+UserDoesNotExist" if user does not exist.
///  - With error code "ELM+ProductDoesNotExist" if product does not exist.
///  - With error code "ELM+NoLicenseIsAvailable" if no free license is
///    available.
///    for the product.
///  - With error code "ELM+ValidLicenseAlreadyExistsForUser" if a valid
///    license already exists for the user for the product.
/// </exception>
[FaultContract(typeof(ELMFault))]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
Guid AssignLicense(ELMProduct product, Guid userGuid, DateTime expiry);

/// <summary>
/// Sets the expiry of the license assignment. Available only if the server
/// is using ELM licensing, otherwise throws a fault.
/// </summary>
/// <param name="assignmentGuid">The Guid of the license assignment.</param>
/// <param name="newExpiry">The UTC date and time of the new expiry.</param>
/// <exception cref="ELMFault">Thrown in the following cases:
///  - With error code "ELM+ServerUsingCALLicensing" if the server is using
///    CAL licensing.
///  - With error code "ELM+ExpiryIsBackInTime" if expiry is back in time.
///  - With error code "ELM+LicenseDoesNotExist" if license assignment does
///    not exist.
///  - With error code "ELM+LicenseIsReturned" if the license is returned.
/// </exception>
[FaultContract(typeof(ELMFault))]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
void SetAssignmentExpiry(Guid assignmentGuid, DateTime newExpiry);

/// <summary>
```

```
/// Revokes the specified license assignment (makes it returned
/// with the current date and time).
/// memoQWeb licenses can not be revoked.
/// Available only if the server is using ELM licensing,
/// otherwise throws a fault.
/// </summary>
/// <param name="assignmentGuid"></param>
/// <exception cref="ELMFault">Thrown in the following cases:
///  - With error code "ELM+ServerUsingCALLicensing" if the server is using
///    CAL licensing.
///  - With error code "ELM+LicenseDoesNotExist" if license assignment does
///    not exist.
///  - With error code "ELM+LicenseIsReturned" if the license is returned.
///  - With error code "ELM+WebTransLicenseCanNotBeRevoked" if the license
///    is a memoQWeb license.
/// </exception>
[FaultContract(typeof(ELMFault))]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
void RevokeAssignment(Guid assignmentGuid);

#endregion

#region License permissions

/// <summary>
/// Returns the list of (manual) license permission assignment.
/// Does not throw if the product does not exist on the system, an empty array
/// is returned in this case.
/// </summary>
/// <param name="product">The product for which permissions are to be
/// returned.</param>
/// <param name="includeExpired">Indicates whether expired and permissions are
/// also to be included in the list.</param>
/// <returns>An array of license permissions.</returns>
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
ELMPermission[] ListLicensePermissions(ELMProduct product,
    bool includeExpired);

/// <summary>
/// Adds a license permission to a user or group for the specified product.
/// </summary>
/// <param name="product">The product to which the license permission
/// is to be added.</param>
/// <param name="guidOfUserOrGroup">The Guid of the user or group to
/// whom the permission is to be added.</param>
/// <param name="expiry">The UTC date and time of the expiry of the
/// permission.</param>
/// <returns>The Id of the newly created permission.</returns>
/// <exception cref="ELMFault">Thrown in the following cases:
///  - With error code "ELM+ExpiryIsBackInTime" if expiry is back in time.
///  - With error code "ELM+UserOrGroupDoesNotExist" if guidOfUserOrGroup
///    does not represent a valid user or group.
///  - With error code "ELM+ProductDoesNotExist" the product does not exist.
///  - With error code "ELM+ValidPermissionAlreadyExistsForUser" if a valid
///    (unexpired) permission already exists for the user/group for the
///   product.
/// </exception>
[FaultContract(typeof(ELMFault))]
[FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
[OperationContract]
int AddPermission(ELMProduct product, Guid guidOfUserOrGroup,
    DateTime expiry);

/// <summary>
/// Sets the expiry of the license permission.
/// </summary>
```

```
    /// <param name="permissionId">The Id of the permission.</param>
    /// <param name="newExpiry">The UTC date and time of the expiry of the
    /// permission to be set.</param>
    /// <exception cref="ELMFault">Thrown in the following cases:
    ///  - With error code "ELM+ExpiryIsBackInTime" if expiry is back in time.
    ///  - With error code "ELM+PermissionDoesNotExist" if the permission does
    ///    not exist.
    /// </exception>
    [FaultContract(typeof(ELMFault))]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void SetPermissionExpiry(int permissionId, DateTime newExpiry);

    /// <summary>
    /// Revokes the specified permission.
    /// </summary>
    /// <param name="permissionId">The Id of the permission.</param>
    [FaultContract(typeof(ELMFault))]
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    void RevokePermission(int permissionId);

    #endregion

    #region Project permissions

    /// <summary>
    /// Returns the list of project permissions for licesnses.
    /// </summary>
    /// <returns>The array of the project permissions.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    ELMProjectPermission[] ListProjectPermissions();

    #endregion

    #region License pools

    /// <summary>
    /// Returns the list of the ELM license pools available on the computer.
    /// </summary>
    /// <returns>The array of the license pools.</returns>
    [FaultContract(typeof(UnexpectedFault)), FaultContract(typeof(GenericFault))]
    [OperationContract]
    ELMPoolInfo[] ListELMPools();

    #endregion
}
```

ELMProduct

```
/// <summary>
/// Represents an ELM product.
/// </summary>
public enum ELMProduct
{
    /// <summary>
    /// memoQ Translation Pro edition.
    /// </summary>
    MemoQTranslatorPro = 0,
    /// <summary>
    /// memoQ Project Manager edition.
    /// </summary>
    MemoQPM = 1,
    /// <summary>
    /// memoQ Translation Light edition.
    /// </summary>
    MemoQTranslatorLight = 2
```

```
}
```

ELMAssignmentType

```
/// <summary>
/// Represents an ELM assignment type.
/// </summary>
public enum ELMAssignmentType
{
    /// <summary>
    /// The license has been assigned manually.
    /// </summary>
    Manual,
    /// <summary>
    /// The license has been assigned automatically when the user opened a
    /// document in web trans.
    /// </summary>
    WebTrans,
    /// <summary>
    /// The license has been assigned automatically to a TM/TB plugin connected
    /// to memoQ server.
    /// </summary>
    Plugin,
    /// <summary>
    /// The license has been taken by a user owning a permission to the product.
    /// </summary>
    Permission
}
```

ELMAssignmentType

```
/// <summary>
/// Represents an ELM license to user assignment.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class ELMAssignment
{
    /// <summary>
    /// Uniquely identifies an ELM assignment.
    /// </summary>
    [DataMember]
    public Guid AssignmentGuid;
    /// <summary>
    /// Identifies the user to whom the license is assigned to.
    /// </summary>
    [DataMember]
    public Guid UserGuid;
    /// <summary>
    /// The user name of the to whom the license is assigned to.
    /// Represents the same user as the UserGuid member.
    /// </summary>
    [DataMember]
    public string UserName;
    /// <summary>
    /// The product the license is assocated with.
    /// </summary>
    [DataMember]
    public ELMProduct Product;
    /// <summary>
    /// The type of the assigmnent.
    /// </summary>
    [DataMember]
    public ELMAssignmentType AssignmentType;
    /// <summary>
    /// The UTC date and time the license expires.
    /// </summary>
    [DataMember]
```

```
    public DateTime Expiry;
    /// <summary>
    /// The UTC date and time the license is has been assigned to the user.
    /// </summary>
    [DataMember]
    public DateTime AssignedAt;
    /// <summary>
    /// The UTC date and time the license has been returned.
    /// For unreturned liceses value 01/01/2500 is used.
    /// </summary>
    [DataMember]
    public DateTime ReturnedAt;
}
```

ELMPermission

```
/// <summary>
/// Represents an ELM permission.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class ELMPermission
{
    /// <summary>
    /// The ID of the permission assignment
    /// </summary>
    [DataMember]
    public int PermissionId;
    /// <summary>
    /// The product the license pool is assocated with.
    /// </summary>
    [DataMember]
    public ELMProduct Product;
    /// <summary>
    /// The Guid of the user or group the permission is assigned to.
    /// </summary>
    [DataMember]
    public Guid UserOrGroupGuid;
    /// <summary>
    /// The name of the user or group the permission is assigned to.
    /// </summary>
    [DataMember]
    public string UserOrGroupName;
    /// <summary>
    /// The UTC date and time the permission assignment expires.
    /// </summary>
    [DataMember]
    public DateTime Expiry;
    /// <summary>
    /// Indicates whether the same user can claim a license from
    /// different clients at the same time.
    /// </summary>
    [DataMember]
    public bool CALMultiUserAllowed;
}
```

ELMProjectPermission

```
/// <summary>
/// Represents an ELM project permission.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class ELMProjectPermission
{
    /// <summary>
    /// The Guid of the user owning the permission.
    /// </summary>
    [DataMember]
    public Guid UserGuid;
```

```
    /// <summary>
    /// The name of the user user owning the permission.
    /// </summary>
    [DataMember]
    public string UserName;
    /// <summary>
    /// The Guid of the project.
    /// </summary>
    [DataMember]
    public Guid ProjectGuid;
    /// <summary>
    /// The name of the project.
    /// </summary>
    [DataMember]
    public string ProjectName;
    /// <summary>
    /// The UTC date and time the permission assignment expires.
    /// </summary>
    [DataMember]
    public DateTime Expiry;
}
```

ELMPoolInfo

```
/// <summary>
/// Represents an ELM license pool.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public class ELMPoolInfo
{
    /// <summary>
    /// The product the license pool is assocated with.
    /// </summary>
    [DataMember] public ELMProduct ELMProduct;

    /// <summary>
    /// The size of the ELM license pool.
    /// </summary>
    [DataMember] public int PoolSize;

    /// <summary>
    /// The number of available licenses in the license pool.
    /// </summary>
    [DataMember]
    public int AvailableCount;
}
```

ELMFault

```
/// <summary>
/// Represents and ELM related fault.
/// </summary>
[DataContract(Namespace = "http://kilgray.com/memoqservices/2007")]
public partial class ELMFault
{
    /// <summary>
    /// The error code of the ELM fault.
    /// </summary>
    [DataMember]
    public string ErrorCode;
}
```

3.9.4 Reported errors

Important note: When using the ELM API make sure to enable advanced error handling mode (see Section 2.5).

The functions above report various types of errors. Besides the generally used UnexpectedFault and GenericFault one ELM specific fault is used: ELMFault. It has an ErrorCode member so that the caller can differentiate between different types of ELM related errors. Using older WS technologies it is difficult to access the ErrorCode: you can check the fault code instead, which has the same value as the ELMFault.ErrorCode. Possible fault types, ErrorCode/fault code values with their fault message are the following:

Fault type	Fault code (same as ErrorCode in case of ELMFault)	Fault message
ELMFault	ELM+LicenseDoesNotExist	The license does not exist.
	ELM+LicenseIsReturned	The license is returned, this operation is invalid for it.
	ELM+ExpiryIsBackInTime	Expiry cannot be set to a value back in time.
	ELM+UserDoesNotExist	The user does not exist.
	ELM+UserOrGroupDoesNotExist	The user/group does not exist.
	ELM+ProductDoesNotExist	The product does not exist.
	ELM+ProductNotAvailableForAssignment	This license type is not available for direct assignment.
	ELM+NoLicenseIsAvailable	There is no available free license.
	ELM+ValidLicenseAlreadyExistsForUser	A valid (unexpired and unreturned) license already exists for the user.
	ELM+WebTransLicenseCanNotBeRevoked	WebTrans or Plugin licenses can not be revoked.
	ELM+ValidPermissionAlreadyExistsForUser	A valid (unexpired) license already exists for the user.
	ELM+PermissionDoesNotExist	The permission does not exist.
	ELM+ServerUsingCALLicensing	The server is using CAL lincencing.
GenericFault	Generic	<The message of the original exception is used as fault message.>
UnexpectedFault	Unexpected	<The message of the original exception is used as fault message.>

Please note that using C# and WCF these faults can be caught based on their type using regular exceptions:

```
try
{
```

```
        // call operation
    }
    // This catches ELMFault errors only
    catch (System.ServiceModel.FaultException<ELMFault> e)
    {

        // Check e.Code.Name or e.ErrorCode for the fault code
    }
    // This catches all other API errors
    catch (System.ServiceModel.FaultException e)
    {

        // Check e.Code.Name or e.ErrorCode for the fault code if you want to
        // differentiate between errors
    }
```

Using older WS technologies, such as ASMX, you need to check the fault code:

```
    try
    {
        // call operation
    }
    // Important: this catches all API errors, not only ELMFault errors!!!
    catch (SoapException e)
    {
        // Check e.Code.Name or e.ErrorCode for the fault code if you want to
        // differentiate between errors
}
```

4. memoQ Server callback API

MemoQ Server provides a web service based callback API to notify and external service about events in memoQ Server. The callback is performed via standard web services technology, therefore enables integration with any platform supporting standard web services, including applications developed in Microsoft .NET or Java.

This section describes the following.

- A brief description of the technology.
- A description of how this callback can be enabled and used.
- The description of the callback service interface.

4.1 Technology

The callback service in memoQ Server is a web service-based client that calls an external web service. The external service must be published as a standard web service, and accessible by memoQ Server. The web service must be an ASMX-based web services or similar service that conform to the WS-I Basic Profile 1.1 using SOAP 1.1 without security.

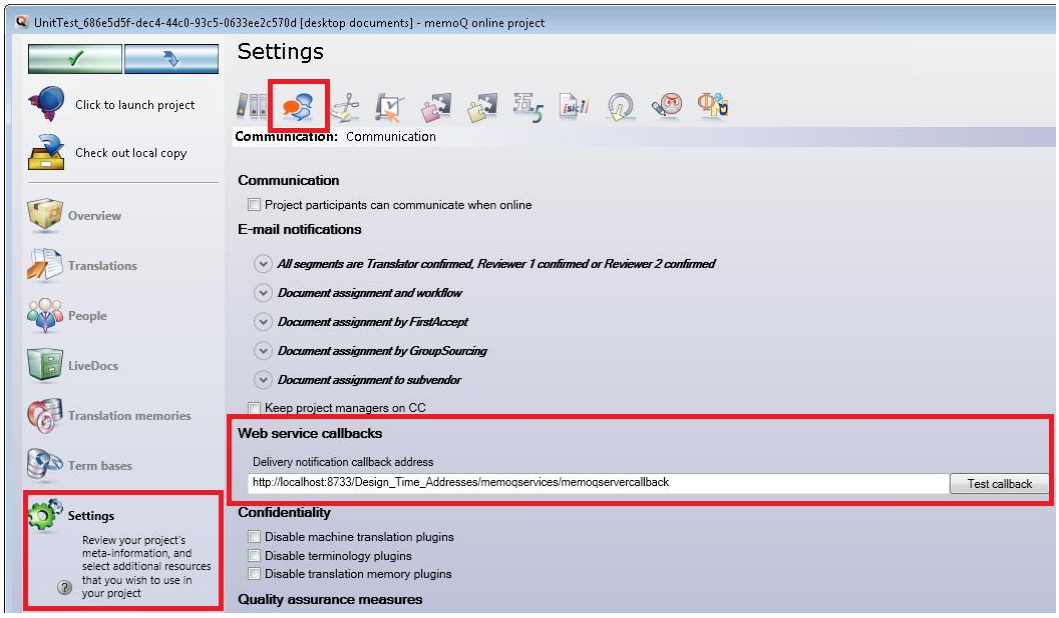
In order to work with this service, memoQ Server defines the service interface (contract) that the external service must implement.

The service is not expected to report any business errors.

4.2 Using the callback service

The callback service is enabled on a per-project basis. Each server project in memoQ Server receives a “callback url.” If this url is provided, the callback service is considered to be enabled for that project. If the url is not provided, the callback service is disabled for that project. MemoQ Server does not check the availability of the service until the service is used.

The remote url of the service can be set when creating new projects and can be updated for existing projects. The url is not validated by memoQ Server. memoQ client provides a test function for server projects that have and url configured. To test the url, open a server project for management in memoQ client, go to Settings / Communication (see Figure below) where the url is displayed, and the test button calls the external service reporting the result.



The location of the test functionality in memoQ client.

4.3 The callback interface

The contract of this callback service is defined in this section. The contract is described in two ways. First, a C# code is listed, that defines the interface and all connected classes. Second, a WSDL is enclosed to this document, which describes the service.

4.3.1 C#

```
/// <summary>
/// The interface which memoQ Server uses to call back an external service.
/// </summary>
[ServiceContract(
    Name = "IMemoQServerCallback",
    Namespace = @"http://kilgray.com/memoq/v1")]
public interface IMemoQServerCallback
{
    /// <summary>
```

```
    /// Notification about delivery of documents.
    /// </summary>
    /// <param name="projectInfo">Describes the project in which the
    /// delivery occurred.</param>
    /// <param name="deliveredItems">The list of delivered items with
    /// details.</param>
    [OperationContract]
    void DocumentDelivery(DeliveryProjectInfo projectInfo,
                          DeliveryItem[] deliveredItems);
    /// <summary>
    /// A method that is used for diagnostics purposes.
    /// </summary>
    /// <param name="data">A random string content sent to the service.</param>
    /// <returns>Any data the service might return.</returns>
    [OperationContract]
    string TestCallback(string data);
}
/// <summary>
/// Describes the project in which documents were delivered.
/// </summary>
[DataContract(
    Name = "DeliveryProjectInfo",
    Namespace = @"http://kilgray.com/memoq/v1")]
public class DeliveryProjectInfo
{
    /// <summary>
    /// The unique identifier of the project.
    /// </summary>
    [DataMember]
    public Guid ProjectGuid;
    /// <summary>
    /// The name of the project.
    /// </summary>
    [DataMember]
    public string ProjectName;
}
/// <summary>
/// Describes a delivered document.
/// </summary>
[DataContract(
    Name = "DeliveryItem",
    Namespace = @"http://kilgray.com/memoq/v1")]
public class DeliveryItem
{
    /// <summary>
    /// The unique identifier of the affected document or slice in memoQ Server.
    /// </summary>
    [DataMember]
    public Guid DocumentGuid;
    /// <summary>
    /// The name of the document or slice.
    /// </summary>
    [DataMember]
    public string DocumentName;
    /// <summary>
    /// The external identifier of the document that was provided
    /// when importing the document, if any. Not interpreted or
    /// changed by memoQ Server, returned in the same format as
    /// received. Null in case of a slice.
    /// </summary>
    [DataMember]
    public string ExternalDocumentId;
    /// <summary>
    /// The three(+two) character language code of the document.
    /// </summary>
    [DataMember]
    public string TargetLanguageCode;
    /// <summary>
```

```
    /// The previous workflow state before the delivery.
    /// </summary>
    [DataMember]
    public WorkflowStatus PreviousWorkflowStatus;
    /// <summary>
    /// The new workflow state after the delivery.
    /// </summary>
    [DataMember]
    public WorkflowStatus NewWorkflowStatus;
    /// <summary>
    /// The unique identifiers of the user(s) (in memoQ Server) who were
    /// the previous assignees of the document before the delivery
    /// (if applicable).
    /// </summary>
    [DataMember]
    public Guid[] PreviousAssignedUserId;
    /// <summary>
    /// The usernames of the user(s) (in memoQ Server) who were
    /// the previous assignees of the document before the delivery
    /// (if applicable).
    /// </summary>
    [DataMember]
    public string[] PreviousAssignedUserName;
    /// <summary>
    /// The unique identifiers of the user(s) (in memoQ Server) who are
    /// the new assignees of the document after the delivery
    /// (if applicable).
    /// </summary>
    [DataMember]
    public Guid[] NewAssignedUserId;
    /// <summary>
    /// The usernames of the user(s) (in memoQ Server) who are
    /// the new assignees of the document after the delivery
    /// (if applicable).
    /// </summary>
    [DataMember]
    public string[] NewAssignedUserName;
}
/// <summary>
/// Workflow status of the document.
/// </summary>
[DataContract(
    Name = "WorkflowStatus",
    Namespace = @"http://kilgray.com/memoq/v1")]
public enum WorkflowStatus
{
    /// <summary>
    /// Initial workflow state, or workflow state when it is not
    /// defined.
    /// </summary>
    [EnumMember]
    Unknown = 0,
    /// <summary>
    /// The Translator is the next actor of the document but has
    /// not started working on it yet.
    /// </summary>
    [EnumMember]
    TranslationNotStarted = 1000,
    /// <summary>
    /// The Translator has started working on the document.
    /// </summary>
    [EnumMember]
    TranslationInProgress = 2000,
    /// <summary>
    /// The Translator has finished with the document, and it
    /// is now in the hands of the Review1 who has not started
    /// working with the document yet.
    /// </summary>
```

```
    [EnumMember]
    Review1NotStarted = 3000,
    /// <summary>
    /// The Reviewer1 has started working on the document.
    /// </summary>
    [EnumMember]
    Review1InProgress = 4000,
    /// <summary>
    /// The Reviewer1 has finished with the document, and it
    /// is now in the hands of the Review2 who has not started
    /// working with the document yet.
    /// </summary>
    [EnumMember]
    Review2NotStarted = 5000,
    /// <summary>
    /// The Reviewer2 has started working on the document.
    /// </summary>
    [EnumMember]
    Review2InProgress = 6000,
    /// <summary>
    /// The document has been finished by everyone.
    /// </summary>
    [EnumMember]
    Completed = 7000

}
```

4.3.2 WSDL

The WSDL of the callback service is enclosed to this documentation among the WSDL descriptors with the name “callbackservice.”

5. Sample code

In this chapter we provide C# source code demonstrating the usage of the web service API Using the API from Java language is very similar. The following aspects of the API are covered by the sample code:

- Creating a server project with live documents and desktop documents too
- Creating and publishing a translation memory
- Creating and publishing a term base
- Listing all memoQ Server users
- Assigning users to a server project
- Upload documents to the memoQ Server
- Importing uploaded documents to a server project
- Assigning imported documents to server project users
- Pre-translating all project documents
- Exporting documents to their primary format
- Downloading exported documents
- Changing the workflow status of documents

The entry point of the sample code is the `DemoTest` operation of the `ServerProjectServiceTest` class.

```
class ServerProjectServiceTest
{
    public string DemoTest()
    {
        // Get server project service object.
        IServerProjectService spService = getServerProjectService();

        // Create server project, the guid of the newly created project
        // is returned.
        // The project will have two target languages: english and german.
        string spName;
        Guid spGuid = createSampleProject(out spName,
            new string[] {"eng", "ger"});

        //------------ Set TM assignments for both target lang -------------

        // Create TMs, the guid of the newly created TM is retuned
        Guid tmGuid1 = createAndPublishSampleTM("hun", "eng");
        Guid tmGuid2 = createAndPublishSampleTM("hun", "eng");
        Guid tmGuid3 = createAndPublishSampleTM("hun", "ger");

        // Prepare TM assignments for target lang "eng"
        ServerProjectTMAssignmentsForTargetLang tmAssignmentsEng
            = new ServerProjectTMAssignmentsForTargetLang();
        tmAssignmentsEng.TargetLangCode = "eng";
        tmAssignmentsEng.TMGuids = new Guid[] {tmGuid1, tmGuid2};
        tmAssignmentsEng.PrimaryTMGuid = tmGuid1;

        // Prepare TM assignments for target lang "ger"
        ServerProjectTMAssignmentsForTargetLang tmAssignmentsGer
            = new ServerProjectTMAssignmentsForTargetLang();
        tmAssignmentsGer.TargetLangCode = "ger";
        tmAssignmentsGer.TMGuids = new Guid[] { tmGuid3 };
        tmAssignmentsGer.PrimaryTMGuid = tmGuid3;

        ServerProjectTMAssignmentsForTargetLang[] tmAssignments =
          new ServerProjectTMAssignmentsForTargetLang[] { tmAssignmentsEng,
              tmAssignmentsGer };

        // Apply TM assignmenst
        spService.SetProjectTMs2(spGuid, tmAssignments);

        // Create a TB, the guid of the newly created TB is retuned
        Guid tbGuid = createAndPublishSampleTB();
        // Assign the TB to the project
        spService.SetProjectTBs(spGuid, new Guid[] { tbGuid }, tbGuid);

        // Assign the users to the project

        // First, select two random users to be assigned.
        MemoQServicesClient.SecurityService.UserInfo[] allUsers =
            getSecurityService().ListUsers();
        List<MemoQServicesClient.SecurityService.UserInfo> allUsersWithoutAdmin =
            new List<MemoQServicesClient.SecurityService.UserInfo>();
        foreach (MemoQServicesClient.SecurityService.UserInfo u in allUsers)
            if (u.UserGuid != TestUsers.TestUserAdminGuid)
                allUsersWithoutAdmin.Add(u);
        if (allUsersWithoutAdmin.Count <= 2)
            throw new Exception(@"To run this test you have to have
         at least three users on MemoQ Server");
        int indexOfUser = new Random().Next(allUsersWithoutAdmin.Count - 1);
        Guid userGuidOfRandomUser1 = allUsersWithoutAdmin[indexOfUser].UserGuid;
        indexOfUser = (indexOfUser + 1) % allUsersWithoutAdmin.Count;
        Guid userGuidOfRandomUser2 = allUsersWithoutAdmin[indexOfUser].UserGuid;

        // First, select the built in admin user to be assigned
        // with project role ProjectManager (but not Terminologist)
```

```
ServerProjectUserInfo spui1 = new ServerProjectUserInfo();
spui1.UserGuid = TestUsers.TestUserAdminGuid;
spui1.ProjectRoles = new ServerProjectRoles();
spui1.ProjectRoles.ProjectManager = true;
spui1.ProjectRoles.Terminologist = false;

ServerProjectUserInfo spui2 = new ServerProjectUserInfo();
spui2.UserGuid = userGuidOfRandomUser1;
spui2.ProjectRoles = new ServerProjectRoles();
spui2.ProjectRoles.ProjectManager = false;
spui2.ProjectRoles.Terminologist = false;

ServerProjectUserInfo spui3 = new ServerProjectUserInfo();
spui3.UserGuid = userGuidOfRandomUser2;
spui3.ProjectRoles = new ServerProjectRoles();
spui3.ProjectRoles.ProjectManager = true;
spui3.ProjectRoles.Terminologist = true;

// Apply project-user assignments
ServerProjectUserInfo[] sampleUserInfos = new ServerProjectUserInfo[]
{
  spui1, spui2, spui3
};
spService.SetProjectUsers(spGuid, sampleUserInfos);

// Upload documents

// Make sure this is a valid path
string pathOfTxt01 = "c:\\_MemoQTest_\\testfiles\\#A#.txt";
Guid fileGuidTxt01 = uploadSampleFile(pathOfTxt01);
// Make sure this is a valid path
string pathOfRtf01 = "c:\\_MemoQTest_\\testfiles\\#G_rtf#.rtf";
//string pathOfFileB = "c:\\_MemoQTest_\\testfiles\\#H_rtf_biling#.rtf";
Guid fileGuidRtf01 = uploadSampleFile(pathOfRtf01);

#region Assign some light resources to the project

// These steps are optional, as defaults for resource types that
// are required for the project are always assigned at project
// creation.
// To find out what are the rules for the assignment of different
// resource types see code comments on class
// ServerProjectResourceAssignmentForResourceType

// Import a segmentation rule resource and assign it to the project
IResourceService resourceService = getResourceService();
// This is a "hun" segrule (has to be as the source lang of the project
// is "hun".
string pathOfSegruleRes =
     "c:\\_MemoQTest_\\testfiles\\Res_SegRuleHun.mqres";
Guid fileGuidSegruleRes = uploadSampleFile(pathOfSegruleRes);
// SegRule type resources use LightResourceInfoWithLang (derived from
// LightResourceInfo. But as the language code of a segrule resource
// can not be changed during import, LightResourceInfo is also accepted.
ResourceService.LightResourceInfo segRuleResInfo =
    new ResourceService.LightResourceInfo();
segRuleResInfo.Name = "SegRule_Name_" + Guid.NewGuid().ToString();
segRuleResInfo.Description = "###unittest###";
Guid segRuleGuid =
    resourceService.ImportNewAndPublish(
    ResourceService.ResourceType.SegRules, fileGuidSegruleRes,
    segRuleResInfo );

ServerProjectResourceAssignment aSr1 =
    new ServerProjectResourceAssignment();
aSr1.ResourceGuid = segRuleGuid;
// Not required to set these, as null and false are the defaults
// aSr1.ObjectId = null;
```

```
// aSr1.Primary = false;

ServerProjectResourceAssignmentForResourceType segRuleAssignments =
     new ServerProjectResourceAssignmentForResourceType();
segRuleAssignments.ResourceType = ResourceType.SegRules;
segRuleAssignments.ServerProjectResourceAssignment =
    new ServerProjectResourceAssignment[] { aSr1 };

// Do not apply the SegRule resource yet. Instead, prepare a
// TMSetting resource to override project defaults and apply
// the SegRule and TMSetting in one step (this is slightly more
// effective),

string pathOfTMSettingsRes =
    "c:\\_MemoQTest_\\testfiles\\Res_TMSettings01.mqres";
Guid fileGuidTMSettingsRes = uploadSampleFile(pathOfTMSettingsRes);
// TMSettings resources use LightResourceInfo objects directly.
ResourceService.LightResourceInfo tmsResInfo =
    new ResourceService.LightResourceInfo();
tmsResInfo.Name = "TMSettings_Name_" + Guid.NewGuid().ToString();
tmsResInfo.Description = "###unittest###";
Guid tmsGuid =
    resourceService.ImportNewAndPublish(
    ResourceService.ResourceType.TMSettings, fileGuidTMSettingsRes,
    tmsResInfo);
// Assign deployed default as project default (this was assigned at
// project creation, but will be deleted at assignment, so we have
// to reassign it.
ServerProjectResourceAssignment aTmsProj =
    new ServerProjectResourceAssignment();
aTmsProj.ResourceGuid = Guid.Empty; // Assignment of deployed default.
aTmsProj.ObjectId = null; // Assignment for project default.
// Override TM settings for TM1
ServerProjectResourceAssignment aTmsTM1 =
    new ServerProjectResourceAssignment();
aTmsTM1.ResourceGuid = tmsGuid; // Assignment of deployed default.
aTmsTM1.ObjectId = tmGuid1.ToString(); // Assignment for TM1
ServerProjectResourceAssignmentForResourceType tmsAssignments =
     new ServerProjectResourceAssignmentForResourceType();
tmsAssignments.ResourceType = ResourceType.TMSettings;
tmsAssignments.ServerProjectResourceAssignment =
    new ServerProjectResourceAssignment[] { aTmsProj, aTmsTM1 };
// Apply the assignments for SegRule and TMSettings in one step
// (also could be done in two steps)
spService.SetProjectResourceAssignments(
    spGuid,
    new ServerProjectResourceAssignmentForResourceType[]
    { segRuleAssignments, tmsAssignments }
  );

#endregion

//  --- Import document(s) now ---

TranslationDocImportResultInfo trResultInfo;
// Import first file based on its extension with default
// settings into all project target langs.
trResultInfo = spService.ImportTranslationDocument(spGuid, fileGuidTxt01,
    null, null);
// Delete file on server, we no longer need it
getFileManagerService().DeleteFile(fileGuidTxt01);

Guid docAGuidEng;
Guid docAGuidGer;

if (trResultInfo.ResultStatus == ResultStatus.Error)
{
    // Handle error, we have the error message in
```

```
        // trResultInfo.MainMessageField
        // and trResultInfo.MainMessageField

        throw new Exception("Document import failed: " +
            trResultInfo.MainMessage);
    }
    else
    {
        // If all went OK...

        // We received the guids of the imported doc for all
        // requested target languages.
        docAGuidEng = trResultInfo.DocumentGuids[0];
        docAGuidGer = trResultInfo.DocumentGuids[1];

        // Just to be fully sure we could examine that none of the
        // received Guids is "00000000-0000-0000-0000-000000000000"
        // We skip this step for simplicity.

        // Assigne doc to users with requested document assignment
        // role (Translator/Reviewer1/reviewer2) and deadlines.

        // First prepare assigmnent of docA target lang "eng"
        ServerProjectTranslationDocumentUserAssignments spdaDocAEng =
            new ServerProjectTranslationDocumentUserAssignments();
        spdaDocAEng.DocumentGuid = docAGuidEng;
        spdaDocAEng.UserRoleAssignments =
            new TranslationDocumentUserRoleAssignment[3];

        // Assign admin to the as Translator document assignment role
        spdaDocAEng.UserRoleAssignments[0] =
            new TranslationDocumentUserRoleAssignment();
        spdaDocAEng.UserRoleAssignments[0].UserGuid =
            TestUsers.TestUserAdminGuid;
        spdaDocAEng.UserRoleAssignments[0].DocumentAssignmentRole = 0;
        spdaDocAEng.UserRoleAssignments[0].DeadLine = new DateTime(2010, 5, 1);

        // Assign random user to the as Reviewer1 document assignment role
        // with no deadline
        spdaDocAEng.UserRoleAssignments[1] =
            new TranslationDocumentUserRoleAssignment();
        spdaDocAEng.UserRoleAssignments[1].UserGuid = userGuidOfRandomUser1;
        spdaDocAEng.UserRoleAssignments[1].DocumentAssignmentRole = 1;
        spdaDocAEng.UserRoleAssignments[1].DeadLine = new DateTime(2500, 1, 1);

        // Assign another random user to the as Reviewer2 document
        // assignment role
        // with no deadline
        spdaDocAEng.UserRoleAssignments[2] =
            new TranslationDocumentUserRoleAssignment();
        spdaDocAEng.UserRoleAssignments[2].UserGuid = userGuidOfRandomUser2;
        spdaDocAEng.UserRoleAssignments[2].DocumentAssignmentRole = 2;
        spdaDocAEng.UserRoleAssignments[2].DeadLine = new DateTime(2500, 1, 1);

        // Next prepare assigmnent of docA target lang "ger"

        ServerProjectTranslationDocumentUserAssignments spdaDocAGer =
            new ServerProjectTranslationDocumentUserAssignments();
        spdaDocAGer.DocumentGuid = docAGuidGer;
        spdaDocAGer.UserRoleAssignments =
            new TranslationDocumentUserRoleAssignment[1];

        // Assign admin to the as Translator document assignment role
        spdaDocAGer.UserRoleAssignments[0] =
            new TranslationDocumentUserRoleAssignment();
        spdaDocAGer.UserRoleAssignments[0].UserGuid =
            TestUsers.TestUserAdminGuid;
        spdaDocAGer.UserRoleAssignments[0].DocumentAssignmentRole = 0;
```

```
        spdaDocAGer.UserRoleAssignments[0].DeadLine = new DateTime(2010, 5, 1);

        // We assign no user to the Reviewer1 and Reviewer2 roles for
        // target lang "ger"

        // Now save the assignment we have prepared
        ServerProjectTranslationDocumentUserAssignments[]
         documentUserAssignments =
            new ServerProjectTranslationDocumentUserAssignments[]
        {
            spdaDocAEng, spdaDocAGer
        };

        spService.SetProjectTranslationDocumentUserAssignments(spGuid,
            documentUserAssignments);

    }

    // Import next file based on its extension with default
    // settings into target lang english only.
    trResultInfo = spService.ImportTranslationDocument(spGuid, fileGuidRtf01,
        new string[] {"eng"}, null );
    if (trResultInfo.ResultStatus == ResultStatus.Error)
        throw new Exception("Document import failed: "
            + trResultInfo.MainMessage);
    Guid docRtf01GuidEng = trResultInfo.DocumentGuids[0];
    // Delete file on server, we no longer need it
    getFileManagerService().DeleteFile(fileGuidRtf01);

    // We could prepare user-doc assigmnent here for all imported
    // docs the same way as for the first one, and save all of
    // them in one step (calling SetProjectTranslationDocumentUserAssignments
    // only once).

    // Pre-translate all docs for all target languages
    // Please note that our TMs are empty, so no mathes are found.
    PretranslateOptions pretranslateOptions = new PretranslateOptions();
    pretranslateOptions.GoodMatchRate = 80;
    pretranslateOptions.OnlyUnambiguousMatches = false;
    pretranslateOptions.PretranslateBehavior =
        PretranslateOverwriteBehavior.KeepExistingSegments;
    pretranslateOptions.PretranslateLookupBehavior =
        PretranslateLookupBehavior.AnyMatch;
    pretranslateOptions.UseMT = true;
    spService.PretranslateProject(spGuid, null, pretranslateOptions);

    // Statistics
    StatisticsOptions statOptions = new StatisticsOptions();
    statOptions.Algorithm = StatisticsAlgorithm.Trados;
    statOptions.Analysis_Homogenity = true;
    statOptions.Analysis_ProjectTMs = true;
    statOptions.Analyzis_DetailsByTM = true;
    statOptions.IncludeLockedRows = true;
    statOptions.ShowCounts = true;
    statOptions.ShowCounts_IncludeTargetCount = true;
    statOptions.ShowCounts_StatusReport = true;
    statOptions.ShowResultsPerFile = true;
    StatisticsResultInfo statResult
        = spService.GetStatisticsOnProject(spGuid, new string[] { "eng", "ger"
          },
        statOptions, StatisticsResultFormat.CSV_Trados);
    if (statResult.ResultStatus == ResultStatus.Error)
        throw new Exception("Statistics failed: " + trResultInfo.MainMessage);

    // In statResult.ResultsForTargetLangs[0].ResultData we have the result
    // for the first target language requested.
    // In statResult.ResultsForTargetLangs[1].ResultData we have the result
    // for the second target language requested.
```

```
// Export document A
string docName;
TranslationDocExportResultInfo resultInfo;
string exportedFileName;
string exportDir = "c:\\_MemoQTest_\\out";

// Export and download the first imported document, english version.
resultInfo = spService.ExportTranslationDocument(spGuid,
    docAGuidEng);
if (resultInfo.ResultStatus == ResultStatus.Error)
    throw new Exception("There was an error during the export," +
    " with message: " + resultInfo.MainMessage);
// Download the result of the export of document A
downloadSampleDoc(resultInfo.FileGuid, out exportedFileName, exportDir);
// Delete file on server, we no longer need it
getFileManagerService().DeleteFile(resultInfo.FileGuid);

// Export and download the first imported document, german version.
resultInfo = spService.ExportTranslationDocument(spGuid,
    docAGuidGer);
if (resultInfo.ResultStatus == ResultStatus.Error)
    throw new Exception("There was an error during the export," +
    " with message: " + resultInfo.MainMessage);
// Download the result of the export of document A
downloadSampleDoc(resultInfo.FileGuid, out exportedFileName, exportDir);
// Delete file on server, we no longer need it
getFileManagerService().DeleteFile(resultInfo.FileGuid);


// -------------- Create DesktopDocs project, set and query document
//                workflow status
ServerProjectDesktopDocsCreateInfo spDDCreateInfo =
        new ServerProjectDesktopDocsCreateInfo()
{
    Name = "DemoSP_" + Guid.NewGuid().ToString(),
    SourceLanguageCode = "hun",
    TargetLanguageCodes = new string[] { "eng" },
    Subject = "###unittest###",
    Deadline = DateTime.Now + TimeSpan.FromDays( 1 ),
    CreatorUser = TestUsers.TestUserAdminGuid
};
spGuid = spService.CreateProject2( spDDCreateInfo );

// Upload a file which is to be imported next
fileGuidTxt01 = uploadSampleFile( pathOfTxt01 );
// Import first file based on its extension with default
// settings into all project target langs.
trResultInfo = spService.ImportTranslationDocument( spGuid, fileGuidTxt01,
    null, null );
// Delete file on server, we no longer need it
getFileManagerService().DeleteFile( fileGuidTxt01 );
if( trResultInfo.ResultStatus == ResultStatus.Success )
{
    // Check the workflow status of the document (we know that there is
    // one document in the project.
    WorkflowStatus workflowStat =
        spService.ListProjectTranslationDocuments( spGuid )[0]
        .WorkflowStatus;
    // Change the workflow status
    spService.SetDocumentWorkflowStatus( spGuid,
        new ServerProjectTranslationDocumentWorkflowStatusChange[]
        {
            // One instance for every document to change
            new ServerProjectTranslationDocumentWorkflowStatusChange()
            {
                DocumentGuid = trResultInfo.DocumentGuids[0],
                WorkflowStatus = WorkflowStatus.Review1InProgress
```

```
                }
            } );
        }

    return spName;
}

struct TestUsers
{
    // Built in administrator
    public static readonly Guid TestUserAdminGuid =
        new Guid("00000000-0000-0000-0001-000000000001");
};

#region Getting service objects

/// <summary>
/// Gets a ITMService service object. This is platform and technology
/// specific.
/// </summary>
/// <returns></returns>
ITMService getTMService()
{
    ChannelFactory<ITMService> channelFactory =
        new ChannelFactory<ITMService>("TMService");
    return channelFactory.CreateChannel();
}

/// <summary>
/// Gets a ITBService service object. This is platform and technology
/// specific.
/// </summary>
/// <returns></returns>
ITBService getTBService()
{
    ChannelFactory<ITBService> channelFactory =
        new ChannelFactory<ITBService>("TBService");
    return channelFactory.CreateChannel();
}

/// <summary>
/// Gets a ISecurityService service object. This is platform and technology
/// specific.
/// </summary>
/// <returns></returns>
ISecurityService getSecurityService()
{
    ChannelFactory<ISecurityService> channelFactory =
        new ChannelFactory<ISecurityService>("SecurityService");
    return channelFactory.CreateChannel();
}

/// <summary>
/// Gets a IServerProjectService service object. This is platform and
/// technology specific.
/// </summary>
/// <returns></returns>
IServerProjectService getServerProjectService()
{
    ChannelFactory<IServerProjectService> channelFactory =
        new ChannelFactory<IServerProjectService>("ServerProjectService");
    return channelFactory.CreateChannel();
}

/// <summary>
/// Gets an IResourceService service object. This is platform and
/// technology specific.
/// </summary>
```

```
/// <returns></returns>
IResourceService getResourceService()
{
    ChannelFactory<IResourceService> channelFactory =
        new ChannelFactory<IResourceService>("ResourceService");
    return channelFactory.CreateChannel();
}

/// <summary>
/// Gets a IFileManagerService service object. This is platform and
/// technology specific.
/// </summary>
/// <returns></returns>
IFileManagerService getFileManagerService()
{
    ChannelFactory<IFileManagerService> channelFactory =
        new ChannelFactory<IFileManagerService>("FileManagerService");
    return channelFactory.CreateChannel();
}

#endregion

#region Helper methods

/// <summary>
/// Creates and publishes a sample translation memory.
/// </summary>
/// <returns>The guid of the newly created TM.</returns>
Guid createAndPublishSampleTM(string sourceLangCode,
    string targetLangCode)
{
    MemoQServicesClient.TMService.TMInfo tmInfo;
    tmInfo = new MemoQServicesClient.TMService.TMInfo(
        Guid.NewGuid(), // This guid is ignored.
        "TestTM1_Name_" + Guid.NewGuid().ToString(),
        "##UnitTest##",
        false, false, false, false,
        sourceLangCode, targetLangCode);
    ITMService tmService = getTMService();
    // CreateAndPublish returns the guid of the new TM
    return tmService.CreateAndPublish(tmInfo);
}

/// <summary>
/// Creates and publishes a sample translation memory.
/// </summary>
/// <returns>The guid of the newly created TB.</returns>
Guid createAndPublishSampleTB()
{
    MemoQServicesClient.TBService.TBInfo tbInfo;
    tbInfo = new MemoQServicesClient.TBService.TBInfo(
        Guid.NewGuid(), // This guid is ignored.
        "TestTB1_Name" + Guid.NewGuid().ToString(),
        "##UnitTest##", false,
        new string[] { "hun", "eng", "ger"}
        );
    ITBService tbService = getTBService();
    // CreateAndPublish returns the guid of the new TB
    return tbService.CreateAndPublish(tbInfo);
}

/// <summary>
/// Creates a sample server project.
/// </summary>
/// <param name="name">The name of the project.</param>
/// <param name="targetLangCodes">The target language codes of the
/// project.</param>
/// <returns>The guid of the newly created project.</returns>
```

```
Guid createSampleProject(out string name, string[] targetLangCodes)
{
    var spCreateInfo = new ServerProjectDesktopDocsCreateInfo()
    {
        // Resulted in PathTooLongException, cut down a little bit
        Name = "DemoSP_" + Guid.NewGuid().ToString(),
        //Name = "SP" + DateTime.Now.ToString("MMdd.HHmmss.fff"),
        SourceLanguageCode = "hun",
        TargetLanguageCodes = targetLangCodes,
        Subject = "###unittest###",
        Deadline = DateTime.Now + TimeSpan.FromDays(1),
        CreatorUser = TestUsers.TestUserAdminGuid
    };

    IServerProjectService serverProjectService = getServerProjectService();
    name = spCreateInfo.Name;
    return serverProjectService.CreateProject2(spCreateInfo);
}

/// <summary>
/// Uploads the sample file.
/// </summary>
/// <param name="filePath">The file path.</param>
/// <returns>The guid of the uploaded file.</returns>
Guid uploadSampleFile(string filePath)
{
    FileStream fileStream = null;
    Guid fileGuid = Guid.Empty;
    int chunkSize = 1000000;
    byte[] chunkBytes = new byte[chunkSize];
    byte[] dataToUpload;
    int bytesRead;

    IFileManagerService fmService = getFileManagerService();

    try
    {
        // Open souce file stream
        fileStream = File.OpenRead(filePath);

        // Begin chunked upload
        fileGuid = fmService.BeginChunkedFileUpload(filePath, false);

        // Upload chunks
        while ((bytesRead = fileStream.Read(chunkBytes, 0, chunkSize)) != 0)
        {
            if (bytesRead == chunkSize)
                dataToUpload = chunkBytes;
            else
            {
                // If we are at the end, we want to upload only
                // the bytes read from the stream.
                dataToUpload = new byte[bytesRead];
                Array.Copy(chunkBytes, dataToUpload, bytesRead);
            }

            fmService.AddNextFileChunk(fileGuid, dataToUpload);
        }

        return fileGuid;
    }
    finally
    {
         if (fileStream != null)
              fileStream.Close();

        // End chunked upload
        if (fileGuid != Guid.Empty)
```

```
                fmService.EndChunkedFileUpload(fileGuid);
            }
        }


        /// <summary>
        /// Downloads the sample document.
        /// </summary>
        /// <param name="fileGuid">The file GUID.</param>
        /// <param name="fileName">Name of the file.</param>
        /// <param name="targetGir">The target gir.</param>
        void downloadSampleDoc(Guid fileGuid, out string fileName, string targetGir)
        {
            FileStream fileStream = null;
            Guid sessionGuid = Guid.Empty;
            int chunkSize = 1000000;
            byte[] chunkBytes = new byte[chunkSize];
            int fileSize;
            int fileBytesLeft;

            IFileManagerService fmService = getFileManagerService();

            try
            {
                sessionGuid = fmService.BeginChunkedFileDownload(out fileName, out
fileSize,
                    fileGuid, false);
                fileStream = new FileStream(Path.Combine(targetGir, fileName),
                    FileMode.Create);
                fileBytesLeft = fileSize;

                while (fileBytesLeft > 0)
                {
                    chunkBytes = fmService.GetNextFileChunk(sessionGuid, chunkSize);
                    fileStream.Write(chunkBytes, 0, chunkBytes.Length);
                    fileBytesLeft -= chunkBytes.Length;
                }
            }
            finally
            {
                if (fileStream != null)
                      fileStream.Close();

                if (sessionGuid != Guid.Empty)
                    fmService.EndChunkedFileDownload(sessionGuid);
            }
        }

        private void exportTranslationDocument(IServerProjectService spService,
            Guid spGuid, Guid docGuid, string exportDir)
        {
            TranslationDocExportResultInfo resultInfo;
            string exportedFileName;
            resultInfo = spService.ExportTranslationDocument(spGuid,
                                                             docGuid);
            if (resultInfo.ResultStatus == ResultStatus.Error)
                throw new Exception("There was an error during the export," +
                                    " with message: " + resultInfo.MainMessage);
            // Download the result of the export of document A
            downloadSampleDoc(resultInfo.FileGuid, out exportedFileName, exportDir);
            // Delete file on server, we no longer need it
            getFileManagerService().DeleteFile(resultInfo.FileGuid);
        }

        #endregion
}
```

6. Migration from previous versions

6.1 Summary of changes when migrating from memoQ server 3.x versions to 4.2

The Security API (ISecurityService) and the File Management API (IFileManagerService) remained unchanged.

6.1.1 Major Conceptual changes

Project have become multilangual: more than one target languages are supported. Documents are idetified by Guid.

Documents within a project are no longer identified by their name: when a file is imported into a project, it receives a Guid (called „document Guid") that uniquely identifies the document within the project. When a document is imported into multiple target languages, it receives a separate Guid for each target language. Please note that two documents within two separate projects may have the same document Guid (e.g. when a document is exported as a bilingual document and then imported into two separate projects: the imported documents will have the same Guid as the exported one). The document Guid together with the server project Guid is unique within the system.

6.1.2 Brief list of changes

Translation Memory and TermBase API related

The Translation Memory API (ITMService), the Term Base API (ITBService)

- The ResourceInfo class has been split into two classes:
  - ResourceInfo class: The same as the old ResourceInfo class excluding metadata (Domain, Subject, etc.)
  - HeavyResourceInfo class: It is derived from the ResourceInfo class, and adds metadata fields (Domain, Subject, etc.). This is the base class of the TMInfo and TBInfo classes.
- The base class of TMInfo and TBInfo is HeavyResourceInfo instead of ResourceInfo.

Server project API related

- CreateProject
  - ServerProjectCreateInfo parameter expects a list of target language codes instead of a single language code.
- ListProjects
  - Changes related to the ServerProjectListFilter parameter
    - Field IsArchived has been eliminated and TimeClosed has been introduced instead.
    - Field TargetLanguageCode has been reinterpreted
  - Changes related to the returned ServerProjectInfo objects

        - Field string TargetLanguageCode is changed to string[] TargetLanguageCodes
        - TimeClosed field introduced
- SetArchivedStatusOfProject has been renamed to SetClosedStatusOfProject
    - Projects no longer can be archived. They can be closed intead, which basically does the same thing as in previous versions, but the new name better reflects what is done by the system. The only difference that the closed status is no longer a flag: memoQ server stores the date and time when the project has been closed. This enables users to filter on the time, when the project has been closed. See ServerProjectListFilter.TimeClosed for the details.
- SetProjectTMs
    - The third parameter is an array of Guids instead of a single Guid as the primary TM has to be specified for each project target language for which at least one TM is assigned.
- SetProjectTBs
    - The third parameter is an array of Guids instead of a single GUID as the primary TB has to be specified for each project tartget language for which at least one TB is assigned.
- SetProjectUsers and ListProjectUsers
    - The ProjectRole member of the ServerProjectUserInfo and ServerProjectUserInfo classes involved has been renamed to ProjectRoles, and its type has been changed to class ServerProjectRoles from enum ServerProjectRole. One user can have multiple project roles now.
- ImportTranslationDocument and ImportTranslationDocuments
    - The list of target languages the document is to be imported to has to be specified with the new targetLangCodes parameters
    - The type of the returned object(s) is TranslationDocImportResultInfo which has one extra field compared to the formerly used ResultInfo: an array of document Guids (one for each target language). Each document Guid uniquely identifies a document within a project.
    - The format of the importSettingsXML parameter of ImportTranslationDocument and ImportTranslationDocuments has been slightly changed.
- ImportBilingualTranslationDocument
- UpdateTranslationDocumentFromBilingual
- SetProjectTranslationDocumentUserAssignments
    - ServerProjectTranslationDocumentUserAssignment parameter related changes
        - The type ServerProjectTranslationDocumentUserAssignment has been renamed to ServerProjectTranslationDocumentUserAssignments.
        - The new DocumentGuid member is used instead of the old DocName to identify documents.
        - The UserGuids member has been replaced by UserRoleAssignments (of type TranslationDocumentUserRoleAssignment), so that the document assignment role (Translatior/Reviewer1/reviewer2) and the deadline for this role can be defined for each user assigned.
- ListProjectTranslationDocuments
    - The ServerProjectTranslationDocInfo objects still have an array of UserInfoHeader objects. But the UserGuid member of the UserInfoHeader class has been replaced by an array of TranslationDocumentUserRoleAssignment objects, that can hold the DocumentAssignmentRole and DeadLine besides the Guid of the assigned users (UserGuid).

    - UserInfoHeader member of ServerProjectTranslationDocInfo has been replaced by TranslationDocumentUserRoleAssignmentDetails, which holds the UserInfoHeader, the DocumentAssignmentRole and the Deadline
- DeleteTranslationDocument
    - The docName parameter has been replaced by documentGuid
- PretranslateProject
    - The targetLangCodes parameter has been introduced.
    - Class PreTranslateOptions has a new UseMT (use machine translation) member
- PretranslateDocuments
    - The translationDocNames parameter has been replaced by translationDocGuids.
- GetStatisticsOnProject
    - The targetLangCodes parameter has been introduced.
    - Separate result is returned for each target language (in form of a StatisticsResultInfo object instead of a byte array)
- GetStatisticsOnTranslationDocuments
    - The translationDocNames parameter has been replaced by translationDocGuids.
    - The result is returned in a different way (StatisticsResultInfo instead of a byte array)

6.2 Summary of changes when migrating to 4.2.12 from previous version

6.2.1 Brief list of changes

Server project related

- AddLanguageToProject has been introduced

6.3 Summary of changes migrating to version 4.5

6.3.1 Major Conceptual changes

The way TMs are assigned to server projects have been changed.

Support for managing light resources (segmentation rules, auto translatables, non translatables, ignore lists, autocorrect lists, TM settings, filter configurations, keyboard shortcuts, export path rules and QA settings have been added.

Support for creating server projects with the new document handling mode called Desktop Documents.

6.3.2 Manual migration steps required

The installer of memoQ Server 4.5 is expected to update the WSIF.config file located under Documents and Settings\All Users\MemoQ Server. It adds the following XML fragment to the

<services> section of this file (to the same position as other existing <service> elements can be found in the file). This step if required only when upgrading from previous version of memoQ Server (not for a clean installation):

```
<service behaviorConfiguration="commonServiceBehavior"
    name="MemoQServices.ResourceService">
    <endpoint     address="ResourceService"     binding="basicHttpBinding"
    bindingConfiguration="basicHttpBindingConfig"
    behaviorConfiguration="commonEndpointBehavior"  name="serviceEndPoint"
    contract="MemoQServices.IResourceService" />
    <endpoint       binding="mexHttpBinding"       bindingConfiguration=""
    name="mtxEndpoint" contract="IMetadataExchange" />
      <host>
          <baseAddresses>
          </baseAddresses>
      </host>
   </service>
```

6.3.3 Brief list of changes

Server project API – Assigning TMs

When assigning TMs to a server project now it can be explicitely defined to which target language the specific TM is to be assigned to. Therefore, TMs are no longer implicitely assigned to a target language of the project based on the target language of the TM. This enables assigning a different set of TMs to target languages of the project differing only the minor part of the target language code (e.g. english us and english uk).

The Translation Memory API (ITMService):

- The ResourceInfo class has been split into two classes:
  - ResourceInfo class: The same as the old ResourceInfo class excluding metadata (Domain, Subject, etc.)
  - HeavyResourceInfo class: It is derived from the ResourceInfo class, and adds metadata fields (Domain, Subject, etc.). This is the base class of the TMInfo and TBInfo classes.
- The base class of TMInfo and TBInfo is HeavyResourceInfo instead of ResourceInfo.

Resource API – Managing light resources

API for managing light resources has been added via the new ResourceService service.

Server Project API – Light resource related

- SetProjectResourceAssignments added
  - Array of ServerProjectResourceAssignmentForResourceType objects as parameter
- ListProjectResourceAssignments added
- ListProjectResourceAssignmentsForMultipleTypes added
- ImportTranslationDocumentWithFilterConfigResource added

Server Project API – Statistics

- StatisticsOptions.ShowCounts_IncludeWhitespacesInCharCount added

Server project API – Creating and managing projects

A new project type has been introduced, called Desktop Docs. The "old" project type is called Live Docs. The management of both types of projects are exactly the same, the only difference is how projects are created.

The Server Project API (IServerProjectService):

- The CreateProject function behavior extended:
  - Accepts classes ServerProjectCreateInfo and ServerProjectDesktopDocsCreateInfo as the parameter; the type of the project will be a LiveDocs project in the first case and a DesktopDocs project in the latter case.
  - ServerProjectDesktopDocsCreateInfo class inherits from ServerProjectCreateInfo and extends it with options related to Desktop Docs projects only.
- ServerProjectInfo received a new member called DesktopDocs which is true for Desktop Docs projects.
- ServerProjectTranslationDocInfo received a new member called WorkflowStatus which contains the documents current workflow status.
- SetDocumentWorkflowStatus operation and class ServerProjectTranslationDocumentWorkflowStatusChange added
- DistributeProject operation added.
- DeliverDocument operation added.

6.4 Summary of changes migrating to version 5.0

6.4.1 Major Conceptual changes

Projects now have a new property which specifies if the documents in the project record their version history. This property is defined when the project is created and cannot be altered later. Every document in projects with version tracking has a major and a minor version number (different for each document in the project).

6.4.2 Brief list of changes

Server project API – Creating and managing projects

The Server Project API (IServerProjectService):

- ServerProjectCreateInfo received a new member called RecordVersionHistory which specifies that the project should record the versions of the documents.
- ServerProjectDesktopDocsCreateInfo received a new member called EnableWebTrans indicating whether to enable web based translation for the project (enabling both web based translation and split/join is not supported) .
- ServerProjectInfo received a new member called RecordVersionHistory which is true if the project records the versions of the documents.

- ReimportTranslationDocuments operation added.
- ServerProjectTranslationDocInfo was extended with four new members MajorVersion and MinorVersion that describe the current version of the document and ImportPath and ExportPath contain the file path the document was imported from, and exported to by default.

Server Project API – Pre-translation

The pre-translation options class (PreTranslateOptions)

- The member PretranslateLookupBehavior is now obsolete.
- New configuration options include members
  - LockPretranslated
  - ConfirmLockPretranslated
  - ConfirmLockUnambiguousMatchesOnly
  - FinalTranslationState

  which define new behavior for pre-translation. The options allow locking and confirming segments after pre-translation with specific options limiting when to perform this operation. Also the segment status can be changed by the process if a segment has been pre-translated.

Server Project API – Statistics

The statistics options class (StatisticsOptions)

- TagWordWeight and TagCharWeight fields added

The statistics result is changed: tag counts are included in the result.

Server Project API – Getting the version of the API

- GetApiVersion operation added

6.5 Summary of changes migrating to version 5.0.50

6.5.1 Major Conceptual changes

There are numerous conceptual changes in the API.

Error (fault) handling has been changed: the API now supports a new, advanced fault handling mechanism, and the old “legacy” mode.

User sessions are introduced. Sessions are used to perform certain actions impersonating a memoQ user. User sessions are created through explicit login with a user name and password. No existing services employ the user sessions; they still work with an implicit administrator user. Only new functions use the sessions.

The TM services offer lookup, concordance, and add/update entry services. These functions allow searching and modifications of TMs. These functions require a special license in the memoQ server, and require user sessions.
It is now possible to manage ELM licenses via the API.

Support for live documents has been added:

- Listing, creating and deleting corpora
- Assigning corpora to a server project, retrieving the list of corpora assigned to a server project

6.5.2 Brief list of changes

Error handling

From this point memoQ Server now supports two different modes: legacy mode and advanced error handling mode. When using TM API lookup/concordance/addupdateentry or the ELM API, make sure to use advanced error handling mode. See 202.5.1 Error handling configuration for the details.

File management API

Demo code demonstrating the use of the file management API has been added (C#).

Security API – User sessions

The Security API (ISecurityService):

- Login and a Logout function to create and terminate user sessions.

TM API – Lookup, concordance, add/update entry

The TM Service API (ITMService):

- LookupSegment function for single segment lookup in a TM
- Concordance for free-text search in a TM
- AddOrUpdateEntry for adding or updating an existing entry in a TM
- All of the above functions require a valid user session, and perform user authorization to check if the user has the required permissions regarding the TM.

The TMInfo class:

- A new member AllowReverseLookup is added which is a new property of TMs. It can be used to create TMs allowing reverse lookup and to query if a particular TM allows it.

LiveDocs API

This is a new API. Supports creating, publishing, deleting and listing corpora.

ELM API

This is a new API. Supports:

- Managing license assignments (assigning licenses to users and listing assignments, etc.)
- Managing license permissions (adding license permissions to users/groups, listing license permissions, etc.)
- Listing project license permissions
- Listing ELM license pools

Server project API changes

Project license permission can be added to users:

- ServerProjectUserInfo.PermForLicense has been added
- ServerProjectUserInfoHeader.PermForLicense has been added

6.6 Summary of changes migrating to version 6.0

6.6.1 Major Conceptual changes

There are a number of changes in the API ion version 6.0.

Some previously marked obsolete functions and related classes have not been removed. These affect the IServerProjectService interface's SetProjectTMs and ListProjectTMs functions, and the PretranslateOverwriteBehavior enumeration.

The memoQ Bilingual (mbd) format is no longer supported. Exporting into this format or updating a document in a server project is not available, but importing is still possible.

6.6.2 Brief list of changes

Server Project API – Project TM management

Removed classes and functions

- ServerProjectTMAssignments class
- IServerProjectService.SetProjectTMs
- IServerProjectService.ListProjectTMs

Server Project API – PreTranslation

Removed classes and members

- PretranslateOverwriteBehavior enum
- PreTranslateOptions.PretranslateLookupBehavior

Server Project API – Project and document information

ServerProjectInfo and ServerProjectTranslationDocInfo classes received new properties containing more detailed statistics about the project/document, listing segment, character, and word counts.

Server Project API – Document export and update

Removed functions

- IServerProjectService.ExportTranslationDocumentAsMbdBilingual
- UpdateTranslationDocumentFromBilingual no longer accepts memoQ Bilingual (mbd) format (they can be imported, but not used for update).

XliffBilingualExportOptions has a new member FullVersionHistory to export complete major and minor version history into the (compressed) xliff.

6.7 Summary of changes migrating to version 6.2.14

6.7.1 Major Conceptual changes

Asia Online service appeared in API version 6.2.

6.7.2 Brief list of changes

TBService

ListTBs2 operation has been introduced. This has finer control than ListTBs on how language filters are to be used when filtering the returned TB list.

ServerProjectService

ServerProjectService.SetProjectTBs behavior has been made consistent with the UI. See function documentation for details.

Asia Online API

This is a new service over the API. Supports:

- Getting the language pair code for selected documents.
- Listing the supported domain combinations for the language pairs.
- Submitting a documents with translation options to Asia Online translation operation.
- Querying the status of the translation.
- Getting project IDs supported by Asia Online for the specified user account.

6.8 Summary of changes migrating to version 6.5

6.8.1 Brief list of changes

New light resources

New light resource types (LiveDocs settings, LQA, WebSearch settings) are supported for creation, import, and assignment to project and project resources (except WebSearch settings).

New properties of server projects

ServerProjectInfo has been extended with properties of server projects which were defined when creating a project, but could not be queried.

X-translate API

This is a new operation in the API. Supports X-translation on specific documents of the server project.

Getting/setting light resource permissions

This documentation has been extended to describe how to set/get permissions on light resources. See ISecurityService for the details.

Automatic TB language maintenance in AddLanguageToProject function

The AddLanguageToProject function has been extended with automatic TB language maintenance. The ServerProjectAddLanguageInfo has been extended with a new parameter. Based on the value of this parameter the AddLanguageToProject function will

- extend the languages of the project termbases with the new target lang if needed
- remove the project termbases which do not contain the new target language
- throw a fault if the the project termbases do not contain the new target language

6.9 Summary of changes migrating to version 6.7

New properties of server projects

ServerProjectInfo has been extended with the PreventDeliveryOnQAError property. The DeliverDocument function of the IServerProjectService interface throws if this property is true and the document contains QA errors.

ServerProjectInfo has been extended with the AllowPackageCreation and with the AllowOverlappingWorkflow properties. For projects with packages the resources in the package can be configured via the PackageResourceHanding propery. Creating the project (ServerProjectCreateInfo) has been extended with the same properties.

ServerProjectCreateInfo.DownloadSkeleton2 and ServerProjectCreateInfo.DownloadPreview2 properties replace the same properties in ServerProjectDesktopDocCreateInfo. The previous properties are kept for compatibility but the use of the new ones are recommended.

Project Management

Server project management interface is extended with package management options: listing, and downloading packages, uploading and processing deliveries, creating and updating projects from packages, and creating delivery packages from such projects.

Document history API

This is a new operation in the API. With this operation we are able to the the documentum history for the project documents.

New property of users

UserInfo has been extended with the PackageWorkflowType property.

TB prioritization

Server project TB handling has been made consistent with the TB prioritization changes.

Structural alignment

LiveDocs services can structurally align two documents and the matching segment pairs can be downloaded as a TMX.

IceSpice

WS-API has been extended to be able to support IceSpice TMs.

6.10 Summary of changes migrating to version 6.7.2

Run QA

Server project management interface is extended with running QA checks on a project or specified documents, and getting a report of the QA errors and warnings in the scope.

6.11 Summary of changes migrating to version 6.8.50

CAL licensing

ELM related functions all apply to CAL license management.

Post translation analysis report

Server project management interface is extended with running post-translation analysis and retrieving the report on a project or specified documents. New analysis can be executed and the result of existing reports can be downloaded as entities or exported as a CSV.

6.12 Summary of changes migrating to version 6.8.55

Pre-translation option to copy source to target

IServerProjectManager.PretranslateOptions class is extended with a member PretranslateCopySourceToTargetBehavior which adds the option to copy the source segment to the target during pre-translation.

6.13 Summary of changes migrating to version 6.8.58

Alignment with segmentation

ILiveDocsService.AlignDocumentsGetTmx function is extended with options for segmentation and turning off structural alignment. Default behavior is no segmentation and structural alignment (previous behavior of the function).

Reimport of documents with filter configuration resource in memoQ Server

IServerProjectService.ReImportTranslationDocumentsWithFilterConfigResource function allows reimporting documents and specifying the filter configuration by the resource guid if the filter configuration resource is stored in memoQ Server.

6.14 Summary of changes migrating to version 7.0

Creating server projects

Server projects are now always created in “desktop document” mode. The class ServerProjectCreateInfo is obsolete, and all methods that accept it as parameter will behave as if a ServerProjectDesktopDocsCreateInfo was used. Methods IServerProjectService.CreateProject and CreateProjectFromPackage are now obsolete. They behave as described above. New methods (CreateProject2 and CreateProjectFromPackage2) are provided for creating projects.

Images in projects

Server projects can contain images. Image files are imported as regular translation documents. These images can be exported into a localization package and imported back into the project after localization (see IServerProjectService. CreateImageLocalizationPack and ImportImageLocalizationPack). See ServerProjectTranslationDocInfo.IsImage property to determine whether a translation document is an image.

Embedded objects and images

Embedded objects, such as an Excel spreadsheet embedded in a Word document, can be imported as separate documents. Such embedded objects, and embedded images appear in the project as regular translation documents, but they are also linked to their parent (see ServerProjectTranslationDocInfo.ParentDocumentId). The default behavior for import is that embedded objects and images are not imported. To import them, use a new import function IServerProjectService.ImportTranslationDocumentsWithOptions or use a filter configuration which specifies embedded objects filter configuration too. When exporting a document that has linked child documents (its embedded objects), the child documents will always be exported automatically and substituted into the parent document.

Reviewer 1 confirmed

The reviewer 1 confirmed segment counts are added to ServerProjectInfo and ServerProjectTranslationDocInfo. Pre-translate, X-translate and Confirm and update operation settings objects are extended with new options to filter for this row status.

Working and master TMs

The concept of working and master TMs are introduced in server project TM assignment.

Document history

Some new history item type have been introduced.

Document export

When exporting unfinished documents (with untranslated rows or with errors) a new method IServerProjectService.ExportTranslationDocument2 allows specifying how to handle unconfirmed, untranslated and erroneous rows.

6.15 Summary of changes migrating to version 7.0.50

Statistics and post-translation analysis options

Getting an analysis of a project has two new options (in class StatisticsOptions) regarding cross file repetitions (RepetitionPreferenceOver100) and repetition preference over 100%

matches (DisableCrossFileRepetition). Creation a post-translation analysis has the same new RepetitionPreferenceOver100 option.

memoQWebTrans URL

If the memoQ Server has memoQWeb enabled and configured, for web-enabled projects the document information includes the memoQWebTrans URL of the document.

Advanced document-user assignment modes

From now on the advanced document-user assignment modes are supported over the API. The old assignment functions still exist but you cannot use these functions if you would like to use FirstAccept, GroupSourcing or subvendor assignments. The new methods: IServerProjectManager.ListProjectTranslationDocuments2, IServerProjectManager.ListTranslationDocumentAssignments and IServerProjectManager.SetTranslationDocumentAssignments.

Callback for notification about delivery

MemoQ Server is capable of notifying a web service about delivery operations executed not via the WS API. A callback URL must be provided when creating a new server project (ServerProjectDesktopDocsCreateInfo.CallbackWebServiceUrl or for existing project changing project properties using ServerProjectUpdateInfo.CallbackWebServiceUrl). Each document in the projects can have an identifier provided by external system, which identifier is sent back during the callback. This ID is set when importing documents using ServerProjectService.ImportTranslationDocumentsWithOptions using the member ImportTranslationDocumentOptions.ExternalDocumentId).

See Section 4 for more details.

List subvendor groups

ISecurityService.ListSubvendorGroups function lists the subvendor groups of the server.

6.16 Summary of changes migrating to version 7.0.65

Specify a path to store with the document during import

When importing translation documents using IServerProjectService. ImportTranslationDocumentsWithOptions, the path to save as the import path of the document can be specified using the field ImportTranslationDocumentOptions.PathToStoreAsImportPath).

6.17 Summary of changes migrating to version 7.0.67

Project status has been introduced in ServerProjectInfo class

The ServerProjectInfo class contains information about the actual status of the project. This status can be Live and WrappedUp.

6.18 Summary of changes migrating to version 7.5

Confirm and Update options

Confirm and Update has a new options (in class ConfirmAndUpdateOptions). From now it can be exactly specified in which role will be the segments confirmed and sent to the TM. The previous MakeRowsProofread options, which sets translation state to Translator or Reviewer 2 confirmed, is obsolete.

RunQAGetReport

The method IServerProjectService.RunQAGetReport returns a per document detailed report about the number of errors and warnings. Previously, this report did not contain documents without any error or warning. This behavior is changed: the report per document contains an item for all documents the operation is executed on. For documents without errors and warnings the report item indicates 0 error and warning count.

Project default custom meta values can be set and queried

The project default custom meta values can be set on project creation and project update. The ServerProjectCreateInfo and the ServerProjectUpdateInfo classes have been extended with the CustomMetas member. The ServerProjectInfo class also has a new member (CustomMetas), which contains the default meta values of the project.

6.19 Summary of changes migrating to version 7.5.50

Allow same user in multiple roles

The same user can be assigned to the same document in different roles as well.

Allow separate subvendor assignment in TR/R1/R2 roles

Different subvendor groups can be assigned to the same document in different roles.

Subvendor group management

The UserInfo and the GroupInfo classes have been extended with a new member. These fields indicate whether the current user/group is subvendor manager/group or.

The ListUser and the ListGroups functions work as earlier, do not give back the subvendor managers/groups. The caller can use the ListSubvendorManagers and the ListSubvendorGroups functions to get the list of the subvendor managers and the subvendor groups.

Create, get, update and delete user/group operations work with the subvendor managers and groups as well.

There are two new functions which can be used if the caller wants to assign/remove a user to/from a subvendor group. These functions are: AddSubvendorManagerToGroup, RemoveSubvendorManagerFromGroup.

6.20 Summary of changes migrating to version 7.8.50

Template-based project creation

The function IServerProjectService.CreateProjectFromTemplate can be used to create online projects from templates.

Project template resource management

The light resource service supports the management of the project template resources:

- Creating, publishing, cloning, deleting and listing template resources
- Importing from and exporting to memoQ resource format

Export of multilingual documents

Multilingual documents are created when a file is imported using one of the multilingual filters. A single file is imported as multiple docuents into multiple target languages. The export of such documents is special, because the documents can be merged togather to produce a single result file. Document export behavior is changed to support multilingual documents. The IServerProjectServive.ExportTranslationDocument operation will always perform this merging, while the caller can specify the expected behavior in IServerProjectServive.ExportTranslationDocument2.

6.21 Summary of changes migrating to version 7.8.53

Heavy resource management

The TM/TB/LiveDocs services support:

- Update the properties of the resource
- Get the resource info based on the guid of the resource

6.22 Summary of changes migrating to version 7.8.54

Project management

Wrap up project support.

6.23 Summary of changes migrating to version 7.8.100

ELM

ELM license functions support a new license pool type: Translator Light.

TM/TB lookup/concordance/addentry, login, session handling

TM/TB lookup/concordance/addentry, login, session handling are marked deprecated. These will be eliminated in version 8.0. The new HTTP API is to be used instead.