

Interacting with the Project Tracker Web Service: An Overview

Table of Contents

Introduction.....	1
Invoking a Project Tracker service	1
Platforms.....	2
Interfaces.....	2
The Web Service API.....	2
Summary of operations.....	2
XML Validation on the client side.....	3
Handling failure/errors.....	3
SOAP fault messages.....	4
Synchronizing with Project Tracker.....	5
Example.....	5
Potential error conditions during synchronization.....	6

Introduction

This document describes at a high level how customers can interact with Project Tracker without using the Web interface.

Customers need to be able to interact with Project Tracker without using the Web interface. For example, they may need to synchronize data between Project Tracker and a third party application, such as Test Director, Telelogic DOORS, or salesforce.com. Other customers may want to be able to import data from Excel. Yet others may want to use a third party user interface (e.g., an Eclipse plugin) to view and edit Project Tracker artifacts.

Invoking a Project Tracker service

To invoke a project tracker service, clients will need to follow these steps:

1. Authenticate.
2. Create the appropriate request document.
3. Invoke the 'execute' Web Service method with the document passed as its argument
4. Receive the Web Service result as an XML Document.
5. Process the received document as needed.
6. Repeat steps 2 to 5 as needed.

Platforms

Client applications (users of the interface) can be written in any language that supports connecting to Web Services.

Interfaces

The Web Service API

The following Web Services make up the API:

- **SystemStatusService:** Provides system level information
- **Dispatcher:** Provides document API to Project Tracker to query, create and update artifacts.
- **MetadataService:** Provides the ability to query Project Tracker metadata, modify attribute options.
- **AttributeService:** Provides the ability to add attribute options, etc. to artifact types
- **QueryService:** Provides the ability to determine named queries (DDQs) that can be executed. In the future it will provide the ability to define, update and delete named queries as well.
- **AttachmentService:** Provides the ability to upload and download attachments associated with Project Tracker artifacts.
- **HistoryService:** Provides the ability to query the history of a set of Project Tracker artifacts.

A detailed description of the API is available in [api.html](#)

Summary of operations

The API is implemented as a set of Web Services. Table 1 summarizes the Web Services and the operations they provide:

Table 1 Summary of operations

<i>“Operation”</i>	<i>“Returned type”</i>	<i>Needed for use case</i>	<i>P or D¹</i>
getArtifactList	artifactList	Create/Update/Eclipse/Sync	P
getArtifactById	artifactList	Eclipse	P
getNextPage	artifactList	Create/Update/Eclipse/Sync	P
updateArtifactList	artifactList	Create/Update/Eclipse/Sync	P
createArtifactList	artifactList	Create/Update/Eclipse/Sync	P

¹ Does the API in question work on Project (P) or Domain (D) or both (P,D).

<i>“Operation”</i>	<i>“Returned type”</i>	<i>Needed for use case</i>	<i>P or D</i>
getSchema	Schema	All	P,D
loadMetadata	metadataList	Project Template Installer	D
saveMetadata	loadMetadata	Project Template Installer	D
removeMetadata	metadataList	Project Template Installer	D
getArtifactMetadata	artifactMetadata	Eclipse	P, D
getArtifactTypes	artifactTypeList	Eclipse	P,D
getQueries	queryList	Eclipse	P,D
addAttributeOption	Void	Create/Update/Eclipse/Sync	P,D
getAttributeOptions	attributeOptionList	Create/Update/Eclipse/Sync	P,D
addAttachment	artifactList	Create/Update/Eclipse/Sync	P
removeAttachment	void	Create/Update/Eclipse/Sync	P
getAttachment	attachment	Create/Update/Eclipse/Sync	P
getArtifactHistory	ArtifactHistoryList	Eclipse/Sync	P
getArtifactChanges	HistoryTransactionList	Sync	P
getCurrentTime	long	Sync	D
getVersion	Version	All	P,D
getProjectInfo	ProjectInfo	All	P,D

XML Validation on the client side

Since the Web Service exchanges generic Document objects, the Web Service WSDL file doesn't provide any information what the content of these documents should be.

To facilitate document validation, clients can download (access) the schema associated with a particular project using the getSchema API.

Handling failure/errors

The proposed API handles errors differently depending on their types:

- **Invalid documents:** If a document doesn't match its corresponding XML Schema document, then the entire document is rejected and a SOAP fault message is returned. See below for the format of the SOAP fault message.
- **Unexpected (unchecked) errors:** A SOAP fault message is returned.
- **Artifact/attribute business rule violations:** If a business rule is violated during the update/creation of an artifact, then the response document (*artifactList*) contains information about the failures. The reason to handle such failures differently is that in an update/create scenario some artifacts may be updated successfully but some may fail. When this happens, it is important to return

information about both the successful and the unsuccessful artifacts. We could use the SOAP failure message to contain both the unsuccessfully and successfully updated artifacts, but this would make the client more complicated than necessary.

- **Network connectivity is lost during the use of the API.** If network connectivity is lost between API calls, the client will not be able to make calls to the server after the network connectivity is lost. If the network connectivity is lost during an API call, the possibilities are:
 - The request is received fully, but the response cannot be sent fully. In that case the server will process the request, but the response will be lost. A log message will be written in the standard server log indicating that failure.
 - If the request is not received fully, the server will not process the request. A log message will be written in the standard server log indicating that failure.

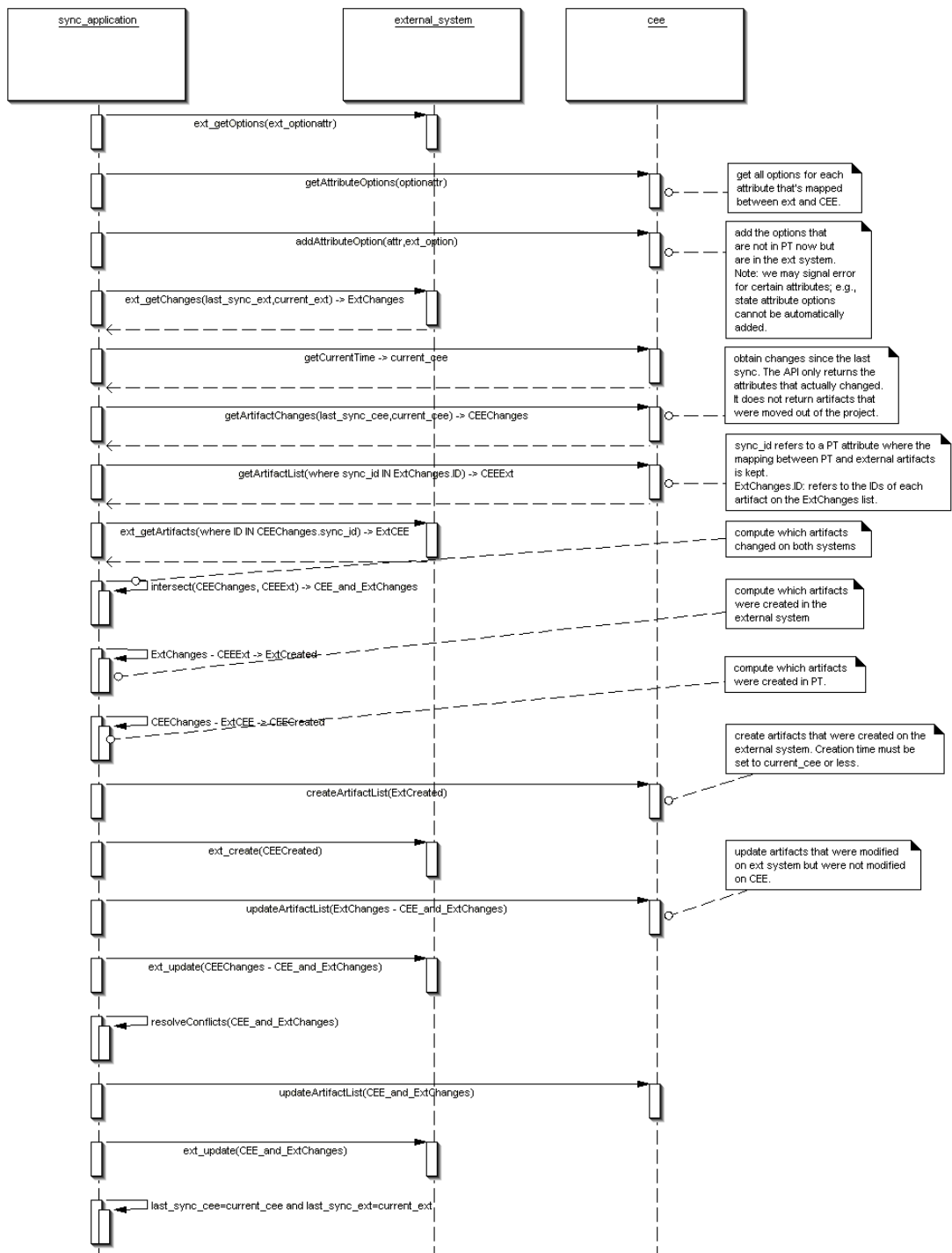
SOAP fault messages

In the case of an error the Web Services return SOAP fault messages. At minimum, Web Services can return a WSException SOAP fault message (they may declare additional failure types). A WSException can be used to wrap arbitrary server side exceptions. The WSException message contains the following information:

- The fully qualified class name of the original Exception.
- The error message of the original Exception.
- The server side stack trace of the original Exception.

Synchronizing with Project Tracker

Example



Some comments on this diagram:

- Authentication is not shown as a separate step. Of course the sync application will need to authenticate with the external system and will need to authenticate with CEE.
- Operations starting with *ext_* are operations that the external system's API must support.
- This interaction diagram assumes that the *ext* system and the sync application live on the same hardware; that is why there is no need to get the "ext current time" value, and simply using the current time as detected by the sync application is sufficient.
- The assumption is made that Project Tracker keeps track of the IDs of corresponding external system artifacts (the diagram refers to *sync_id*).
- The assumption is that in *updateArtifactList* and *createArtifactList* calls the modified and created times can be specified to occur in the past (i.e. instead of Project Tracker recording when the actual update is made, it can record the time when the artifact was created/last updated in the external system).
- The *resolveConflicts* operation can resolve conflicting changes at the attribute level.
- The above diagram shows how updates occur between the two systems when there are no attachments involved. If attachments are involved, then after the *createArtifactList/updateArtifactList* calls additional calls are made to add attachments to the appropriate artifacts.

Potential error conditions during synchronization

The following errors may occur during synchronization:

- Network connectivity lost. If network connectivity is lost midstream in the process, the sync application should try to resume the process when the network is back up again.
- *createArtifactList* and *updateArtifactList* may fail because user associated with the external system's artifact does not exist or exists under a different username in CEE. The artifact cannot be synchronized with CEE.
- *Ext_create* and *ext_update* may fail because CEE user associated with a Project Tracker artifact does not exist in the external system or exists under a different username.
- *addAttributeOption* may fail if between the time *getAttributeOption* is called and *addAttributeOption* is called the option in question is added by some other means (e.g., a different Web Service call or a user creates the option through the web UI).

- Stale update exception may be thrown by *updateArtifactList(ExtChanges - CEE_and_ExtChanges)* and *updateArtifactList(CEE_and_ExtChanges)*, if after the *getArtifactChanges(last_sync_cee,current_cee)* and *getArtifactList(where sync.id IN ExtChanges.ID)* calls changes are made either to artifacts in *CEEChanges* or *CEEExt* by some other process (e.g. a user through the web UI or another Web Service).
- Stale update exception may be thrown by external system (*ext_create*, *ext_update*)
- The diagram does not show what happens if an attribute option exists in CEE that does not exist in the external system. It may cause exception; alternatively there may be APIs in the external system to create the necessary options automatically.

Skipping state attribute options during sync

A possible scenario is where the ‘state’ attribute of the external artifact changes more than once between two syncs. E.g., a bug may go from New to Open to Fixed. During the sync process the CEE version of the same artifact needs to be updated to show ‘Fixed’ in its state attribute; however, that may not be a valid transition in CEE. This may throw an exception. (An alternative is to relax the state transition validation on the CEE/external system side to allow illegal transitions.)

Option filter conflict

In CEE a change in one attribute may limit the available options in another attribute (e.g., Platform > OS). Let’s assume that the system is set up that the external system wins if there is a conflict in Platform, or OS, but if there is no conflict, then the sync is bidirectional. Let’s assume that the user of the external system changes the dependent attribute (i.e. OS, from Windows 95 to Windows XP) and the user of CEE changes both Platform and OS (from x86 to SPARC and from Windows 95 to Solaris 8). During sync there will be no conflict with Platform, therefore Platform from CEE will be kept in both systems (SPARC). Since both systems changed the OS value, the external value “wins”, and the OS will be set to “Windows XP”, which is clearly not a valid combination with the “SPARC” platform. The external system or CEE may throw an exception.

Dependent required values

Project Tracker has the notion of dependent required values. Such notion may or may not exist in the external system, or the dependency rule may be different between the two systems.

Even if the two systems are identical in such capabilities and the dependency rules are configured correctly, there is still a problem analogous to the option filter conflict described above.

Dealing with comments

Comments are different from other attributes in that their values are cumulative. This can lead to other types of inconsistencies. For instance, it is possible to have two contradicting

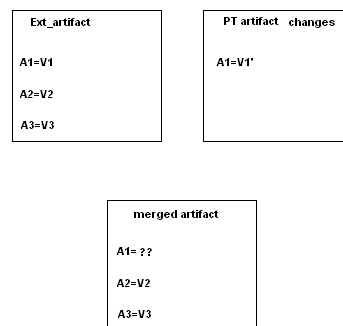
comments to be entered (one in the external system and one in CEE); during sync time these two contradictory comments will be part of both the external and CEE artifact. (The assumption is made that no conflict resolution is needed for comment fields).

It is also possible to enter a comment while making a change to another attribute; if there is a conflict in that attribute then the comment may be disconnected from the attribute change. E.g., imagine that both the external and CEE users change the priority of the artifact (external: P2 -> P3; CEE P2 -> P1); also, assume that the CEE user actually enters a comment explaining the change (“increasing priority ...”, while the external user does not. If the system is set up to resolve conflicts between the “priority” attributes by letting the external system win, then we will have a situation where the “priority” field reflects the value entered by the external user but the contemporaneous comment will be entered by the CEE user. So, the end result is that we see the artifact priority change from P2 to P3, while the comment states “increasing priority...”

Many of the above problems can be mitigated by entering comments in both systems indicating that a conflict was detected and resolved, so users reading the comments can at least start to make sense as to what may have happened.

External change detection limitations

The external change detector may have limitations; for example, it may not be able to determine which attributes changed, it may only be able to determine that an object changed. In other words, *ext_getChanges(last_syn_ext,current_ext)* may return full objects instead of just the list of attributes that changed in the given time period. In this case the synchronizer may not be able to determine what changes need to be propagated from the external system to Project Tracker. For instance, imagine the scenario shown on the diagram below:



As shown on the diagram, there is no way to know what the value of A1 should be in the merged artifact. In order to know what to set that attribute to, we would need to know what the value of A1 was in the Ext_artifact during the last sync; if the value was V1 (i.e. not changed), then we could safely change the value to V1'. However, if the value was V1'', then we would need to follow conflict resolution rules to resolve the conflict.

A solution to this is to not only obtain the new values in Project Tracker (V1') but also to obtain the value that the attribute had at the last sync time (V1''). If V1''' is the same as V1 then there is a strong likelihood that attribute A1 did not change during the sync

period. However, this is not necessarily the case, because it is possible that a user changed A1 from V1 to V1'' and then at a later time (before the sync) changed it back to V1.