Revit Server 2012

REST Service

API Version: 1.0

Table of Contents

Overview

This reference specifies the REST service that Revit Server offers for server administration. It consists of two groups of APIs:

1. Information querying, which includes:
- Querying the properties of the server.
- Querying the contents of a specified folder.
- Querying the directory information of a specified folder.
- Querying the history of a specified model.
- Querying the thumbnail of a specified model.
2. Data managing, which includes:
- Locking and unlocking the entire server, a specified folder including all its descendents, or a specified model.
- Querying and unlocking - a specified folder's descendents.
- Copying or moving a specified folder, or a specified model.
- Creating a new folder.
- Renaming a specified folder or model.
- Deleting a specified folder or model.

Versions

The version of Revit Server specified in this reference is 2012, and the version of APIs is 1.0.

Glossary

- Object: an object refers to the server, a folder, or a model.

Common Behaviors

This section specifies constraints that apply to all REST APIs.

Transport Protocol

All APIs are based on the Hypertext Transfer Protocol, version 1.1 (RFC 2616).

Media Type

Request bodies and response bodies are normally encoded in JSON, as specified in RFC4627.

Base URI

```
http://<host>/RevitServerAdminRESTService/AdminRESTService.svc
```

- <host>: the name or IP address of Revit server.

Object Path

In most APIs, object paths are included as segments or parameters in the URLs. All paths are relative to the server's root.

Since slashes "\" and "/" are URL's special characters, so object paths are formatted as:

- Server's root: *"|"*
- Folder Path: *"folderName1[|folderName2[…]]"*
- Model Path: *"[folderName1[|folderName2[…]|]]modelName.rvt"*

Query String

Some APIs require query strings. These strings are case-sensitive. Invalid parameter names will result in "405 Method Not Allowed" errors, and invalid parameter values will result in "400 Bad Request" errors.

Common Request Headers

Header Name	Description	Required
API-Version	Specifies the version of API the request needs to be directed to. Supported value: 1.0 Default value: 1.0	Optional
User-Name	Specifies the user name of the client. It is used to form an application-level session token. Supported value: *string*	Yes
User-Machine-Name	Specifies the machine name of the client. It is used to form an application-level session token. Supported value: *string*	Yes
Operation-GUID	Specifies a GUID for the request. It is used for server side logging and diagnostics, so a unique GUID for every request is preferred. Supported value: *GUID in string format*	Yes

Common Response Headers

Header Name	Description	Required
Content-Length	Describes the length in bytes of the response body. Supported value: *integer in string format*	Yes

Content-Type	Describes the representation and syntax of the response body. Supported value: application/json, image/png Condition: Required by the APIs that return data.	Conditional
Location	Returns a new URI that can be used to request a representation of the newly created object. Supported value: *absolute URI*. Condition: Required by the APIs that create new objects.	Conditional

Common Status Codes

Status Code	**Description**
200 OK	The request has succeeded.
201 Created	The request has been fulfilled and resulted in a new object being created.
400 Bad Request	The request could not be understood by the server due to incorrect syntax or invalid parameter values.
404 Not Found	The object specified in the request's URL could not be found.
405 Method Not Allowed	The method (GET, PUT, DELETE, POST) specified in the request is not allowed for the request's URL.
414 Request-URI Too Long	Path of the object specified in the request's URL is longer that the server supports.
500 Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.
501 Not Implemented	The server does not support the API version required to fulfill the request.
502 Bad Gateway	The server, while acting as a local server, received an invalid response from the central server.
503 Service Unavailable	The server is currently unable to handle the request due to a temporary overloading or maintenance of the server.
504 Gateway Timeout	The server, while acting as a local server, did not receive a timely response from the central server.

Information Querying APIs

This section specifies APIs that are used to query information about server's properties and information about the data on the server.

GET /serverProperties

Queries server's properties.

Request

URL

```
GET /serverProperties
```

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Body

```
{
   "IsCentralServer": Boolean,
   "CentralServerName": string,
   " MaximumFolderPathLength": int,
   " MaximumModelNameLength": int
}
```

- IsCentralServer: whether the server is a central server;
- CentralServerName: the name of the central server;
- MaximumFolderPathLength: the maximum folder path length that Revit server supports;
- MaximumModelNameLength: the maximum model path length that Revit server supports.

Response Status Codes

The implementation of this API only uses status codes common to all APIs. For more information, see Common Status Codes.

Example

The following example queries the properties of central server. The server's name is CentralServer, and the maximum lengths of folder path and model name it supports are 119 and 40 respectively.

Request:

```
GET /serverProperties

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 200 OK
Content-Length: 118
Content-Type: application/json; charset=utf-8

{
   "CentralServerName":"CentralServer",
   "IsCentralServer":true,
   "MaximumFolderPathLength":119,
   "MaximumModelNameLength":40
}
```

GET /{folderPath}/contents

Queries the contents of a folder.

Request

URL

```
GET /{folderPath}/contents
```

- *FolderPath*: the path of the specified folder.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Body

```
{
   "Path": string,
   "LockState": int,
   "Folders":
   [
      {"Name": string, "HasContents": false/true, "LockState": int, "Size": int}
      …
   ],
   "Models":
   [
      {"Name": string, "LockState": int, "ModelSize": int, "SupportSize": int}
      …
   ],
   "DriveSpace": int,
   "DriveFreeSpace": int
}
```

- Path: the folder path;
- LockState: the lock state of the folder or model (0 - unlocked; 1- locked; 2- has a locked ancestor; 3 – has one or more locked descendent; 4 – being unlocked; 5 – being locked);
- Folders: the list of sub-folders;
- Models: the list of models under the folder;
- Name: the name of the sub-folder or model;
- HasContents: whether the sub-folder has any contents;
- Size: the size in bytes of the sub-folder;

- ModelSize: the size in bytes of the model;
- SupportSize: the size in bytes of the auxiliary data (such as user temporary data) for the model;
- DriveSpace: the total space in bytes of the drive where the folder exists;
- DriveFreeSpace: the free space in bytes of the drive where the folder exists.

Response Status Codes

The implementation of this API only uses status codes common to all APIs. For more information, see Common Status Codes.

Example

The following example queries the contents of Folder_A, which has a sub-folder Folder_AB, and two models Model_01.rvt and Model_02.rvt.

Request:

```
GET /Folder_A/contents

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 200 OK
Content-Length: 330
Content-Type: application/json; charset=utf-8

{
   "Path":"Folder_A",
   "LockState":0,
   "Folders":
   [
      {"HasContents":false,"LockState":0,"Name":"Folder_AB","Size":0}
   ],
   "Models":
   [
      {"LockState":0,"ModelSize":4064712,"Name":"Model_01.rvt","SupportSize":4137},
      {"LockState":0,"ModelSize":4064712,"Name":"Model_02.rvt","SupportSize":4403}
   ],
   "DriveFreeSpace":173381672960,
   "DriveSpace":258461396992
}
```

GET /{folderPath}/DirectoryInfo

Queries the folder directory information

Request

URL

```
GET /{folderPath}/DirectoryInfo
```

- *FolderPath*: the path of the specified folder.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Body

```
{
   "Path": string,
   "DateCreated": DateTime,
   "DateModified": DateTime,
   "LastModifiedBy": string,
   "Size": int,
   "ModelSize": int,
   "FolderCount": int,
   "ModelCount": int,
   "IsFolder": Boolean,
   "Exists": Boolean
}
```

- Path: the folder path;
- DateCreated: the creation time;
- DateModified: the last modified time;
- LastModifiedBy: the user who did the last modification;
- Size: the size of the folder;
- ModelSize: the model size if the folder is a model's folder;
- FolderCount: the count of sub-folders under the folder;
- ModelCount: the count of models under the folder;
- IsFolder: whether the folder is a normal folder and not a model's folder;
- Exists: whether the folder exists.

Response Status Codes

The implementation of this API only uses status codes common to all APIs. For more information, see Common Status Codes.

Example

The following example queries the directory information of Folder_A, which has one sub-folder and two models.

Request:

```
GET /Folder_A/DirectoryInfo

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 200 OK
Content-Length: 212
Content-Type: application/json; charset=utf-8

{
   "Path":"Folder_A",
   "DateCreated":"\/Date(1294907700265)\/",
   "DateModified":"\/Date(1294971083276)\/",
   "Exists":true,
   "FolderCount":1,
   "IsFolder":true,
   "LastModifiedBy":null,
   "ModelCount":2,
   "ModelSize":0,
   "Size":8137964
}
```

GET /{modelPath}/history

Queries the submission history of a model.

Request

URL

```
GET /{modelPath}/history
```

- *ModelPath*: the path of the specified model.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Body

```
{
   "Path": string,
   "Items":
   [
      {
         "Comment": string,
         "Date": DateTime,
         "User": string,
         "VersionNumber": int,
         "ModelSize": int,
         "SupportSize": int
      }
      …
   ]
}
```

- Path: the model path;
- Items: the list of a model's submission history;
- Date: the date and time the submission was made to the model;
- User: the user who made the submission;
- VersionNumber: the version of the model by the submission;
- ModelSize: the size of the model at the time the submission was made;
- SupportSize: The size of the auxiliary data (such as user temporary data) for the model at the time the submission was made.

Response Status Codes

The implementation of this API only uses status codes common to all APIs. For more information, see Common Status Codes.

Example

The following example queries the submission history of Folder_B\Model_03.rvt, which has two versions that were saved by Alice and Bob respectively.

Request:

```
GET /Folder_B|Model_03.rvt/history

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 200 OK
Content-Length: 281
Content-Type: application/json; charset=utf-8

{
   "Path":"Folder_B\Model_03.rvt",
   "Items":
   [
      {
         "Comment":"",
         "Date":"\/Date(1294974276000)\/",
         "ModelSize":4023703,
         "SupportSize":3917,
         "User":"Alice",
         "VersionNumber":1
      },
      {
         "Comment":"",
         "Date":"\/Date(1294974347000)\/",
         "ModelSize":4030877,
         "SupportSize":3909,
         "User":"Bob",
         "VersionNumber":2
      }
   ]
}
```

GET /{modelPath}/thumbnail?width={width}&height={height}

Get the thumbnail of a model.

Request

URL

```
GET /{modelPath}/thumbnail?width={width}&height={height}
```

- *ModelPath*: the path of the specified model;
- *Width*: width of expected thumbnail.
- *Height*: height of expected thumbnail.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Body

Response body contains a PNG image.

Response Status Codes

The implementation of this API only uses status codes common to all APIs. For more information, see Common Status Codes.

Example

The following example gets a 128*128 thumbnail of Folder_B\Model_03.rvt.

Request:

```
GET /Folder_B|Model_03.rvt/thumbnail?width=128&height=128

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 200 OK
Content-Length: 1041
Content-Type: image/png

A PNG image in 128*128
```

Data managing APIs

This section specifies APIs that are used to manage data on the server.

Specific Status Codes

Status Code	Description
403 Forbidden	There was a lock conflict that prevented the request from being processed. A lock conflict could occur in following cases: - A lock has already been applied on the object specified in the request's URL, or on one of its ancestors or one of its descendents; - A data-managing request is being processed on the object specified in the request's URL, or one of its ancestors, or one of its descendents. All data-managing APIs except for lock-related ones have implicit locking and unlocking operations. Some APIs may allow certain types of lock conflict. See the APIs' "Remarks" section for details.
409 Conflict	The request could not be completed due to a conflict with the current state of the object specified in the request's URL. The conflict could be: - There is a Revit client's user lock on the object; - In "DELETE /{objectPath}" API: a. Destination already exists; b. Destination and source are same. - In "POST /{folderPath}/descendent" API: a. Destination folder is under source; b. When **CopyIncrement** is not specified, destination folder is the parent of source; c. When **Merge** is specified, destination object already exists and it is a model; d. When **Replace** or **ForceReplace** is specified, destination object doesn't exist; e. When **Replace** is specified, source model and destination model have duplicated data files.

PUT /{objectPath}/lock

Locks the server, a folder or a model.

Request

URL

```
PUT /{objectPath}/lock
```

- *ObjectPath*: the path of the server, the specified folder or the specified model.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Status Codes

The implementation of this API uses status codes common to all APIs, and codes specific to data managing APIs. For more information, see Common Status Codes and data managing APIs' Specific Status Codes.

Remarks

This API is idempotent, which means locking an already-locked object is allowed.

If a lock has been applied on an object, all data-managing requests except for lock-related requests on the object, its ancestors, and its descendents will be forbidden. Revit clients will also be forbidden to read and write any models related to the object in this case.

Example

The following example locks Folder_A\Model_01.rvt

Request:

```
PUT /Folder_A|Model_01.rvt/lock

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 200 OK
Content-Length: 0
```

DELETE /{objectPath}/lock?objectMustExist={objectMustExist}

Unlocks the server, a folder or a model.

Request

URL

```
DELETE /{objectPath}/lock?objectMustExist={objectMustExist}
```

- *ObjectPath*: the path of the server, the specified folder or the specified model.
- *ObjectMustExist*: whether the folder or model must exist.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Status Codes

The implementation of this API uses status codes common to all APIs, and codes specific to data managing APIs. For more information, see Common Status Codes and data managing APIs' Specific Status Codes.

Remarks

Parameter *ObjectMustExist* is used in the case of removing the lock record of a deleted folder or model.

This API is idempotent, which means unlocking an already-unlocked object is allowed.

Example

The following example unlocks Folder_A\Model_01.rvt

Request:

```
DELETE /Folder_A|Model_01.rvt/lock?objectMustExist=true

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 200 OK
Content-Length: 0
```

DELETE /{objectPath}/inProgressLock

Cancel the in-progress locking operation on the server, a folder or a model.

Request

URL

```
DELETE /{objectPath}/inProgressLock
```

- *ObjectPath*: the path of the server, the specified folder or the specified model.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Status Codes

The implementation of this API uses status codes common to all APIs, and codes specific to data managing APIs. For more information, see Common Status Codes and data managing APIs' Specific Status Codes.

Remarks

This API is idempotent, which means no error will be reported if the object has been locked or has been unlocked.

Example

The following example cancels the in-progress locking operation on Folder_A.

Request:

```
DELETE /Folder_A/inProgressLock

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 200 OK
Content-Length: 0
```

GET /{folderPath}/descendent/locks

Gets the lock information of the descendents of a folder.

Request

URL

```
GET /{folderPath}/descendent/locks
```

- *FolderPath*: the path of the specified folder.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Body

```
{
   "Path": string,
   "Items": [string…]
}
```

- *Path*: the folder path;
- *Items*: the list of paths of locked descendents.

Response Status Codes

The implementation of this API only uses status codes common to all APIs. For more information, see Common Status Codes.

Example

The following example gets the lock information of Folder_A's descendents, in which Model_01.rvt and Folder_AB have been locked.

Request:

```
GET /Folder_A/descendent/locks

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 200 OK
Content-Length: 76
Content-Type: application/json; charset=utf-8

{
   "Path":"Folder_A",
   "Items":
   [
      "Folder_A\\Model_01.rvt",
      "Folder_A\\Folder_AB"
   ]
}
```

DELETE /{folderPath}/descendent/locks

Unlocks all locks of the descendents of a folder.

Request

URL

```
DELETE /{folderPath}/descendent/locks
```

- *FolderPath*: the path of the specified folder.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Body

```
{
   "Path": string,
   "FailedItems": [string…]
}
```

- *Path*: the folder path;
- *FailedItems*: the list of paths of descendents that are not unlocked.

Response Status Codes

The implementation of this API uses status codes common to all APIs, and codes specific to data managing APIs. For more information, see Common Status Codes and data managing APIs' Specific Status Codes.

Example

The following example unlocks all locks of Folder_A's descendents.

Request:

```
DELETE /Folder_A/descendent/locks

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 200 OK
Content-Length: 38
Content-Type: application/json; charset=utf-8

{
   "Path":"Folder_A",
   "FailedItems":null
}
```

PUT /{folderPath}

Creates a new folder.

Request

URL

```
PUT /{folderPath}
```

- *FolderPath*: the path of the new folder to be created.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Status Codes

The implementation of this API uses status codes common to all APIs, and codes specific to data managing APIs. For more information, see Common Status Codes and data managing APIs' Specific Status Codes.

Remarks

The folder's parent folder must exist as a prerequisite of the request.

This API is idempotent, which means no error is reported if the folder already exists. "200 OK" instead of "201 Created" is returned in this case.

Example

The following example creates Folder_AB under Folder_A.

Request:

```
PUT /Folder_A|Folder_AB

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 201 Created
Content-Length: 0
Location: http://host/RevitServerAdminRESTService/ Folder_A|Folder_AB
```

DELETE /{objectPath}?newObjectName={newObjectName}

Rename or delete a folder or model.

Request

URL

```
DELETE /{objectPath}?newObjectName={newObjectName}
```

- *ObjectPath*: the path of the specified folder or model.
- *NewObjectName*: new name for the folder or model. Empty value means deleting the object.

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Status Codes

The implementation of this API uses status codes common to all APIs, and codes specific to data managing APIs. For more information, see Common Status Codes and data managing APIs' Specific Status Codes.

Remarks

Deletion is recursive and undoable, so please be cautious in using this operation.

If a model or one of its ancestors is renamed, all the corresponding local models on Revit clients will become invalid, even if you rename the central model back.

Example

The following example renames Folder_A\Folder_AB to Folder_A\Folder_AB_renamed.

Request:

```
DELETE /Folder_A|Folder_AB?newObjectName=Folder_AB_renamed

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 201 Created
Content-Length: 0
Location: http://host/RevitServerAdminRESTService/ Folder_A|Folder_AB_renamed
```

POST /{folderPath}/descendent?sourceObjectPath={sourceObjectPath}&pasteAction={pasteAction}&duplicateOption={duplicateOption}

Copies or moves a folder or a model to another folder.

Request

URL

```
POST /{folderPath}/descendent?sourceObjectPath={sourceObjectPath}&pasteAction={pasteAction}&duplicateOption={duplicateOption}
```

- *FolderPath*: the path of the destination folder.
- *SourceObjectPath*: the path of the source object;
- *PasteAction*: the action type, which must be one of following values:

Value	Description
Copy	Copy the source object to the destination folder.
Move	Move the source object to the destination folder.

- *DuplicateOption*: the duplicate option, which must be one of following values:

Value	Description	Conditions for Use
CopyIncrement	Append an incremental number to the destination object if it already exists.	/
Merge	Merge the source object into the destination object.	1) The destination object already exists; 2) The object to copy or move is a folder.
Replace	Replace the destination object with the source object.	1) The destination object already exists.
ForceReplace	Forcibly replace the destination object with the source object.	1) The destination object already exists; 2) The object to copy or move is a model; 3) The source and destination models have duplicated data file(s);

Request Headers

The implementation of this API only uses request headers common to all APIs. For more information, see Common Request Headers.

Response

Response Headers

The implementation of this API only uses response headers common to all APIs. For more information, see Common Response Headers.

Response Status Codes

The implementation of this API uses status codes common to all APIs, and codes specific to data managing APIs. For more information, see Common Status Codes and data managing APIs' Specific Status Codes.

Remarks

If a model or one if its ancestors is moved, all the local models on Revit clients will become invalid, even if you move the central model back.

Example

The following example move Folder_A\Model_01.rvt to Folder_B.

Request:

```
POST
/Folder_B/descendent?sourceObjectPath=Folder_A|Model_01.rvt&pasteAction=Move&duplicateOption=CopyIncrement

User-Name: Tester
User-Machine-Name: TestMachine
Operation-GUID: 45FB8158-8BE3-43E5-9DFA-318BDBCB4C93
```

Response:

```
HTTP/1.1 201 Created
Content-Length: 0
```