Telelogic® Focal Point™

Web Services API version 2.0

Reference Manual

Document version 1.1

15 September 2008

General

This document describes the Web Services API version 2.0 for Focal Point version 6.2. Example files `JavaWebServices2Example.zip` are available at Telelogic Support Web.
Note: Some of the described functionality is first released in Focal Point 6.3, as detailed below.

WSDL

The Focal Point Web Services API uses WSDL (Web Services Description Language) for defining the services. The current WSDL can always be downloaded from a Focal Point server at *http(s)://<your Focal Point server URL>/fp/services2/FPServices?wsdl*
for example: http://focalpoint.example.com/fp/services2/FPServices?wsdl.

The WSDL version used is 1.1.

Data types are defined using an included XML Schema. The data types are described below in the section *Schema guide*. This document describes the data types, whereas the XML Schema documents the formal details. Both documents are needed to fully understand the Focal Point Web Services API.

The WSDL uses the document/literal wrapped pattern.

SOAP

The Focal Point Web Services API uses SOAP for calling the services.

The SOAP version used by the server is 1.1.

The URL to the SOAP endpoint is http(s)://<your Focal Point server URL>/fp/services2/FPServices
for example: http://focalpoint.example.com/fp/services2/FPServices.
The URL is also found through the WSDL.

Faults

A service may sometimes need to return a SOAP Fault instead of a value. This can be the due to a user error (for exampling referencing an element that does not exist), or due to an internal server error. The faults that may be returned are documented in this document and also in formal detail in the WSDL.

Security

When SOAP messages are sent over HTTP they are sent in clear text, which means that anyone who has access to the traffic can read the sent data, for example user name and password.

For security reasons, we suggest using HTTPS instead. When using HTTPS, all communication with the Focal Point server is encrypted (not only SOAP messages).

Please refer to the Focal Point Installation Guide for a description of how to set up HTTPS for Focal Point.

Aliases

All resources in Focal Point are identified by aliases. For example, each workspace in Focal Point can be identified by a unique alias. The following resources can have aliases: workspaces, modules, views, elements and attributes. Modules and views are grouped together and are collectively known as domains. Resources are always assigned a default random alias, which later can be changed to something more readable or easier to remember. Workspace, domain and attribute aliases can be edited in each workspace from navigation bar *Configure > Alias.*

An alias is a string that can contain any unicode character, but it is not advisable to use special or invisible characters (such as line breaks).

When exporting and importing a workspace in Focal Point, aliases will be preserved for domains, attributes and workspace (if possible).

Elements

Aliases for elements are unique for a Focal Point installation. That means that two elements cannot have the same alias.

Element alias can be assigned when an element is created (using *addElement*, *addElements*, *writeElement* or *writeElements*) but cannot be changed later. If no alias is supplied when the element is created, a new random alias will be generated. New aliases will be generated when a module or workspace is imported.

Domains

Modules and views are collectively known as domains, that is, a domain can either be a module or a view. Aliases for domains are unique within a workspace. There can, for example, be a module in workspace A and a view in workspace B with the alias *requirements* but they cannot co-exist in the same a workspace. A domain is uniquely identified with a workspace alias and a domain alias.

A domain alias can be assigned when the domain is created, and can also be changed by a workspace administrator at a later time. When a workspace is imported, domain aliases are preserved.

Workspaces

Aliases for workspaces are unique within the Focal Point installation. That means that two workspaces cannot have the same alias.

A workspace alias can be assigned when the workspace is created , and can also be changed by a workspace administrator at a later time. When a workspace is imported, the

alias is preserved unless there is another workspace with the same alias.

Attributes

Alias for attributes are unique within a module. This means that the same attribute can exist in different modules, whereas two attributes within the same module cannot have the same alias. For example, there can be a workspace with two modules, with the aliases *requirements* and *defects*, each with an attribute alias *estimated_time*. There might even be another workspace containing a *requirements* module with an *estimated_time* attribute. An attribute is uniquely defined by workspace alias, module alias and attribute alias.

An attribute alias can be assigned when the attribute is created, and can also be changed by a workspace administrator at a later date. When a workspace or module is imported, aliases are preserved.

Service overview

Name	Short description	Applies from Focal Point version
addElement	Creates a new element.	
addElements	Creates several elements in one batch	
authenticate	Authenticates a user (log in).	
endSession	Invalidates a session and release licenses (log out)	
getAttributeDefinitions	Retrieves a list of attribute definitions for a specified domain in a workspace.	
getAttributeHistory	Retrieves the change history for an attribute of an element.	
getChangedElements	Retrieves aliases for elements that have changed within a specified time range.	
getElement	Retrieves a single element and its attributes.	
getElementHistory	Retrieves the change history for an element.	
getElements	Retrieves elements for a workspace and/or domain.	
getElementsByAlias	Retrieves elements from a list of element aliases.	
getExternalLink	Retrieves a URL to an element.	6.3
getFileContent	Retrieves the file data contents of a file attribute	

getHistory	Retrieves the change history for a domain.	
getModules	Retrieves a list of modules in a workspace.	
getViews	Retrieves a list of views that the current user has access to.	
getWorkspaces	Retrieves a list of workspaces that the current user has access to.	
lookupAttributeAlias	Translates an attribute ID into an alias	6.3
lookupElementAlias	Translates an element ID into an alias	6.3
lookupViewAlias	Translates a view ID into an alias	6.3
lookupWorkspaceAlias	Translates a workspace ID into an alias	6.3
updateElement	Changes the specified attributes of an existing element.	
updateElements	Updates several elements in one batch.	
writeElement	Creates an element or updates it, if it already exists.	
writeElements	Writes several elements in one batch.	

Authentication

authenticate

Authenticates (logs in) a user. *authenticate* has to be called before any other service is used. A user will stay authenticated as long as the session is active, time out will occur after 1-1,5 hours of inactivity. A user can be logged in via the user interface and this API concurrently, as the logins will be handled as two different sessions.

A session can be explicitly ended by calling *endSession*. Licenses will be held until the session is ended or times out.

Returns

authenticate returns an XML Schema string as a token. This token can be used for subsequent calls.

Parameters

Name	Description	Optional	Type
username	the user's login name	no	XML Schema string
password	the user's password	no	XML Schema string

Faults

Name	Returned when
AuthenticationFailedFault	the username password combination is not accepted

The AuthenticatedFailedFault contains one of the *authenticationFailedReason*s below:

Reason	Description
accountLocked	The account is locked for login.
waitForDelay	If *login delay* is enabled there is a delay between every login attempt. The delay is 5 seconds after the first attempt, 15 seconds after the second attempt, 60 seconds after the third attempt, 5 minutes after the fourth attempt, and 1 hour after more than four attempts.
incorrectPassword	The password was not correct for this user.
licenseError	There was an error getting a license.
unspecified	Authentication failed for any other reason.

endSession

endSession logs out a user. The session and token are invalidated, and held licenses are released.

Returns

Nothing.

Parameters

Name	Description	Optional	Type
token	token for the session that should be ended.	no	XML Schema string

Faults

Name	Returned when
NotAuthenticatedFault	there is no active session for this token.

Reading data

getElement

Retrieves a single element and its attributes.

Element alias must be specified. View may be empty, nil/null or left out. If no view is specified, the element will be retrieved from its module.
Note: Regular users can only access elements via views that they are granted access to. To be able to retrieve an element from a module, the current user must be a workspace administrator. For regular users, a view that the user has access to must be specified, and the element has to be part of that view.

Returns

An *Element* is returned by *getElement*. See the section *Schema guide* below.

Note that the attribute definitions are not included. They can be retrieved independently by calling *getAttributeDefinitions*.

Parameters

Name	Description	Optional	Type
element	alias for the element	no	XML Schema string
view	alias for a view	yes	XML Schema string
token	authentication token	no	XML Schema string

Faults

Name	Returned when
ElementNotFoundFault	no element matches the alias.
AccessDeniedFault	the current user does not have access to element or the view (or the module, if no view was specified).
DomainNotFoundFault	the specified view or the module (if no view was specified) cannot be found.

InvalidArgumentFault	no, null or empty element alias is supplied
NotAuthenticatedFault	no session is active for the supplied token

Examples

Example set up: a module with alias *module1*; two views based on module1, *view1* and *view2* and an element in module1 with alias *element1*. The element is part of view1 but not view2.

Scenario 1: a regular user (with access to view1) calls *getElement* with Element set to element1 and View set to view1. Result: element1 is returned with the attributes visible in view1.

Scenario 2: a workspace administrator calls *getElement* with Element set to element1 and View set to nil/null. Result: element1 is returned with all attributes in module1.

Scenario 3: a regular user (with access to view2) calls *getElement* with Element set to element1 and View set to view2. Result: an AccessDeniedFault is returned since element1 is not part of view2.

Scenario 4: a workspace administrator calls *getElement* with Element set to element2 and View set to nil/null. Result: an ElementNotFoundFault is returned, since there is no element with alias element2 (in this example).

Scenario 5: a user (any type) calls *getElement* with Element set to empty string and View set to view1. Result: an InvalidArgumentFault is returned.

Scenario 6: a workspace administrator calls *getElement* with Element set to element1 and View set to module1. Result: a DomainNotFoundFault is returned, since module1 is not a view.

getElements

Retrieves elements for a workspace and/or domain.

If both workspace and domain are specified, elements for that combination will be retrieved. If workspace is not specified (left out, empty or nil/null), elements for the domain alias in all workspaces that the current user has access to will be retrieved.

If the user is a workspace administrator, the domain may be a module or a view. If the user is a regular user, the domain must be a view that the user has access to.

Returns

An *ElementsResult* is returned by *getElements*. See the section *Schema guide* below.

Parameters

Name	Description	Optional	Type
workspace	alias for a workspace	yes	XML Schema string
domain	alias for a domain	no	XML Schema string
token	authentication token	no	XML Schema string

Faults

Name	Returned when
AccessDeniedFault	the user does not have access to the workspace
DomainNotFoundFault	no matching domains are found
InvalidArgumentFault	domain is left out, empty or nil/null
NotAuthenticatedFault	no session is active for the supplied token
WorkspaceNotFoundFault	the workspace cannot be found

Examples

Example set up: two workspaces, one with alias *workspace1* and the other with *workspace2*. In both workspace1 and workspace2 there are modules with alias *module1* and views based on module1 with alias *view1*.

Scenario 1: A regular user (with access to view1 in workspace1) calls *getElements* with Workspace set to workspace1 and Domain set to view1. Result: all elements in view1 in workspace1 are returned.

Scenario 2: A regular user (with access to view1 in workspace1 and view1 in workspace2) calls *getElements* with Workspace set to nil/null and Domain set to view1. Result: all elements in view1 in workspace1, and all elements in view1 in workspace2 are returned.

Scenario 3: A regular user (with no access to workspace1) calls *getElements* with Workspace set to workspace1 and Domain set to view1. Result: an AccessDeniedFault is returned.

Scenario 4: A regular user (with access to workspace1) calls *getElements* with Workspace set to workspace1 and Domain set to module1. Result: a DomainNotFoundFault is returned (since regular users don't have access to modules).

Scenario 5: A workspace administrator (with access to workspace1 and view1 in workspace1 and access to workspace2 but not view1 in workspace2) calls *getElements* with Workspace set to nil/null and View set to view1. Result: An ElementsResult is returned with all elements in view1 in workspace1 and also an error with ErrorCode domainNoAccess (since the user does not have access to view1 in workspace2).

Scenario 6: A regular user calls *getElements* with Workspace set to workspace3 and Domain set to view1. Result: a WorkspaceNotFoundFault is returned since there is no workspace with alias workspace3 (in this example).

Scenario 7: A regular user (with access to workspace1) calls *getElements* with Workspace set to workspace1 and Domain set to nil/null. Result: an InvalidArgumentFault is returned.

getElementsByAlias

Retrieves elements from a list of element aliases. This can be useful as an alternative to calling *getElement* for each element that you want to retrieve. Just like *getElements, getElementsByAlias* will include information about attributes, i.e. *AttributeDefinitions*. There is however an upper limit of 100 elements. To retrieve more than 100 elements, you will have to call *getElementsByAlias* again.

View may be empty. If view is not specified, the module will be determined from each element alias and the current user has to be a workspace administrator. If the user is a regular user, a view that the user has access to must be specified.

Returns

getElementsByAlias returns ElementsResult. See the section Schema guide below.

Parameters

Name	Description	Optional	Type
elementAliases	a list of aliases for elements that should be retrieved	no	XML Schema string
view	alias for a view	yes	XML Schema string
token	authentication token	no	XML Schema string

Faults

Name	Returned when
InvalidArgumentFault	there are more than 100 aliases in the elementAliases list
NotAuthenticatedFault	no session is active for the supplied token

Examples

Example set up: two workspaces, one with alias *workspace1* and the other with *workspace2*. In both workspace1 and workspace2 there are modules with alias *module1* and views based on module1 with alias *view1*. In module1 in workspace1 there are two elements, *element1* (which is a part of view1) and *element2* (which isn't part of view1). In

module1 in workspace2 there are two elements, *element3* (which is a part of view1) and *element4* (which isn't part of view1).

Scenario 1: A workspace administrator (with access to workspace1 and workspace2) calls getElementsByAlias with elementAliases set to [element1, element2, element3, element4] and view set to nil/null. Result: all 4 elements are returned.

Scenario 2: A regular user (with access to workspace1 and workspace2 and view1 in both workspaces) calls *getElementsByAlias* with elementAliases set to [element1, element2, element3, element4] and View set to view1. Result: element1 and element3 are returned, since only these are part of view1. The ElementsResult will include Errors for the other elements.

Scenario 3: A workspace administrator (with access to workspace1 and workspace2) calls *getElementsByAlias* with elementAliases set to [element1, element2, element3, element4] and View set to view1. Result: element1 and element3 are returned since only these are part of view1. The ElementsResult will include Errors for the other elements.

Scenario 4: A workspace administrator (with access to workspace1) calls *getElementsByAlias* with elementAliases set to [element1, element2, element3, element4] and View set to nil/null. Result: element1 and element2 are returned since the user does not have access to workspace2. The ElementsResult will include Errors for the other elements.

getAttributeDefinitions

Retrieves a list of attribute definitions for a specified domain in a workspace.

If both workspace and domain are empty, the default Add Domain will be assumed. The default Add Domain can be configured in *Application > Data Access*.

Returns

getAttributeDefinitions returns AttributeDefinitionsResult.

Property name	Description	Type
attributeDefinitions	A list of AttributeDefinitions.	AttributeDefinition
errors	A list of Errors, if there were any errors or warnings when retrieving AttributeDefinitions.	Error

Parameters

Name	Description	Optional
workspace	alias for a workspace	may be empty if domain also is empty
domain	alias for a domain	may be empty if workspace also

		is empty
token	authentication token	no

Faults

Name	Returned when
AccessDeniedFault	the user does not have access to the workspace or the domain
DomainNotFoundFault	the specified domain cannot be found
InvalidArgumentFault	either workspace or domain is nil/null, empty or left out.
NotAuthenticatedFault	no session is active for the supplied token
WorkspaceNotFoundFault	the specified workspace cannot be found

Examples

Example set up: a workspace with alias *workspace1* including a module with alias *module1* and a view with alias *view1* which is an add view. The Add Domain is set to view1 in workspace1.

Scenario 1: A workspace administrator calls *getAttributeDefinitions* with Workspace set to workspace1 and Domain set to module1. Result: the AttributeDefinitions for module1 in workspace1 are returned.

Scenario 2: A regular user calls *getAttributeDefinitions* with Workspace set to workspace1 and Domain set to module1. Result: an AccessDeniedFault is returned, since regular users do not have access to modules.

Scenario 3: A regular user (with access to view1 in workspace1) calls *getAttributeDefinitions* with Workspace set to nil/null and and Domain set to nil/null. Result: the AttributeDefinitions for view1 in workspace1 are returned.

Scenario 4: A user calls *getAttributeDefinitions* with Workspace set to nil/null and Domain set to view1. Result: an InvalidArgumentFault is returned since no workspace was specified.

Scenario 5: A user calls *getAttributeDefinitions* with Workspace set to workspace2 and Domain set to view1. Result: a WorkspaceNotFoundFault is returned, since there is no workspace2 (in this example).

getFileContent

Retrieves the file data contents of a file attribute. This is the only way to retrieve actual file data from a file attribute, since file attributes only include meta data about the files.

If the view is nil/null, empty or left out, the element's module is assumed.

Returns

Binary data in XML Schema base64binary format.

Parameters

Name	Description	Optional	Type
element	alias for an element	no	XML Schema string
attribute	alias for a file attribute	no	XML Schema string
view	alias for a view	yes	XML Schema string
fileNumber	the number of the file within the file attribute. Use 0 if the file attribute only can contain one file	no	XML Schema int
token	authentication token	no	XML Schema string

Faults

Name	Returned when
AccessDeniedFault	the user does not have access to the workspace, domain, element or attribute
AttributeNotFoundFault	the supplied attribute cannot be found, the attribute is not a file attribute, or the fileNumber cannot be found in the file attribute
DomainNotFoundFault	the supplied view cannot be found
ElementNotFoundFault	the supplied element cannot be found
InvalidArgumentFault	element is empty, attribute is empty, or fileNumber is not a positive integer
NotAuthenticatedFault	no session is active for the supplied token

Examples

Example set up: a module with alias *module1* and two views, the first with alias *view1* and the other with alias *view2*. In module1 there is an element with alias *element1*. The module has a file attribute with alias *fileAttribute* and element1 has a fileAttribute with two files with numbers 0 and 1. The element is part of both view1 and view2. The fileAttribute is visible in view1 but not in view2.

Scenario 1: A workspace administrator calls *getFileContent* with Element set to element1, Attribute set to fileAttribute, View set to nil/null and fileNumber set to 0. Result: the data for file number 0 in fileAttribute of element1 is returned.

Scenario 2: A regular user calls *getFileContent* with Element set to element1, Attribute set to fileAttribute, View set to view1 and fileNumber set to 0. Result: the data for file number 0 in fileAttribute of element1 is returned.

Scenario 3: A regular user calls *getFileContent* with Element set to element1, Attribute set to fileAttribute, View set to view2 and fileNumber set to 0. Result: an AccessDeniedFault is returned since fileAttribute isn't visible in view2.

Scenario 4: A regular user calls *getFileContent* with Element set to element1, Attribute set to fileAttribute, View set to view1 and fileNumber set to 3. Result: an AttributeNotFoundFault is returned since there is no file number 3 in fileAttribute.

getModules

Retrieves a list of modules in a workspace. The user must be a workspace administrator in the specified workspace.

Returns

getModules returns a list of *Domains*. See the section *Schema guide* below.

Parameters

Name	Description	Optional	Type
workspace	alias for a workspace	no	XML Schema string
token	authentication token	no	XML Schema string

Faults

Name	Returned when
AccessDeniedFault	the user is not a workspace administrator
InvalidArgumentFault	workspace is nil/null, empty or left out
NotAuthenticatedFault	no session is active for the supplied token
WorkspaceNotFoundFault	the supplied workspace cannot be found

Examples

Example set up: A workspace with alias *workspace1*.

Scenario 1: A workspace administrator calls *getModules* with Workspace set to workspace1. Result: all modules in workspace1 are returned.

Scenario 2: A regular user calls *getModules* with Workspace set to workspace1. Result: an

AccessDeniedFault is returned.

getViews

Retrieves a list of views that the current user has access to.

Workspace and/or module may be specified. If workspace is specified, only views for that workspace will be retrieved. If workspace is empty, all workspaces the current user has access to will be considered. If module is specified only views for that module will be retrieved. Both module and workspace may not be empty.

Returns

getViews returns a list of *Domains*. See the section *Schema guide* below.

Parameters

Name	Description	Optional	Type
workspace	alias for a workspace	if module has a value	XML Schema string
module	alias for a module	if workspace has a value	XML Schema string
token	authentication token	no session is active for the supplied token	XML Schema string

Faults

Name	Returned when
AccessDeniedFault	the user does not have access to workspace
InvalidArgumentFault	both workspace and module is nil/null, empty or left out
NotAuthenticatedFault	no session is active for this token
WorkspaceNotFoundFault	the supplied workspace cannot be found

Examples

Example set up: two workspaces, with aliases *workspace1* and *workspace2*. Each workspace has two modules with alias *module1* and *module2* respectively. Each workspace also has two views, *view1* based on module1 and *view2* based on module2.

Scenario 1: A regular user (with access to all views in the set up) calls *getViews* with Workspace set to workspace1 and Module set to module1. Result: view1 for workspace1 is returned.

Scenario 2: A regular user (with access to all views in the set up) calls *getViews* with Workspace set to nil/null and Module set to module1. Result: view1 for workspace1 and

view1 for workspace2 are returned.

Scenario 3: A regular user (with access to all views in the set up) calls *getViews* with Workspace set to workspace1 and Module set to nil/null. Result: view1 and view2 for workspace1 are returned.

Scenario 4: A regular user calls *getViews* with workspace set to nil/null and view set to nil/null. Result: an InvalidArgumentFault is returned.

getWorkspaces

Retrieves a list of workspaces that the current user has access to.

Returns

A list of *Workspaces*:

Property name	Description	Type
alias	The alias for this workspace	XML Schema string
name	The display name for this workspace	XML Schema string
settings	A list of name/value pairs that contain various settings for workspaces. Currently not used.	Setting

Parameters

Name	Description	Optional	Type
token	authentication token	no	XML Schema string

Faults

Name	Returned when
NotAuthenticatedFault	no session is active for this token

getExternalLink

Retrieves a URL to an element. The URL can be entered by a user in a web browser in order to display the element.

Note: *getExternalLink* is released in version 6.3 of Focal Point.

Returns

The URL as an XML Schema string.

Parameters

Name	Description	Optional
element	alias for the element	no
token	authentication token	no

Faults

Name	Returned when
ElementNotFoundFault	the supplied element cannot be found
InvalidArgumentFault	element is nil/null, empty or left out
NotAuthenticatedFault	no session is active for this token

Change history

getChangedElements

Retrieves aliases for elements that have changed within a specified time range. An element is considered changed if at least one attribute that logs history has changed.

If both workspace and domain are specified, only elements matching that combination will be included in the result. If workspace is not specified, elements for the domain alias in all workspaces that the current user has access to will be included in the result.

Only the aliases for elements are returned. To retrieve the elements, including the current attribute values, use *getElementsByAlias*.

Returns

A ChangedElementsResult

Property name	Description	Type
changedDomains	A list of ChangedDomains	ChangedDomain
errors	A list of Errors that occured when retrieving changedElements	Error

ChangedDomain

Property name	Description	Type
module	The alias of the module where the changes occurred.	XML Schema string

view	The alias of the view where the changes occurred. May be empty, left out or nil/null if the changes were retrieved from a module.	XML Schema string
errors	A list of Errors that occurred when retrieving changes from this domain.	Error
workspace	The alias for the workspace where the changes occurred.	XML Schema string
elements	A list of aliases for elements that have changed.	XML Schema string

Parameters

Name	Description	Optional	Type
workspace	alias for a workspace	yes	XML Schema string
domain	alias for a domain	no	XML Schema string
start	the start time of the range for which history should be retrieved	no	XML Schema dateTime
end	the end time of the range for which history should be retrieved	no	XML Schema dateTime
token	authentication token	no	XML Schema string

Faults

Name	Returned when
AccessDeniedFault	the user does not have access to the workspace
DomainNotFoundFault	no matching domains found
InvalidArgumentFault	domain, start or end is left out, empty or nil/null, or end is before start
NotAuthenticatedFault	no session is active for the supplied token
WorkspaceNotFoundFault	the workspace cannot be found

Examples

Example set up: two workspaces, with alias *workspace1* and *workspace2* respectively, where each workspace has two modules with aliases *module1* and *module2*. In workspace1, module1, there is an element *element1* that was changed on 1 September 2008, in workspace1 module2 there is an element *element2* that was changed on 2 September 2008, in workspace2 module1 there is an element *element3* that was changed

on 3 September 2008, in workspace2 module2 there is an element *element4* that was changed on 4 September 2008.

Scenario 1: A workspace administrator calls *getChangedElements* with Workspace set to nil/null, Domain set to module1, Start set to 2 September 2008 and End set to 4 September 2008. Result: the alias for element3 is returned.

Scenario 2: A workspace administrator calls *getChangedElements* with Workspace set to workspace1, Domain set to module2, Start set to 2 September 2008 and End set to 4 September 2008. Result: the alias for element2 is returned.

getHistory

Retrieves the change history for a domain.

If both workspace and domain are specified, history for that combination will be retrieved. If workspace is not specified, history for the domain alias in all workspaces that the current user has access to will be retrieved.

The HistoryResult will include the found history entries, sorted in date order, starting with the newest. The history is limited to 100 entries for each workspace. If the limit has been reached, the HistoryResult will have the property hasMore set to true, and have a continueDate. If more history items should be retrieved, *getHistory* may be called again using continueDate as *end* (and *start* same as before).

Returns

getHistory returns a list of *HistoryResults*, one for each workspace. See the section *Schema guide* below.

Parameters

Name	Description	Optional	Type
workspace	alias for a workspace	yes	XML Schema string
domain	alias for a domain	no	XML Schema string
start	the start time of the range for which history should be retrieved	no	XML Schema dateTime
end	the end time of the range for which history should be retrieved	no	XML Schema dateTime
token	authentication token	no	XML Schema string

Faults

Name	Returned when

AccessDeniedFault	the user does not have access to the specified workspace
DomainNotFoundFault	the specified domain cannot be found in any workspace
InvalidArgumentFault	domain, start or end is left out, empty or nil/null or end is before start
NotAuthenticatedFault	no session is active for the supplied token
WorkspaceNotFoundFault	the workspace cannot be found

getElementHistory

Retrieves the change history for an element.

If view is specified, history for the element in that view will be retrieved. Otherwise history for the element in the module will be retrieved.

The HistoryResult will include the found history entries, sorted in date order, newest first. The history is limited to 100 entries. If the limit has been reached, the HistoryResult will have the property hasMore set to true, and have a continueDate. If more history items should be retrieved, *getElementHistory* may be called again using continueDate as *end* (and *start* same as before).

Returns

getElementHistory is returned by HistoryResult. See the section Schema guide below.

Parameters

Name	Description	Optional	Type
element	alias for an element	no	XML Schema string
view	alias for a view	yes	XML Schema string
start	the start time of the range for which history should be retrieved	no	XML Schema dateTime
end	the end time of the range for which history should be retrieved	no	XML Schema dateTime
token	authentication token	no	XML Schema string

Faults

Name	Returned when
AccessDeniedFault	the user does not have access to the element or the view

DomainNotFoundFault	the view cannot be found
ElementNotFoundFault	the element cannot be found
InvalidArgumentFault	element, start or end is left out, empty or nil/null or end is before start
NotAuthenticatedFault	no session is active for the supplied token

getAttributeHistory

Retrieves the change history for an attribute of an element.

If view is specified, history for the attribute of the element in that view will be retrieved. Otherwise history for the element in the module will be retrieved.

The HistoryResult will include the found history entries, sorted in date order, newest first. The history is limited to 100 entries. If the limit has been reached, the HistoryResult will have the property hasMore set to true, and have a continueDate. If more history items should be retrieved, *getElementHistory* may be called again using continueDate as *end* (and *start* same as before).

Returns

getAttributeHistory returns a HistoryResult. See the section Schema guide below.

Parameters

Name	Description	Optional	Type
attribute	alias for an attribute	no	XML Schema string
element	alias for an element	no	XML Schema string
view	alias for a view	yes	XML Schema string
start	the start time of the range for which history should be retrieved	no	XML Schema dateTime
end	the end time of the range for which history should be retrieved	no	XML Schema dateTime
token	authentication token	no	XML Schema string

Faults

Name	Returned when
AccessDeniedFault	the user doesn't have access to the element or the view
AttributeNotFoundFault	the attribute cannot be found

DomainNotFoundFault	the view cannot be found
ElementNotFoundFault	the element cannot be found
InvalidArgumentFault	attribute, element, start or end is left out, empty or nil/null or end is before start
NotAuthenticatedFault	no session is active for the supplied token

Adding and updating

addElement

Creates a new element.

Workspace and domain alias may be specified. The domain may be a module (in which case the user has to be a workspace administrator) or view. Note that only add views can be used when adding an element. If both workspace and domain is empty or null, the default Add Domain will be used.

It is possible to supply an element alias. If specified, the element will get this alias when created. If no alias has been supplied, the element will be created with a random alias.

A list of attribute changes may be included, and the new element will be created with the supplied values. The aliases and attribute types must match the aliases and types in the module where the element is added. Attributes not included will be assigned their default values.

Returns

addElement returns ElementAddResult. See the section Schema guide below.

Parameters

Name	Description	Optional	Type
workspace	alias for the workspace where the element should be added	may be empty if domain also is empty	XML Schema string
domain	alias for the domain where the element should be added, must be a module or an add view	may be empty if workspace also is empty	XML Schema string
element	alias for the element to be created. If specified, the element will get this alias when created. If empty the element will get a new alias when created	yes	XML Schema string

attributes	a list of attribute changes that will be assigned to the new element. Attributes that should have the default value shall not be included.	yes	AttributeChange
token	authentication token	no	XML Schema string

Faults

Name	Returned when
AccessDeniedFault	the user does not have access to the workspace or domain
DomainNotFoundFault	the domain cannot be found or, if the domain is a view, its module cannot be found
ElementNotCreatedFault	element could not be created
ElementAlreadyExistsFault	an element with the specified alias already exists
InvalidArgumentFault	either domain or workspace parameter is nil/null, empty or left out
NotAuthenticatedFault	no session is active for the supplied token
WorkspaceNotFoundFault	the workspace cannot be found

addElements

Creates several elements in one batch.

A maximum of 100 elements can be created for each call. If more than 100 elements should be added, *addElements* need to be called several times.

Returns

Returns a list of *ElementAddResults*. Each ElementAddResult holds the alias of the created element and a list of errors that may have occurred during the creation.

Parameters

Name	Description	Optional	Type
elements	A list of NewElements	no	NewElement
token	authentication token	no	XML Schema string

NewElement:

Property name	Description	Type
workspace	Alias for the workspace where the element should be added, may be empty if domain also is empty	XML Schema string
domain	Alias for the domain where the element should be added, must be a module or an add view, may be empty if workspace also is empty	XML Schema string
element	Alias for the element to be created. If specified, the element will get this alias when created. If empty, the element will get a new alias when created	XML Schema string
attributeChanges	A list of attribute changes that will be assigned to the new element. Attributes that should have the default value shall not be included.	AttributeChange

Faults

Name	Returned when
InvalidArgumentFault	there are more than 100 supplied elements
NotAuthenticatedFault	no session is active for the supplied token

updateElement

Changes the specified attributes of an existing element.

If the user is a regular user, a view must be specified. If the user is a workspace administrator, specifying a view is optional. Since the update may include several attribute updates, some attributes may be saved and some may result in errors. Errors that occurred during the update will be returned.

Important:In the list of attributes, ensure to include only attributes that should be changed. As the values of the attribute list will replace the current value on the server, any changes made on the server after the value was read will be overwritten when calling *updateElement*.

For example: user 1 reads the title attribute of element1, then user 2 changes the title of element 1 and the user 1 calls *updateElement* including the old value for the title of element 1. In this case, the title of element 1 will be reverted to the previous value, since user 1 read the value before it was changed by another user.

Returns

A list of *Errors* that occurred during the update are returned. See the section *Schema guide* below.

Parameters

Name	Description	Optional
element	an alias for the element that will be updated	no
view	an alias for the view that will be used when updating	yes
attributes	a list of attribute changes that should be applied to the element	no
token	authentication token	no

Faults

Name	Returned when
AccessDeniedFault	the user does not have access to the view, module or workspace or the element is not part of the view
DomainNotFoundFault	the view cannot be found
ElementNotFoundFault	the element cannot be found
ElementUpdateFault	an error ocurred when updating the element
InvalidArgumentFault	element alias is nil/null, empty or left out
NotAuthenticatedFault	no session is active for the supplied token

updateElements

Updates several elements in one batch.

The specified attributes of each ElementUpdate will be updated. Element alias for the ElementUpdate must be specified. If the user is a regular user, a view must be specified. If the user is an administrator, specifying a view is optional. Since the update may include update of several elements and attributes, some elements and attributes may be saved and some may result in errors. Errors that occurred during the update will be returned.

A maximum of 100 elements can be updated for each call. If more than 100 elements should be updated, *updateElements* need to be called several times.

Important:In the list of attributes, ensure to include only attributes that should be change. As the values of the attribute list will replace the current value on the server, any changes made on the server after the value was read will be overwritten when calling

updateElements.

For example: user 1 reads the title attribute of element1, then user 2 changes the title of element 1, and the user 1 calls *updateElements*, including the old value for the title of element 1. In this case, the title of element 1 will be reverted to the previous value, since user 1 read the value before it was changed by another user.

Returns

A list of *ElementUpdateResults* is returned. If errors occurred during the element update, the result will include a set of errors for each affected element.

Property name	Description	Type
errors	A list of Errors.	Error
element	An alias for the element for which the errors occurred	XML Schema string

Parameters

Name	Description	Optional	Type
updates	a list of ElementUpdates	no	ElementUpdate
token	authentication token	no	XML Schema string

ElementUpdate:

Property name	Description	Type
attributeChanges	a list of attribute changes that should be applied to the element	AttributeChange
element	an alias for the element that will be updated	XML Schema string
view	an alias for the view that will be used when updating	XML Schema string

Faults

Name	Returned when
InvalidArgumentFault	there are more than 100 elements to be updated
NotAuthenticatedFault	no session is active for the supplied token

writeElement

Creates an element or updates it, if it already exists.

Element alias must be specified. If no element exists with that alias, a new element is created. If an element already exists, it will be be updated.

Workspace and domain may be specified. When creating an element, workspace and domain will be used to determine the module or add view that shall be used. If neither workspace nor domain is specified, the default Add Domain will be assumed.

When updating an element, workspace will be ignored (the workspace will be determined by the existing element) and domain will be used to determine which view to use for the update (if domain is empty the module will be assumed). Only workspace administrators have access to modules.

Returns

A list of *Errors* that occurred during the update are returned. See the section *Schema guide* below.

Parameters

Name	Description	Optional	Type
workspace	alias for the workspace where the element should be created	yes	XML Schema string
domain	alias for the domain used for creating or updating the element	yes	XML Schema string
element	alias for the element	no	XML Schema string
attributes	a list of attribute changes that will be applied.	yes	AttributeChange
token	authentication token	no	XML Schema string

Faults

Name	Returned when
AccessDeniedFault	the user does not have access to the domain or the element
DomainNotFoundFault	the specified domain does not exist
ElementUpdateFault	an error occurred during the update of the element

ElementNotCreatedFault	an element should have been, but was not created
InvalidArgumentFault	element is nil/null, empty or left out
NotAuthenticatedFault	no session is active for the supplied token
WorkspaceNotFoundFault	the supplied workspace cannot be found

writeElements

Writes several elements in one batch.

One element will be written for each supplied ElementChange.

Element aliases must be specified for each ElementChange. If no element exists with that alias, a new element is created. If an element already exists, it will be be updated.

Workspace and domain may be specified for each ElementChange. When creating an element, workspace and domain will be used to determine the module or add view that shall be used. If neither workspace nor domain is specified the default Add Domain will be assumed.

When updating elements, workspace will be ignored (the workspace will be determined by the existing element) and domain will be used to determine which view to use for the update (if domain is empty the module will be assumed). Only workspace administrators have access to modules.

A maximum of 100 elements can be written for each call. If more than 100 elements should be written, *writeElements* need to be called several times.

Returns

A list of *ElementWriteResults* is returned. If errors occurred during the write process, the result will include a set of errors for each affected element.

Property name	Description	Type
errors	A list of errors	Error
element	An alias for the element for which the errors occurred	XML Schema string

Parameters

Name	Description	Optional	Type
changes	A list of ElementChanges	no	ElementChange
token	authentication token	no	XML Schema string

ElementChange:

Property name	Description	Type
workspace	alias for the workspace where the element should be created	XML Schema string
domain	alias for the domain used for creating or updating the element	XML Schema string
element	alias for the element	XML Schema string
attributeChanges	a list of attribute changes that will be applied.	AttributeChange

Faults

Name	Returned when
InvalidArgumentFault	changes contain more than 100 elements
NotAuthenticatedFault	no session is active for the supplied token

Other

lookupWorkspaceAlias

Translates a workspace ID into an alias. This is included for compatibility with the previous Web Services API, which used internal identifiers instead of aliases. It can also be used if you know the internal identifier of a workspace, but not the alias.

Note: *lookupWorkspacesAlias* is released in version 6.3 of Focal Point.

Returns

The workspace alias as an XML Schema string.

Parameters

Name	Description	Optional	Type
workspaceId	An internal identifier for a workspace	no	XML Schema int

Faults

Name	Returned when
WorkspaceNotFoundFault	no workspace with the supplied identifier can be found

lookupViewAlias

Translates a view ID into an alias. This is included for compatibility with the previous Web Services API, which used internal identifiers instead of aliases. It can also be used if you know the internal identifier of a view, but not the alias.

Note: *lookupViewAlias* is released in version 6.3 of Focal Point.

Returns

The view alias as an XML Schema string.

Parameters

Name	Description	Optional	Type
viewId	An internal identifier for a view	no	XML Schema int
workspaceId	An internal identifier for a workspace	no	XML Schema int

Faults

Name	Returned when
DomainNotFoundFault	no view with the supplied identifier can be found

lookupElementAlias

Translates an element ID into an alias. This is included for compatibility with the previous Web Services API, which used internal identifiers instead of aliases. It can also be used if you know the internal identifier of an element, but not the alias.

Note: *lookupElementAlias* is released in version 6.3 of Focal Point.

Returns

The element alias as an XML Schema string.

Parameters

Name	Description	Optional	Type
elementId	An internal identifier for an element	no	XML Schema int
workspaceId	An internal identifier for a workspace	no	XML Schema int

Faults

Name	Returned when
ElementNotFoundFault	no element with the supplied identifier can be found

lookupAttributeAlias

Translates an attribute ID into an alias. This is included for compatibility with the previous Web Services API, which used internal identifiers instead of aliases. It can also be used if you know the internal identifier of an attribute, but not the alias.

Note: *lookupAttributeAlias* is released in version 6.3 of Focal Point.

Returns

The attribute alias as an XMLSchema string

Parameters

Name	Description	Optional	Type
attributeId	An internal identifier for an attribute	no	XML Schema int
workspaceId	An internal identifier for an attribute	no	

Faults

Name	Returned when
AttributeNotFoundFault	no attribute with the supplied identifier can be found

Schema guide

This section describes the data types in the XML Schema included in the WSDL. Also read the XML Schema to understand all the details.

ElementsResult

Property name	Description	Type
domains	A list of ElementSets, one for each domain.	ElementSet
errors	If there were any errors or warnings when accessing a domain.	Error

ElementSet

Property name	Description	Type
attributeDefinitions	A list of attribute definitions. They define, for example, the attributes and their names.	AttributeDefinition
elements	A list of elements	Element
module	The alias for the module from which this element set was retrieved.	XML Schema string
view	The alias for the view from which this element set was retrieved. May be nil/null, empty or left out if the element set was retrieved from a module.	XML Schema string
errors	A list of errors and warnings that occurred when reading elements or attributeDefinitions.	Error
workspace	The alias for the workspace from which this element set was retrieved.	XML Schema string

Element

An element has the following properties:

Property name	Description	Type
alias	The alias identifying this element.	XML Schema string
attributes	A list of attributes.	Attribute
module	The alias of the module that the element belongs to.	XML Schema string
view	The alias of view that the element was retrieved from. May be empty if the element was retrieved from a module.	XML Schema string
workspace	The alias for the workspace that the element belongs to.	XML Schema string
errors	A list of errors or warnings that occurred when reading any of the attribute values.	Error

Attribute

An attribute has the following properties

Property name	Description	Type
alias	The alias for the attribute in its module.	XML Schema string
type	The type of the attribute	AttributeType
writeable	True if the attribute value can be changed, false if it cannot.	XML Schema boolean
Exactly one of the following: checkBox, choice, date, float, file, integer, link, textList, linkList, matrix, multichoice, text, uniqueId, url, version or other	A representation of the attribute value. The property will depend on the attribute type.	CheckBox, Choice, Date, Float, File, Integer, Link, TextList, LinkList, Matrix, Multichoice, Text, UniqueId, Url, Version or XML Schema anyType.

The attribute value will be represented by exactly one property, but the name and type of that property will be different for each type of attribute in Focal Point. For example, a text attribute will have a *text* property, an integer attribute will have an *integer* property. The property called *other* is currently not used but may be used in future versions.

CheckBox

Property name	Description	Type
value	True if the checkbox is selected, false if not	XML Schema boolean

Choice

Property name	Description	Type
value	The selected choice item for this attribute. The value is the choice item name.	XML Schema string
availableValues	A list of choice items that this choice is allowed to be changed to.	XML Schema string

Date

Property name	Description	Type
dateTime	The value of a date expressed as a date and time in UTC. A date attribute is stored at the time	XML Schema dateTime

	00:00 (midnight) for the time zone of the server. If the server is in time zone GMT+2, the date 2008-08-13 will be represented as 2008-08-12T22:00. The *Last Changed Date* and *Created Date* attributes will include the exact time when the element was created or changed.	
expression	A date attribute may contain an expression. If it does not have an expression, this property will be nil/null.	Expression

Float

Property name	Description	Type
value	The float value	XML Schema double
expression	A float attribute may contain an expression. If it does not contain an expression, this property will be nil/null	Expression

File

Property name	Description	Type
fileInfos	A list of FileInfos, one for each file in the file attribute.	FileInfo

Note: the actual file content is not included in the file attribute, only meta data about the files in the file attribute. The file content can be retrieved by calling *getFileContent*.

FileInfo

Property name	Description	Type
fileName	The name of the file.	XML Schema string
contentType	The MIME type of the file.	XML Schema string
size	The size of the file (in bytes).	XML Schema long
fileNumber	Identifier for the file within a particular file attribute.	XML Schema int

Integer

Property name	Description	Type
value	The integer value.	XML Schema long
expression	An integer attribute may contain an expression. If it does not contain an expression, this property will be nil/null.	Expression

Link

Property name	Description	Type
displayText	Display text for a link.	XML Schema string
element	The alias for the linked element.	XML Schema string
module	The alias for the module of the linked element	XML Schema string
workspace	The alias for the workspace of the linked element	XML Schema string

Note: if a link is not set (has no value), all of these properties will be left out or be nil/null.

TextList

A TextList is the representation of the value of a list attribute of the type text.

Property name	Description	Type
textListEntries	A list of TextListEntries, one for each entry in the list	TextListEntry

TextListEntry

Property name	Description	Type
id	An identifier for the entry within its list.	XML Schema string
author	A link to the workspace member who created the entry.	Link
createdDate	The date and time when the entry was created.	XML Schema dateTime
lastChangedBy	A link to the workspace member who last edited this entry.	Link
lastChangedDate	The date and time when this entry was last edited.	XML Schema dateTime
formattedTextValue	The text value of the entry	XML Schema string

	formatted with a limited set of HTML rich text tags. The following tags are allowed: <B> for bold <I> for italics <U> for underlined <S> for strike-through <OL> for numbered list <UL> for bullet list	
textValue	The unformatted text value of the entry.	XML Schema string

LinkList

A LinkList is the representation of the value of a list attribute of the type link

Property name	Description	Type
links	A list of Links.	Link

Matrix

Property name	Description	Type
rows	A list of the rows that the matrix contains.	MatrixRow

MatrixRow

Property name	Description	Type
rowId	The ID for a row within a matrix. The column title row has the ID *ColumnTitle* and the next (first) row will have ID 1.	XML Schema string
cells	A list of the matrix cells in this row	MatrixCell

MatrixCell

Property name	Description	Type
rowId	The ID for the row where this cell is placed. The column title row has the ID *ColumnTitle* and the next (first) row will have ID 1, the next row will have ID 2, etc.	XML Schema string
columnId	The ID for the column where this cell is placed. The row title	XML Schema string

	column has ID *RowTitle*, the next (first) column has ID *A*, the next column ID *B*, etc.	
value	The value of the cell as an unformatted text.	XML Schema string
formattedText	The text value of the cell formatted with a limited set of HTML rich text tags. The following tags are allowed: <B> for bold <I> for italics <U> for underlined <S> for strike-through <OL> for numbered list <UL> for bullet list	XML Schema string
expression	A matrix cell may contain an expression. If it does not have an expression, this property will be nil/null.	Expression
locked	If true, the cell cannot be edited.	XML Schema boolean

Multichoice

Property name	Description	Type
selected	A list of the selected items of the multichoice attribute. The value of each item is the item's name.	XML Schema string

Text

Property name	Description	Type
formattedTextValue	The value of the text attribute formatted with a limited set of HTML rich text tags. The following tags are allowed: <B> for bold <I> for italics <U> for underlined <S> for strike-through <OL> for numbered list <UL> for bullet list	XML Schema string
textValue	The unformatted value of the text attribute	XML Schema string
expression	A text attribute may contain an expression. If it does not contain an expression, this	Expression

	property will be nil/null.	

UniqueId

Property name	Description	Type
value	The value of a unique ID attribute.	XML Schema string

Url

Property name	Description	Type
value	The value of a URL attribute	XML Schema string

Version

Property name	Description	Type
major	The major part of a version, that is, the x part of x.y.z	XML Schema string
minor	The minor part of a version, that is, the y part of x.y.z.	XML Schema string
patch	The patch part of a version, that is, the z part of x.y.z.	XML Schema string

Other

This type may contain any XML. It is currently not used, but may be used for future expansion or attribute values that cannot be represented in any other way.

Expression

Text, integer, float and date attributes may contain an expression. If they contain an expression, the value of the attribute will contain the result of the expression, unless the expression is pending update or is invalid.

Property name	Description	Type
expression	The actual expression. It is represented without the initial equals symbol (=) used in the user interface.	XML Schema string
pendingUpdate	If set to true, the expression is waiting to be evaluated.	XML Schema boolean
valid	If set to false, there is an error in the expression, for example due to incorrect expression syntax.	XML Schema boolean

AttributeDefinition

Property name	Description	Type
alias	The alias for this attribute	XML Schema string
availableValues	A list of values valid for this attribute. Currently only used for choice and multichoice attributes.	XML Schema string
description	The description of the attribute.	XML Schema string
mandatory	Whether this attribute must contain a value or not.	XML Schema boolean
module	The alias of the module this attribute belongs to.	XML Schema string
name	The name of the attribute.	XML Schema string
settings	A list name/value pairs that contain various settings for attributes. Currently not used.	Setting
sortOrder	The order for this attribute relative to other attributes. Used when displaying element to ensure that attributes are sorted correctly.	XML Schema int
systemName	Some attribute have a systemName which gives the attribute a special meaning. For example the systemName “Title” means that the attribute is the title attribute. See table below for a list of some systemNames	XML Schema string
type	The type of attribute.	AttributeType
view	The alias of the view that this attribute was retrieved from. May be empty, nil/null or left out if the attribute was retrieved from a module.	XML Schema string
workspace	The alias of the workspace this attribute belongs to	XML Schema string
writeable	True if the value of the attribute can be edited, false if not. Note that a specific attribute for an element may not be editable for other reasons (for example if the element is locked) see the writeable property of *Attribute*.	XML Schema boolean

mirror	True if this is a mirror attribute, reflecting the value of another attribute. If so, the type of this attribute will be the same as the mirrored attribute.	XML Schema boolean

Here is a list of a few common systemNames and what they mean:

SystemName	Description
Title	The Title attribute.
Descr	The Description attribute.
createdby	The Creator attribute.
createddate	The Created Date attribute.
lastchangedby	The Last Changed By attribute.
lastchangedate	The Last Changed Date attribute.
isGroup	This is a checkbox attribute that tells whether an element is a folder or not. An element can be turned into a folder (or vice versa) by changing the value of this attribute.
parent	The Parent attribute tells in which folder an element can be found.
Prefix	The Prefix attribute. Which attribute it is can be configured by a user.
lock	A lock attribute is a checkbox attribute that can be added to module. If set to true, the element (or some attributes of the element) is locked for editing.
change_assigned	This is an invisible checkbox attribute, only used when integration with Telelogic® Change™ is enabled.

Error

Property name	Description	Type
code	The error code, indicating which kind of error this is.	ErrorCode
message	A human readable description of the error	XML Schema string
alias	An alias associated with error. The kind of alias depends on the errror code. For example, if the error code is *attributeReadError,* the alias will be an attribute alias	XML Schema string

	indicating for which attribute the error occurred. May be nil/null, empty or left out.

ErrorCode

ErrorCode defines a set of errors that may of occur (see Error). The code may be one of the following:

Error code	Description
attributeNotEditable	Tried to change the value of an attribute that cannot be edited.
attributeNotPartOfDomain	Tried to read or change an attribute that was not part of a previously specified domain.
attributeNotFound	An alias was given for an attribute that does not exist.
attributeWriteError	An error occurred when changing the value of an attribute.
attributeReadError	An error occurred when reading the value of an attribute.
emptyDomain	A domain with no elements was specified.
domainNotFound	An alias was given for a domain that does not exist.
domainNoAccess	Tried to access a domain that the current user does not have access to.
domainError	An error occurred when reading a domain
elementWriteError	An error occurred when creating or updating an element.
elementNotFound	An alias was given for an element that does not exist.
elementNotCreated	An error occurred when creating an element and the element was never created.
elementNoAccess	Tried to access an element that the current user does not have access to.
elementAlreadyExists	Tried to create an element with a specified alias but an element with that alias already exists.
invalidArgument	An input parameter had a value (or no value) that was not allowed.
accessDenied	Tried to access something that the current user doe not have access to.
workspaceNotFound	An alias was given for a workspace that does not exist.
workspaceNoAccess	Tried to access a workspace that the current user does not have access to.

expressionError	An error related to expressions occurred, for example trying to add an expression with incorrect syntax.
internalExpressionError	An internal (systems) expressions error occurred. If this occurs, the Focal Point server is not functioning properly.
generalError	An unspecified error occurred. The message property might include details.
internalError	An unspecified internal error occurred. If this occurs, the Focal Point server is not functioning properly, for example due to a inconsistent database.

AttributeType

AttributeType may be one of the following: CheckBox, Choice, Date, File, Float, Heading, Integer, Link, TextList, LinkList, Matrix, Multichoice, Text, Url, UniqueId, Version or Other.

Heading attributes will never have a value. They are included only as a separator between attributes.

The type Other is currently not used. It is included for future expansion of the attribute types.

There is no Mirror attribute type. Mirror attributes will instead be of the attribute type of the mirrored attribute, and the property *mirror* of *AttributeDefinition* will be true.

Domain

A domain is representation of the meta data for a module or a view.

Property name	Description	Type
workspace	The alias of the workspace that the domain belongs to.	XML Schema string
alias	The alias for this domain.	XML Schema string
name	The display name of this domain	XML Schema string
type	“View” or “Module”	XML Schema string
module	The module this domain is based on, if the domain is a view. If the domain is a module, this will have the same value as *alias*.	XML Schema string
addable	True if elements can be added to this domain.	XML Schema boolean

displayable	True if this domain is used for reading elements	XML Schema boolean
settings	A list name/value pairs that contain various settings for domains. Currently not used.	Setting

HistoryResult

Property name	Description	Type
historyEntries	A list of HistoryEntries for this result.	HistoryEntry
errors	A list of Errors that occurred when retrieving history	Error
hasMore	This will be true if the limit for number of HistoryEntries has been hit, but there are more HistoryEntries to retrieve.	XML Schema boolean
continueDate	If hasMore is true, there will be a continueDate. This date can be used to retrieve the remaining HistoryEntries.	XML Schema dateTime

HistoryEntry

A HistoryEntry represents one specific change for an attribute in an element, at a given time.

Property name	Description	Type
username	The name of the user who made the change.	XML Schema string
userAlias	The alias of the user who made the change.	XML Schema string
date	The date and time when the change occurred.	XML Schema dateTime
workspace	The alias of the workspace this HistoryEntry comes from.	XML Schema string
module	The alias of the module this HistoryEntry comes from.	XML Schema string
view	The alias of the view this HistoryEntry comes from. May be nil/null, empty or left out if the HistoryEntry was not retrieved from a view.	XML Schema string
element	The alias of the element	XML Schema string
attribute	The value the attribute had at	Attribute

	this particular time.	
errors	A list of errors that occurred when retrieving the HistoryEntry	Error

ElementAddResult

Property name	Description	Type
element	The alias of the created element	XML Schema string
errors	A list of Errors that occurred when adding the element.	Error

AttributeChange

Property name	Description	Type
alias	The alias of the attribute	XML Schema string
Exactly one of the following: checkBox, choice, date, float, file, integer, link, textList, linkList, matrix, multichoice, text, uniqueId, url, version or other	A representation of the attribute value. The property depends on the attribute type.	CheckBox, ChoiceChange, DateChange, FloatChange, FileChange, IntegerChange, LinkChange, TextListChange, LinkListChange, MatrixChange, Multichoice, TextChange, UniqueId, Url, VersionChange or XML Schema anyType.

ChoiceChange

Property name	Description	Type
value	The value (name of the choice item) the choice attribute should be changed to.	XML Schema string

DateChange

The DateChange shall have exactly one of the following:

Property name	Description	Type
value	A new date value. The date should be within the the range of the date for the servers time zone. If the server is in time zone GMT+2, the range 2008-08-12T22:00 to 2008-08-13T22:00 will be interpreted as the date 2008-08-	XML Schema dateTime

	13.	
expression	A new expression for the date attribute. See the Focal Point help for instructions on how to write expressions. The initial equals symbol (=) may be omitted.	XML Schema string

FloatChange

The FloatChange shall have exactly one of the following:

Property name	Description	Type
value	A new float value.	XML Schema double
expression	A new expression for the float attribute. See the Focal Point help for instructions how to write expressions. The inital equals symbol (=) may be omitted.	XML Schema string

FileChange

Property name	Description	Type
removeFiles	A list of files that will be removed from the attribute.	RemoveFile
addFiles	A list of files (including binary content) that will be added to the attribute.	AddFile

RemoveFile

Property name	Description	Type
fileNumber	The number of the file that will be removed.	XML Schema int

AddFile

Property name	Description	Type
fileName	The name of the file	XML Schema string
contentType	The MIME type of the file.	XML Schema string
fileData	The content of the file	XML Schema base64Binary

IntegerChange

An IntegerChange shall have exactly one of the following:

Property name	Description	Type
value	A new integer value	XML Schema long
expression	A new expression for the integer attribute. See the Focal Point help for instructions on how to write expressions. The initial equals symbol (=) may be omitted.	XML Schema string

LinkChange

Property name	Description	Type
element	Alias of an element that the link attribute will link to	XML Schema string

TextListChange

Property name	Description	Type
deleteEntries	A list of DeleteTextListEntries	DeleteTextListEntry
changeEntries	A list of TextListEntryChanges	TextListEntryChange
addEntries	A list of AddTextListEntries	AddTextListEntry

DeleteTextListEntry

Property name	Description	Type
id	Identifier of an entry in a text list attribute that will be deleted	XML Schema string

ChangeEntries

Property name	Description	Type
id		
textValue or formattedTextValue	Either a plain text value, or a formattedTextValue. Text is formatted using a quoted limited subset of HTML.	XML Schema string

AddTextListEntry

Property name	Descsription	Type
textValue or formattedTextValue	Either a plain text value or a formattedTextValue. Text is formatted using a quoted limited subset of HTML.	XML Schema string

LinkListChange

Property name	Description	Type
id	Identifier of an an entry in a text list attribute that will be changed	XML Schema string
textValue or formattedTextValue	Either a plain text value, or a formattedTextValue. Text is formatted using a quoted limited subset of HTML.	XML Schema string

MatrixChange

A MatrixChange shall have exactly one of the following:

Property name	Description	Type
resetMatrix	Will delete an entire matrix and replace it with new values.	ResetMatrix
changeCells	Changes that will be applied to the specified cells.	ChangeCells
addRows	Rows that will be added to the matrix	AddRows
addColumns	Columns that will be added to the matrix	AddColumns
deleteRow	The ID of a row that will be deleted. The column title row has the ID *ColumnTitle* and the next (first) row will have ID 1, followed by 2, etc.	XML Schema string
deleteColumn	The identifier of a column that will be deleted. The row title column has ID *RowTitle*, the next (first) column has ID *A* followed by *B*, etc.	XML Schema string

ResetMatrix

Property name	Description	Type
newRows	A list of NewCells that will be	NewCells

	added to the matrix. There should be one set of NewCells for each row.	

ChangeCells

Property name	Description	Type
cellChanges	A list of MatrixCellChanges that will be applied to a matrix attribute. New cells will not be created, only existing cells may be changed.	MatrixCellChange

AddRows

Property name	Description	Type
rows	A list if NewCells that will be added to the matrix. There should be one set of NewCells for each new row.	NewCells

AddColumns

Property name	Description	Type
columns	A list if NewCells that will be added to the matrix. There should be one set of NewCells for each new column.	NewCells

NewCells

Property name	Description	Type
newCells	A list of MatrixCellValueChanges. Each one representes a cell.	MatrixCellValueChange
x	Placeholder only. Will be ignored.	N/A

MatrixCellChange

Property name	Description	Type
rowId	An ID for the row where the cell is found. The column title row has the ID *ColumnTitle* and the next (first) row will have ID 1, the following 2, etc.	XML Schema string

columnId	An ID for the column where the cell is found. The row title column has ID *RowTitle*, the next (first) column has ID *A* , the following *B*, etc.	XML Schema string
valueChange	The new value for the cell	MatrixCellValueChange

MatrixCellValueChange

A MatrixCellValueChange shall have exactly one of the following:

Property name	Description	Type
value	A new value (text or number) for a matrix cell.	XML Schema string
formattedText	A formatted text value for a matrix cell. Text is formatted using a quoted limited subset of HTML.	XML Schema string
expression	A new expression for the matrix cell. See the Focal Point help for instructions on how to write expressions. The initial equals symbol (=) may be omitted.	XML Schema string

TextChange

A TextChange shall have exactly one of the following:

Property name	Description	Type
textValue	New value as unformatted text.	XML Schema string
formattedText	A new formatted text value. Text is formatted using a quoted limited subset of HTML.	XML Schema string
expression	A new expression for the text attribute. See the Focal Point help for instructions on how to write expressions. The initial equals symbol (=) may be omitted.	XML Schema string

VersionChange

Property name	Description	Type
nextMajorVersion	If true, a version attribute should be increased to the next major version. If false, nothing will happen.	XML Schema boolean

More information

Contacting IBM Rational Software Support

Support and information for Telelogic products is currently being transitioned from the Telelogic Support site to the IBM Rational Software Support site.
During this transition phase, your product support location depends on your customer history.

Product support

• If you are a heritage customer, meaning you were a Telelogic customer prior to November 1, 2008, please visit the Focal Point Support Web site.
Telelogic customers will be redirected automatically to the IBM Rational Software Support site after the product information has been migrated.
• If you are a new Rational customer, meaning you did not have Telelogic-licensed products prior to November 1, 2008, please visit the IBM Rational Software Support site.
Before you contact Support, gather the background information that you will need to describe your problem. When describing a problem to an IBM software support specialist, be as specific as possible and include all relevant background information so that the specialist can help you solve the problem efficiently. To save time, know the answers to these questions:
• What software versions were you running when the problem occurred?
• Do you have logs, traces, or messages that are related to the problem?
• Can you reproduce the problem? If so, what steps do you take to reproduce it?
• Is there a workaround for the problem? If so, be prepared to describe the workaround.

Other information

For Rational software product news, events, and other information, visit the IBM Rational Software Web site.

Trademarks