Focal Point®

REST XML API Reference Manual

Release 7.5.1

Publication information

FPNR-7510-00 (May 2022)

Information in this publication is subject to change. Changes will be published in new editions or technical newsletters.

Copyright notice

Focal Point®

Disclaimer

Trademarks

Contents

1 REST XML API reference

The Representational State Transfer (REST) API for Focal Point is a RESTful interface through which you can access, create and update Focal Point resources.

To understand and use the RESTful API, you must be familiar with the REST principles and with HTTP, XML, and the XML Schema.

The RESTful API for Focal Point is a provisional API. It is not for production use and can change at any time.

You can refer to the REST API examples to learn how to use the APIs for Focal Point. The examples are in Java source files that can be compiled and run. The RESTful API examples are available in `*fpinstall*\apiexamples\RESTAPIExample.zip`. An administrator (who has access to the application server on which Focal Point is installed) can provide you with the example package. For more information about the source files and the process to compile the code, see the readme file in the example package.

Setup

The server name part of the resource URI can be configured in the Login or Balancer URL page.

To specify the server name for the resource URI, in Focal Point, from the User menu, select Administration. On the Administration page, click **Application > Login Page**; under **Login Page Settings**, click the **Edit** icon for **Login or Balancer URL**, specify the URL, and click the **Save** icon.

Note Make sure that the host name or the server name does not change. A change in the host name can lead to broken links in Focal Point integrated systems that link to the Focal Point resources.

Authentication

The requests to the RESTful API must be authenticated by using HTTP basic authentication. Unless you use HTTPS authentication, the user name and password are sent without encryption.

In HTTP basic authentication, character encoding is not specified for user names and passwords. User names and passwords can include ASCII characters only. You might be able to use ISO-8859-1 characters if the characters are encoded by the client correctly.

Character encoding

The XML information that is retrieved from Focal Point uses UTF-8 character encoding.

For XML that is sent to Focal Point, the charset of the Content-Type HTTP header is used, if present. Else, the encoding that is specified in the XML prolog is used. If no prolog is present, the default for the XML is used. The encoding that is used must match any declared encoding. You can use UTF-8 encoding when you communicate with Focal Point and specify UTF-8 encoding in the HTTP header and XML prolog.

Accessing REST XML APIs in a multi threaded environment

From Focal Point version 6.5.2.3, to access REST API in a multi threaded environment from the client applications, you must follow the these steps:

1. Use `MultiThreadedHttpConnectionManager` object for creating the HTTP connection object. For example:

```
...
MultiThreadedHttpConnectionManager connectionManager = new
MultiThreadedHttpConnectionManager();
client = new HttpClient(connectionManager );
...
```

2. Reuse the `HttpClient` object from each thread to make REST API requests instead of creating a new `HttpClient` object for each thread.

Resources overview

The REST XML resources in Focal Point are service, element collections, element and attributes.These are the resource types and example REST API URIs for the resources:

Resource type	**Example REST API URI**	**Description**	**Supported REST operations**
Service document	`http://`*`fpserver`*`/fp/resources/`	The service document has the high level resources for workspace, module, views that a user has access to.	GET
Element collection	`http://`*`fpserver`*`/fp/resources/workspaces/1/modules/1/elements/`	An element collection has the contents of a view or module.	GET POST PUT
Element	`http://`*`fpserver`*`/fp/resources/workspaces/1/modules/1/elements/1`	Element has contents of attributes	GET
Attribute	`http://`*`fpserver`*`/fp/resources/workspaces/1/modules/1/elements/1/attributes/1`	Attribute	GET PUT POST DELETE

Links

The XML representations of Focal Point resources can be linked to other resources by using the XML element link. The link is similar to the link in the Atom format.

For more information, see *http://www.atomenabled.org/developers/syndication/atom-format-spec.php#element.link*

Link attributes	Description
href	(mandatory) The URI of the referenced resource.
rel	(optional) The rel attribute can have one of the following attributes: ▪ **alternate**: An alternate representation of the resource. Example: an HTML page ▪ **enclosure**: A related resource that might be large and require special handling. Example: a binary file ▪ **related**: A related resource ▪ **self**: The resource itself ▪ **edit**: A reference to a resource that can be edited, which might be the resource itself
title	(optional) A human-readable title of the link.
type	(optional) The media (MIME) type of the resource.
hreflang	(optional) The language of the resource.
length	(optional) The size of the referenced resource in bytes.

Service documents

The service document has the high level resources for workspace, module, views that a user has access to. Only users with workspace administrator rights can access modules.

The resource URI for the service document is:

```
http://focalpointserver/fp/resources/
```

GET method

Accept header: application/xml

Response body:

```
<fp:service xsi:schemaLocation="http://schema.example.com/focalpoint/resources
https://fpserver.com:9443/fp/dtd/service.xsd"> <fp:workspace> <fp:title>SmartCloud</fp:title>
<fp:alias>625ca8ed-d4c6-40a0-b8a2-ccc0319726a5</fp:alias> <fp:modules> <fp:collection
addable="true"> <fp:title>Elements</fp:title> <fp:alias>0f615fdc-ed3d-4ce5-9eff-
4f9b625986e8</fp:alias> <fp:indexList
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/"/> <fp:indexTree
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/?tree=true"/>
<fp:fullList
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/?includeAttribute
s=true"/> <fp:fullTree
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/?tree=true&includ
eAttributes=true"/> </fp:collection>
```

Service document description

Each module and view contains an element collection. You can view the element collection in these ways in the service document:

Link	**Visualization**
indexList	A list of elements that includes only the title and URI of each element. If a sort attribute is defined, the elements are sorted accordingly.
indexTree	A tree structure of elements that includes only the title and URI of each element. The title and not the URI of folders that not are displayed is included.
fullList	A list of elements that includes all attribute values for the attributes in the view or module. If a sort attribute is defined, the elements are sorted accordingly.
fullTree	A tree structure of elements that includes the attribute values for the attributes in the view or module.

For details and annotations of the XML format for the service document, see the XML Schema document. The URI for the schema is in the `schemaLocation` XML attribute of the service document.

Element collection

An element collection has the contents of a view or module. The URI for an element collection is available in the service document and the general URI is:

`http://focalpointserver/context/resources/workspaces/1/modules/1/elements/`

The XML representation of the element collection contains a URI for each element in the element collection. Depending on the link that was chosen in the service document, the collection might include attribute values and might be structured as tree or a flat list.

A single element contains all attribute values and contains more details than the collection. The XML format for the element collection is defined by the XML Schema document in the schemaLocation XML attribute. The XML Schema is unique for each view and module, and dynamically reflects the attribute setup of the module or view. The schema might change in these situations:

- an attribute is added, removed, or changed
- a view definition is changed
- a module is renamed.

If a collection is an add view or a module, you can add an element to the collection by using POST operation. The body of the request must contain an XML representation of the element. The XML format to use is in the XML Schema for the collection. If the collection is an add view, the contents of the view are in the folders. The folders can be parents when a new element is added.

Note RESTful APIs ignore the filters that are applied in views. All the elements of a view are displayed.The resource URI for the service document is:

The resource URI for the service document is in the form: http://*fpserver*/fp/resources/

GET method

Use this method to retrieve the collection of elements in a module. You can use the View parameter to filter the elements and attributes.

URL:
`https://`*`fpserver`*`.com:9443/fp/resources/workspaces/2/modules/1/elements/.xml?includeAttributes=true&view=14`

Accept header: `application/xml`

Response body:

```
<ns:elementCollection
xsi:schemaLocation="https://fpserver.com:9443/fp/namespace/workspaces/2/modules/1/views/14/ele
ments https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements.xsd?view=14">
<ns:Elements count="1"> <ns:Element> <ns:attributes> <ns:Element_Information/> <ns:ID></ns:ID>
<ns:Title></ns:Title> <ns:Description></ns:Description> <ns:type_is_a_folder>
</ns:type_is_a_folder> </ns:attributes> <ns:selfLink
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/15?view=14"/>
<ns:alias>38bfbaaa-6af2-49ed-a6f4-d0cf39704a55</ns:alias> </ns:Element></ns:Elements> <fp:link
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/?view=14"
rel="alternate"/> <fp:link
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/?view=14&includeA
ttributes=true&tree=true" rel="alternate"/> <fp:link
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/?view=14&tree=tru
e" rel="alternate"/></ns:elementCollection>
```

Supported parameters for the GET method

The following parameters can be used with the GET method:

- **includeAttributes**

  Set the `includeAttributes` parameter to `true` to retrieve the attributes of the elements.

  Type: Boolean

- **view**

  Use the `view` parameter to specify the view ID for retrieving the elements or attributes that are in the view.

Type: Integer

- **filter**

  The `filter` parameter must be used with the `view` parameter. Use the `filter` parameter to pass the filter ID for retrieving elements and attributes based on the filter criteria.

  Type: Integer

  For example, `view=14&filter=1`.

- **tree**

  Set the `tree` parameter to `true` to view the element collection in the tree structure

  Type: Boolean

- **modifiedSince**

  Use the `modifiedSince` parameter to retrieve the element that has changed since a specified date. The following formats are supported.

  Note The user can also supply `modifiedSince` as a parameter.

  ```
  yyyy-MM-dd'T'HH:mm:ss
  yyyy-MM-dd'T'HH:mm
  yyyy-MM-dd
  yyyy-MM-dd HH:mm:ss
  yyyy-MM-dd HH:mm
  ```

  Type: Date or DateTime

  For example, `2020-04-19T15:16:38+05:30 or 2020-01-22T12:00:00+01:00`.

- **metadata**

  Set the `metadata` parameter to `schema` to retrieve the schema of the element collection XML document.

  Type: String

  For example, `metadata=schema`.

- **fields**

  Use the `fields` parameter to pass XPath expression to retrieve a specific set of attributes or to filter elements that are based on specific values of attributes.

  Type: XPath expression

  For example, to retrieve a specific set of attributes, use:

  ```
  fields=elementCollection/Element/attributes/(ID|Title|Owner))
  ```

  or to filter elements based on attribute values, use:

```
fields=elementCollection/Element/attributes/(Title contains X OR Status
is Approved )
fields=elementCollection/Element/attributes/(Title =  X | Status is
Approved )
fields=elementCollection/Element/attributes/(Title is X AND Status =
Approved )
fields=elementCollection/Element/attributes/(Title contains X AND Cost
< 100 )
```

Notes

- Separator '|' should be encoded as `'%7C'`.

  For example, `fields=elementCollection/Element/attributes/(ID | Title)` should be encoded as `fields=elementCollection/Element/attributes/(ID %7C Title)`.

- Attribute names and the attribute values should be enclosed in single quotes if they contain the following special characters: `(`, `)`, <, >, `|`, `=`.

  For example, `fields=elementCollection/Element/attributes/('Comments(incl)' contains 'Shift>Technology')`.

  If the names or values already contain single quotes (') then they must be escaped with a backslash as: \'.

- The following characters or character sequences can be used as operators: `=`, <, >, `is`, `contains`.

- Different filtering conditions can be connected by using the following connectors (case sensitive): `AND`, `OR`, `|` (same as `OR`).

- If you are using the `fields` parameter to filter the elements, do not use the `modifiedSince` parameter as well. Instead, use the `fields` parameter with a filtering condition specified on the Last Changed Date attribute.

- You can specify filtering conditions on only the following attributes: Text, Integer, Float, Date, Choice, Check Box, MulitChoice, Parent Folder, Created Date, Created By, Last Changed Date, Last Changed By, Owner, Link, Linklist, Incoming Link, Unique Id.

- For attributes like Choice and MultiChoice, you can specify the name of the items as the filtering value. For Link, Linklist and Incoming Link, you must specify the id of the target element as the filtering condition. For Parent Folder, Owner, Created By and Last Modified By attributes, you can either specify the name or the id of the target element.

- To filter based on more than one attribute value, provide the filtering condition by comma-separating the individual values.

  For example, the following example filters the elements where ‘Geography’ contains both ‘Asia’ and ‘Europe’ (applies 'all' condition):

```
fields=elementCollection/Element/attributes/(Geography contains Asia,
Europe).
```

- For multivalued attributes like MultiChoice and Linklist, you can prefix the filtering condition with `or`, to specify the 'any' condition while filtering.

  For example, the following example filters the elements where ‘Geography’ contains either Asia or Europe (applies 'any' condition):

  ```
  fields=elementCollection/Element/attributes/(Geography contains
  or,Asia,Europe).
  ```

- For parent folder attribute, prefix the filtering condition with `r`, to set recursive rule.

  For example, the following example filters all elements where the parent folder is `System` or any subfolder of `System`:

  ```
  fields=elementCollection/Element/attributes/(Parent Folder is
  r,System).
  ```

- **paging**, **pageSize**, **pageNo**

  Use the `paging`, `pageSize`, and `pageNo` parameters to retrieve the collection of elements that are paginated. When pagination is enabled, the XML shows an additional link to the next page.

  Types: `paging`: Boolean, `pageSize`: Integer, `pageNo`: Integer

  For example:

  ```
  <fp:link
  href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/eleme
  nts/?pageno=1&pageSize=2&paging=true&includeAttributes=true" rel="next"
  Page="0"/>
  ```

  ```
  paging=true&pageSize=2&pageNo=1
  ```

  Note The user can also supply `pageSize` or `pageNo` as a parameter.

- **includeHistoryOfAttributes**, **start**, **end**

  Set the `includeHistoryOfAttribute` parameter to `true` and specify the `start` and `end` date parameters to retrieve the history of the attributes. The following formats are supported for the date parameters.

  ```
  yyyy-MM-dd'T'HH:mm:ss
  yyyy-MM-dd'T'HH:mm
  yyyy-MM-dd
  yyyy-MM-dd HH:mm:ss
  yyyy-MM-dd HH:mm
  ```

  Types: `includeHistoryOfAttributes`: Boolean, `start`: Date or Date Time, `end`: Date or Date time

  For example:

`includeHistoryOfAttributes=true&start=2011-03-22T12:00:00+01:00&end=2011-06- 22T12:00:00+01:00`

`includeHistoryOfAttributes=true&start=2011-03-22T12:00+GMT+01:00&end=2011-06-22T12:00+GMT+01:00`

- **genericTimeGrid**

  Set the `genericTimeGrid` parameter to `true` to retrieve the time grid attribute sheets in a generic format. This can be used for reporting by using Rational Insight.

  Note The user can also supply `genericTimeGrid` as a parameter.

  Type: Boolean

- **optimize**

  Set the `optimize` parameter to `true` to optimize the content. If this is set to `true`, the response does not include rich text and formula information of the attributes. Also alternate URLs to the element are not provided in the response.

  Type: Boolean

- **includeLinkTable**

  Set the `includeLinkTable` parameter to `true` to include attributes from the link targets in the response. The view should have a link table defined for the Link or LinkList attributes. All visible attributes shown in the link table set up will be part of the Link or LinkList attribute response.

  Type: Boolean

- **sortID**

  Use the `sortID` parameter to sort the elements in the collection based on an attribute in the view. Provide the id of the attribute as the value for the parameter.

  Type: Integer

  For example, `sortID=100`.

- **descending**

  Set the `descending` parameter to `true` to sort the elements in the collection in descending order. If this parameter is not specified, sorting is done in ascending order.

  Type: Boolean

POST method

Use the POST method to create a new element in a module.

The following example uses the POST method to create new elements where the user must have administrator permission for creating elements in the module.

URL:

`https://fpserver.com:9443/fp/resources/workspaces/3/modules/1/elements/.xml`

Accept header: `application/xml`

Request body:

```
<ns:newElement
xmlns:ns="https://fpserver.com:9443/fp/namespace/workspaces/3/modules/1/elements"
xmlns:fp="http://schema.ibm.com/focalpoint/resources" > <ns:Title> <fp:textValue
xmlns:fp="http://schema.ibm.com/focalpoint/resources">karthi 1</fp:textValue> </ns:Title>
<ns:MealPreference value="NonVeg"></ns:MealPreference> <ns:type_is_a_folder
value="true"></ns:type_is_a_folder> <ns:Comments> <ns:newTextListEntry> <fp:textValue
xmlns:fp="http://schema.ibm.com/focalpoint/resources">text entry 1</fp:textValue>
</ns:newTextListEntry> <ns:newTextListEntry> <fp:textValue
xmlns:fp="http://schema.ibm.com/focalpoint/resources">text entry 2</fp:textValue>
</ns:newTextListEntry> </ns:Comments> <ns:RelatedReleases> <fp:linkChange
value="https://fpserver.com:9443/fp/resources/workspaces/3/modules/3/elements/40"></fp:linkCha
nge> <fp:linkChange
value="https://fpserver.com:9443/fp/resources/workspaces/3/modules/3/elements/41"></fp:linkCha
nge> </ns:RelatedReleases><ns:Parent_Folder
value="https://fpserver.com:9443/fp/resources/workspaces/3/modules/1/elements/61"></ns:Parent_
Folder> </ns:newElement>
```

The root node `<ns:newElement>` is based on the module name. For example: if the module name is products, then the new element is product.

Supported parameters for the POST method

The following parameters can be used with the POST method:

- **optimize**

  Set the `optimize` parameter to true to optimize element creation.

  Type: Boolean

Using the POST method with the view parameter

The following example shows the method that allows non-administrators to create new elements by using the view parameter.

URL:

`https://fpserver.com:9443/fp/resources/workspaces/3/modules/1/elements/.xml ?view=200&optimize=true`

Accept header: `application/xml`

Request body:

```
<ns:newElement
xmlns:ns="https://fpserver.com:9443/fp/namespace/workspaces/3/modules/1/views/200/elements"
xmlns:fp="http://schema.ibm.com/focalpoint/resources" > ... </ns:newElement>
```

The root node (`<ns:newElement>`) is based on the module name. For example: if the module name is products, then the new element is product.

Multipart POST method

The following example uses the POST method to update multiple attributes through one HTTP request. The content type for this request is `multipart/related`. You can use the POST method only to add or delete element links.

URL:
`https://fpserver.com:9443/fp/resources/workspaces/5/modules/1/elements/.xml`

Accept header: `application/xml`

Content type: `multipart/related`

Request body:

```
<attributeChangeCollection xmlns:ns="https://schema.com/focalpoint/resources" >
<linkListChange href="/14/attributes/224"> <ns:addLink
value="https://fpserver.com:9443/fp/resources/workspaces/5/modules/3/elements/20"></ns:addLink
> <ns:addLink
value="https://fpserver.com:9443/fp/resources/workspaces/5/modules/3/elements/19"></ns:addLink
> <ns:deleteLink
value="https://fpserver.com:9443/fp/resources/workspaces/5/modules/3/elements/18"></ns:deleteL
ink> </linkListChange> <textListEntryChange href="./14/attributes/223/entries/" >
<ns:textValue>one</ns:textValue> </textListEntryChange> </attributeChangeCollection >
```

PUT method

The following example shows how to use the PUT method to update multiple attributes through one HTTP request. The content type for this request is `multipart/related`.

URL:
`https://fpserver.com:9443/fp/resources/workspaces/5/modules/1/elements/.xml`

Accept header: `application/xml`

Content type: `multipart/related`

Request body:

```
<attributeChangeCollection xmlns:ns="http://schema.com/focalpoint/resources">
<textAttributeChange href="./14/attributes/2"> <ns:textValue>new text value</ns:textValue>
</textAttributeChange> <textAttributeChange href="./15/attributes/2"> <ns:textValue>new text
value</ns:textValue> </textAttributeChange> <textListEntryChange
href="./14/attributes/223/entries/212309123" > <ns:textValue>replaced value</ns:textValue>
</textListEntryChange> </attributeChangeCollection>
```

Elements

An element contains all the attribute values of the element.

GET method

Use the GET method to retrieve the element details. You can use the View parameter to filter the attributes.

URL:
`https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/001.xml`

Accept header: `application/xml`

Response body:

```
<ns:Element
xsi:schemaLocation="https://fpserver.com:9443/fp/namespace/workspaces/2/modules/1/element
https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/element.xsd">
<ns:attributes> <ns:ID writable="false"> <ns:uniqueId value=""/> <ns:alias>799af0ed-6316-4116-
8749-ce2353e80ac5</ns:alias> </ns:ID> <ns:Title writable="true"> <ns:text></ns:text>
<ns:alias>7d5c25cb-e2a2-4d94-8b03-1cf603d2833b</ns:alias> <ns:editLink
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/1/attributes/2"/>
</ns:Title> <ns:Description writable="true"></ns:Description> <ns:type_is_a_folder wr
itable="true"></ns:type_is_a_folder> <ns:check_boxes_are_selected_in_the_chart
writable="true"></ns:check_boxes_are_selected_in_the_chart> <ns:Element_Information
writable="false"/> <ns:Owner writable="true"></ns:Owner> <ns:Creator
writable="false"></ns:Creator> <ns:Created_Date writable="false"></ns:Created_Date>
<ns:Last_Changed_By writable="false"></ns:Last_Changed_By> <ns:Last_Changed_Date
writable="false"></ns:Last_Changed_Date> <ns:Parent_Folder writable="true"></ns:Parent_Folder>
<ns:Text_List writable="true"></ns:Text_List> </ns:attributes><ns:selfLink
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/1"/> <ns:mspLink
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/1?format=mspxml"/
> <ns:htmlLink href="https://fpserver.com:9443/fp/servlet/Login?go=2,1"/> <ns:alias>7602dc18-
d64c-4360-8098-de0b0a385bf8</ns:alias> </ns:Element>
```

Supported parameters for the GET method

The following parameters can be used with the GET method:

- **view**

  Use the `view` parameter to specify the view ID for retrieving the attributes that are in the view.

  Type: Integer

- **filter**

  The `filter` parameter must be used with the `view` parameter. Use the `filter` parameter to pass the filter ID for retrieving attributes based on the filter criteria.

  Type: Integer

  For example, `view=14&filter=1`.

- **metadata**

  Set the `metadata` parameter to `schema` to retrieve the schema of the XML document for the element.

  For example, `metadata=schema`.

- **fields**

  Use the `fields` parameter to pass an XPath expression to retrieve specific set of attributes.

  Type: String

  For example, `fields=attributes/(ID|Title|Owner)`.

- **includeHistoryOfAttribute**, **start**, **end**

  Set the `includeHistoryOfAttribute` parameter to `true` and specify the `start` and `end` date parameters to retrieve the history of the attributes. The following formats are supported for the date parameters.

  ```
  yyyy-MM-dd'T'HH:mm:ss
  yyyy-MM-dd'T'HH:mm
  yyyy-MM-dd
  yyyy-MM-dd HH:mm:ss
  yyyy-MM-dd HH:mm
  ```

  Types: `includeHistoryOfAttributes`: Boolean, `start`: Date or Date Time, `end`: Date or Date time

  For example:

  ```
  includeHistoryOfAttributes=true&start=2011-03-22T12:00:00+01:00&end=2011-
  06- 22T12:00:00+01:00
  ```

  ```
  includeHistoryOfAttributes=true&start=2011-03-
  22T12:00+GMT+01:00&end=2011-06-22T12:00+GMT+01:00
  ```

- **genericTimeGrid**

  Set the `genericTimeGrid` parameter to `true` to retrieve the time grid attribute sheets in a generic format. This can be used for reporting by using Rational Insight.

  Note The user can also supply `generictimegrid` as a parameter.

  Type: Boolean

- **optimize**

  Set the `optimize` parameter to `true` to optimize the content. If the `optimize` parameter is set to `true`, the response does not include rich text and formula information of the attributes. Also alternate URLs to the element are not provided in the response.

  Type: Boolean

- **includeLinkTable**

  Set the `includeLinkTable` parameter to `true` to include attributes from the link targets in the response. The view should have a link table defined for the Link or LinkList attributes. All visible attributes that are shown in the link table set up will be part of the Link or LinkList attribute response.

  Type: Boolean

Attributes

The representation of an attribute contains the value of the attribute. Certain attributes have multiple collection of values and each value has attribute URIs. For example: TextList and File. The representation of the single attribute value also includes the edit links for each text entry in the list. You can change the attribute values of elements in the following ways:

- To update the attribute types CheckBox, Choice, Date, Float, Integer, Link, LinkList, Matrix, Multichoice, Text, Time Grid, URL, and UniqueId, use a PUT operation.
- To add, update, or delete a text entry, link entry or a file by using the TextList, LinkList and File attributes, use a POST operation, a PUT operation, or a DELETE operation respectively.
- To increment a Version attribute to the next major version, use a POST operation.

The XML format to update an attribute value is different from the XML representation of the value. For example, a Text attribute might have both a text value and an expression, but when the attribute is updated, it can be given either a new text value or an expression. When you update an attribute, use the XML that is described by the XML Schema document for the attribute.

GET method

Use this method to retrieve the attribute details.

URL:
`https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/15/attributes/220.xml`

Accept header: `application/xml`

Response body:

```
<ns:Text_List writable="true"
xsi:schemaLocation="https://fpserver.com:9443/fp/namespace/workspaces/2/modules/1/attribute
https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/15/attributes/220/attri
bute.xsd"> <ns:textList> <fp:textListEntry createdDate="2013-05-02T13:50:09+05:30"
lastChangedDate="2013-05-02T13:50:09+05:30"> <fp:author title="Admin"/><fp:lastChangedBy
title="Admin"/> <fp:textValue>Sample Text 1</fp:textValue> <fp:formattedTextValue>Sample Text
1<BR></fp:formattedTextValue> <fp:xhtmlTextValue><div>Sample Text 1</div></fp:xhtmlTextValue>
<fp:link
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/15/attributes/220
/entries/1367482809131" rel="edit"/> </fp:textListEntry> <fp:textListEntry createdDate="2013-
05-02T13:50:17+05:30" lastChangedDate="2013-05-02T13:50:17+05:30"> <fp:author
title="Admin"/><fp:lastChangedBy title="Admin"/> <fp:textValue>Sample Text 2</fp:textValue>
<fp:formattedTextValue>Sample Text 2</fp:formattedTextValue> <fp:xhtmlTextValue><div>Sample
Text 2</div></fp:xhtmlTextValue> <fp:link
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/15/attributes/220
/entries/1367482817756" rel="edit"/> </fp:textListEntry> </ns:textList> <ns:alias>ed2ad85f-
5b87-4704-b384-e041888439c0</ns:alias> <ns:editLink
href="https://fpserver.com:9443/fp/resources/workspaces/2/modules/1/elements/15/attributes/220
"/> </ns:Text_List>
```

Supported parameters for the GET method

The following parameters can be used with the GET method:

- **view**

  Use this parameter to specify the view ID for retrieving the attributes that are in the view.

- **filter**

  This must be used with the view parameter. Use this parameter to pass the filter ID for retrieving attributes based on the filter criteria.

  Type: Integer

  For example, `view=14&filter=1`.

- **metadata**

  Set the `metadata` parameter to `schema` to retrieve the schema of the XML document for the element.

  For example, `metadata=schema`.

- **includeHistoryOfAttribute**, **start**, **end**

  Set the `includeHistoryOfAttribute` parameter to `true` and specify the start and the end dates to retrieve the history of the attributes. The following formats are supported.

  ```
  yyyy-MM-dd'T'HH:mm:ss
  yyyy-MM-dd'T'HH:mm
  yyyy-MM-dd
  yyyy-MM-dd HH:mm:ss
  yyyy-MM-dd HH:mm
  ```

  Types: `includeHistoryOfAttributes`: Boolean, `start`: Date or Date Time, `end`: Date or Date time

  For example:

  ```
  includeHistoryOfAttributes=true&start=2011-03-22T12:00:00+01:00&end=2011-
  06- 22T12:00:00+01:00
  ```

  ```
  includeHistoryOfAttributes=true&start=2011-03-
  22T12:00+GMT+01:00&end=2011-06-22T12:00+GMT+01:00
  ```

- **genericTimeGrid**

  Set the parameter to `true` for retrieving the time grid attribute sheets in a generic format. This can be used for reporting using Rational Insight.

  Type: Boolean

POST method

Use the POST method to add a new entry to the attribute collection. The following examples show how to use the POST method to add new values to an attribute TextList collection.

Note The user must have administrator permission for adding values to the attribute collection.

- Adding new values to a TextList attribute

  URL:
  `http://fpserver.com:8091/fp/resources/workspaces/2/modules/1/elements/53/attributes/237.xml`

  Accept header: `application/xml`

  Request body:

  ```
  <textListEntryChange
  xmlns:ns="http://schema.ibm.com/focalpoint/resources">
  <ns:textValue>one</ns:textValue> </textListEntryChange>
  ```

- Adding new values to a File attribute

  URL:
  `http://fpserver.com:8091/fp/resources/workspaces/2/modules/1/elements/53/attributes/228.xml`

  Accept header: `application/xml`

  Request body:

  ```
  <fileChange xmlns:ns="http://schema.ibm.com/focalpoint/resources">
  <ns:fileName>a.txt</ns:fileName>
  <ns:contentType>plain/text</ns:contentType>
  <ns:fileData>YWJj</ns:fileData> </fileChange>
  ```

- Adding new values to a LinkList attribute and deleting a link

  URL:
  `http://fpserver.com:8091/fp/resources/workspaces/2/modules/1/elements/53/attributes/236.xml`

  Accept header: `application/xml`

  Request body:

  ```
  <linkListChange xmlns:ns="http://schema.ibm.com/focalpoint/resources">
  <ns:addLink
  value="http://fpserver.com:8091/fp/resources/workspaces/2/modules/13/elem
  ents/30"> </ns:addLink> <ns:deleteLink
  value="http://fpserver.com:8091/fp/resources/workspaces/2/modules/13/elem
  ents/29"></ns:deleteLink> </linkListChange>
  ```

- Updating the value of the Version attribute to the next major version

  URL:
  `http://fpserver.com:8091/fp/resources/workspaces/2/modules/1/elements/53/attributes/245.xml`

Accept header: `application/xml`

Request body:

```
<VersionAttribute nextMajorVersion="true"></VersionAttribute>
```

PUT method

The following table gives examples of how to update the specified attribute values by using the PUT method:

Attribute type	**Request body**
Check Box, Lock	`<checkBoxChange value="false"> </checkBoxChange>`
Choice	`<choiceAttributeChange value="Closed"> </choiceAttributeChange>`
Date	`<dateChange xmlns:ns="http://schema.ibm.com/focalpoint/resources"> <ns:value>2011-01-01T00:00:00+05:30</ns:value> </dateChange>`
Float	`<floatChange xmlns:ns="http://schema.ibm.com/focalpoint/resources"> <ns:value>201</ns:value> </floatChange>`
File	URL example: http://localhost:8080/fp/resources/workspaces/3/modules/13/elements/16/attributes/237/files/01.xml <fileChange xmlns:ns="http://schema.ibm.com/focalpoint/resources"> <ns:fileName>a.txt</ns:fileName> <ns:contentType>plain/text</ns:contentType> <ns:fileData>YWJj</ns:fileData> </fileChange>
Link	`<linkValueChange value="http://fpserver.com:8091/fp/resources/workspaces/2/modules/1/elements/16"> </linkValueChange>`
Integer	`<integerChange xmlns:ns="http://schema.ibm.com/focalpoint/resources"> <ns:value>201</ns:value> </integerChange>`
List (Link)	`<linkChange xmlns:ns="http://schema.ibm.com/focalpoint/resources"> <ns:linkChange value="http://fpserver.com:8091/fp/resources/workspaces/2/modules/1/elements/72"> </ns:linkChange> <ns:linkChange value="http://fpserver.com:8091/fp/resources/workspaces/2/modules/1/elements/73"> </ns:linkChange> </linkChange>`

Attribute type	Request body
List (Text)	URL example: `http://localhost:8080/fp/resources/workspaces/3/modules/13/elements/19/attributes/247/entries/1369815257240.xml <textListEntryChange xmlns:ns="http://schema.ibm.com/focalpoint/resources"> <ns:textValue>one</ns:textValue> </textListEntryChange>`
Matrix	`<matrixChange xmlns:ns="http://schema.ibm.com/focalpoint/resources"> <ns:row> <ns:cell> <ns:value>A</ns:value> </ns:cell> <ns:cell> <ns:value>B</ns:value> </ns:cell> </ns:row> <ns:row> <ns:cell> <ns:value>Row 1</ns:value> </ns:cell> <ns:cell> <ns:value>Simple text value</ns:value> </ns:cell> <ns:cell> <ns:expression>1 + 2</ns:expression> </ns:cell> </ns:row> </matrixChange>`
MultiChoice	`<multichoiceAttributeValues xmlns:ns="http://fpserver.com:8091/fp/namespace/workspaces/2/modules/1/attribute"> <ns:selected>Open</ns:selected> <ns:selected>One</ns:selected> </multichoiceAttributeValues>`
Text	`<textAttributeChange xmlns:ns="http://schema.ibm.com/focalpoint/resources"> <ns:textValue>new text value</ns:textValue> </textAttributeChange>`
Timegrid (Header)	`<timeGridChange xmlns:ns="http://fpserver.com:8091/fp/namespace/workspaces/2/modules/1/attribute"> <ns:timeGridSetting sheetId="1" startDate="2011-12-31" endDate="2012-03-30"> </ns:timeGridSetting> </timeGridChange>`
TimeGrid (Data)	`<timeGridChange xmlns:ns="http://fpserver.com:8091/fp/namespace/workspaces/2/modules/1/attribute"> <ns:cell sheetId="1" date="2011-12-31" columnId="A" rowId="1"> <ns:value>23</ns:value> </ns:cell> <ns:cell sheetId="1" date="2011-12-31" columnId="A" rowId="2"> <ns:value>24</ns:value> </ns:cell> </timeGridChange>`
UniqueID	`<uniqueIdChange value="5671"> </uniqueIdChange>`
URL	`<urlChange value="www.ibm.com/developerworks"> </urlChange>`

DELETE method

The following table gives examples of how to delete the specified attribute values by using the DELETE method:

Attribute type	Request body
File	URL example: To delete the `01.xml` file from the file attribute, specify the following URL and use the delete command. `http://localhost:8080/fp/resources/workspaces/3/modules/13/elements/16/attributes/237/files/01.xml`
List (Text)	URL example: To delete the `1369815257240.xml` entry from the list text attribute, specify the following URL and use the delete command. `http://localhost:8080/fp/resources/workspaces/3/modules/13/elements/19/attributes/247/entries/1369815257240.xml`