Agilent SLIMS

SLIMS REST API Manual

Notices

Manual Part Number

R4526A-SR670
August 31, 2021

Copyright

Agilent Technologies, Inc.
5301 Stevens Creek Blvd.
Santa Clara, CA 95051
USA

Software Revision

This guide is based on Rest API Version 1.0.

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such a license.

Restricted Rights Legend

U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

TABLE OF CONTENTS

OBJECTIVES

URI SCHEME

BasePath: /rest
Schemes: HTTP

PREFACE

This document covers the fundamental concepts for Developers using the Rest API For SLIMS to do basic database operations. The Rest API allows Developers to interact with SLIMS without using the GUI interface to perform operations such as fetching, removing, adding or updating SLIMS data.

AUDIENCE

Developer users of the SLIMS system who will fetch information for reporting or perform data manipulations using external tools through this API.

ASSISTANCE

To get technical assistance with the SLIMS Rest API, contact the developer.

DEFINITIONS

DisplayableEntityColumn«object»

Name	Schema
editable (*optional*)	Boolean
hidden (*optional*)	Boolean
name (*optional*)	String
position (*optional*)	Integer (int32)
title (*optional*)	String
value (*optional*)	Object

DisplayableEntityResource

Name	Schema
canDelete (*optional*)	Boolean
canUpdate (*optional*)	Boolean
columns (*optional*)	< DisplayableEntityColumn«object» > array
links (*optional*)	< Link > array
pk (*optional*)	Integer (int64)
tableName (*optional*)	String

Link

Name	Schema
href (*optional*)	String
rel (*optional*)	String

Model containing the mix values and ingredients pks

Name	Description	Schema
ingredientsPks (*optional*)	Ingredient Packages (Pks)	< integer (int64) > array
mixValues (*optional*)	Mix values	Object

Represents a fetch request for advanced criteria

Name	Schema

Name	Schema
criteria (*optional*)	Object
endRow (*optional*)	Integer (int64)
sortBy (*optional*)	< String > array
startRow (*optional*)	Integer (int64)

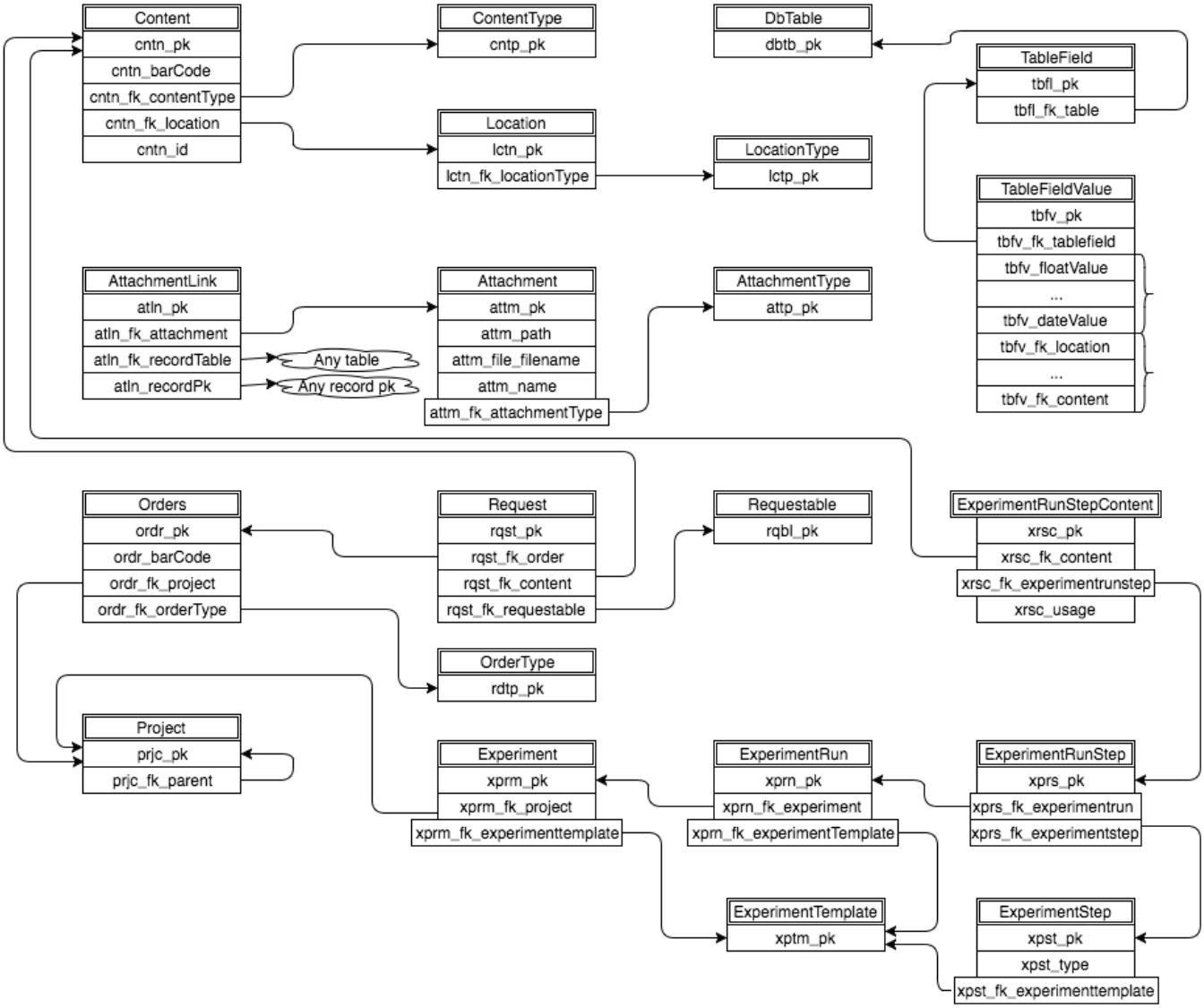
RestFetchResponse

Name	Schema
entities (*optional*)	< DisplayableEntityResource > array

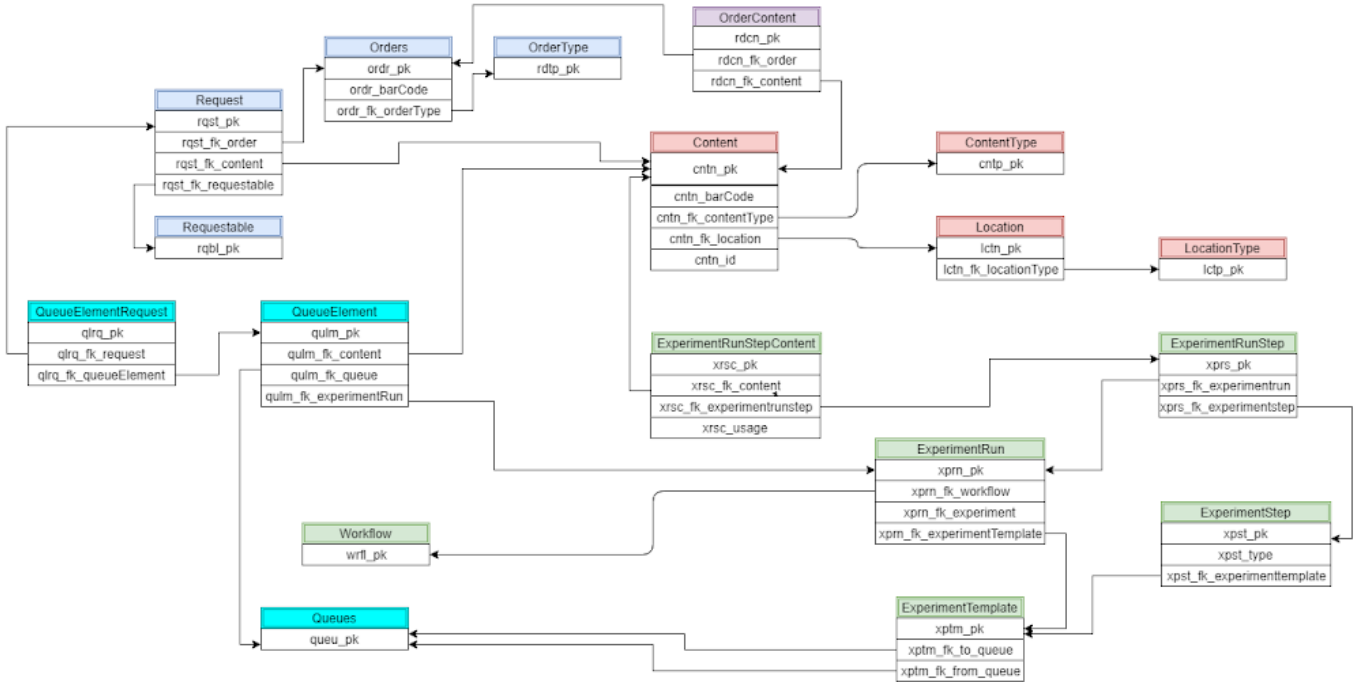
SIMPLIFIED DATABASE DIAGRAMS

These diagrams provide a simplified overview of the database structure for SLIMS Rest and Python users. It is designed to cover the essentials and map out the data model behind SLIMS at a general level.

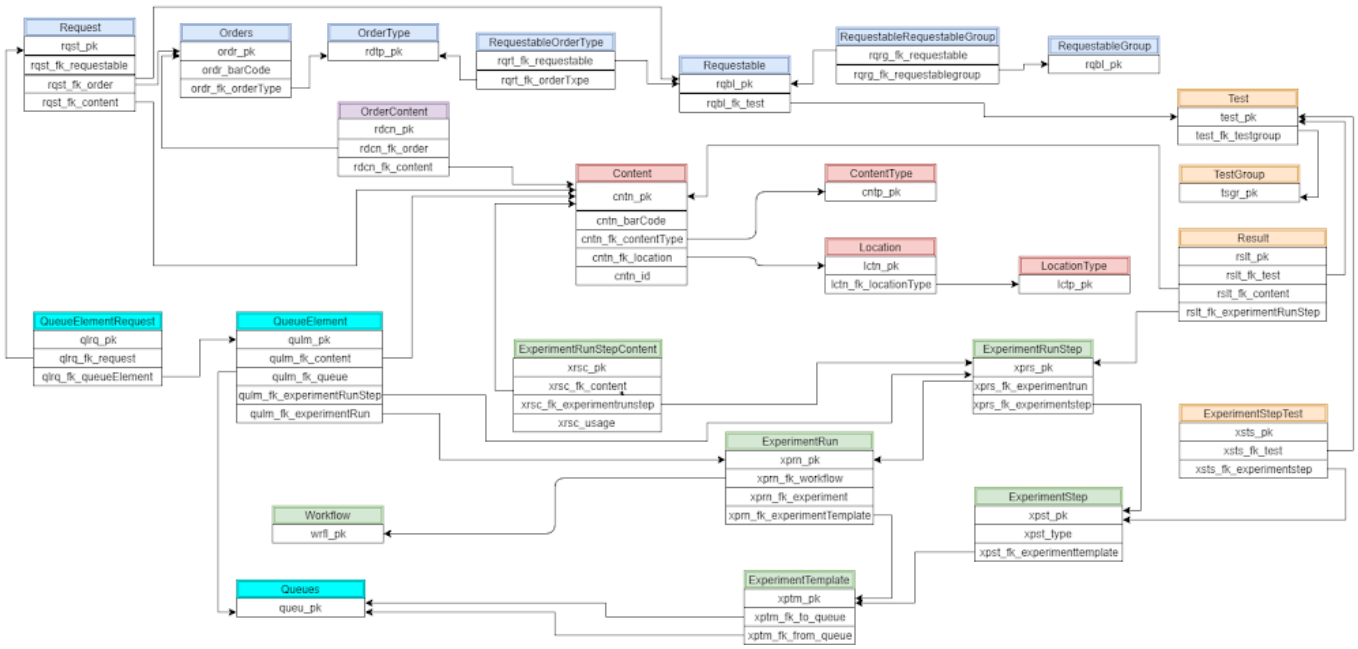
Sequencing Workflow Graph



Workflows



Analytical Workflows



BASIC TABLE OPERATIONS

Remove a record by pk

DELETE

```
REST Endpoint: /rest/{table}/{pk}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Delete a single Content.

Parameters:

Type	Name	Description	Datatype	Example
Path	**table** (*required*)	The table to remove a record from	String	Content
Path	**pk** (*required*)	The primary key of the record to remove	Integer(int64)	1

Responses:

- 200: OK
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Fetch Records by Criteria

GET

```
Rest Endpoint: /rest/{table}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Allows users to fetch on a table with certain basic criteria. For example, you can fetch all Contents by browsing to: http://slims.customer.com/slimsrest/rest/Content, and all Content with a certain id by using query parameters: http://slims.customer.com/slimsrest/rest/Content?cntn_id=SomeId. Usage of query parameters, like in the latter example, is not yet supported to try out in the online version of this manual.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Table** (*required*)	The table to look up	String	Content

Responses:

- 200: OK - Rest Fetch Response
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Fetch Records with Advanced Criteria, Sorting, and Paging

GET

```
REST Endpoint: /rest/{table}/advanced
Consumes: application/json
Produces: */*
```

Implementation Notes:

Allows users to fetch on a table with advanced criteria, server side sorting, and paging. The request body can contain the following criteria. All are optional.

Criteria:

Allows server side filtering. For example, you can fetch content for which the ID is equal to "example" like this:

```
{"fieldName": "cntn_id", "operator": "equals", "value": "example"}
```

Searches on all fields are allowed. The operators we currently support are:

```
equals
notEqual
greaterThan
lessThan
greaterOrEqual
lessOrEqual
contains
startsWith
endsWith
between
betweenInclusive
isNull
isNotNull
```

Some examples for this are:

```
{"fieldName": "cntn_id", "operator": "startsWith", "value": "DNA00"}
```

```
{"fieldName": "cntn_fk_location", "operator": "isNull"}
```

```
{"fieldName": "cntn_createdOn", "operator": "between", "start": "2016-11-
20T00:00:00.000000", "end": "2016-11-22T00:00:00.000000"}
```

Note the way of passing dates here. Criteria can be combined or negated using constructs like this:

```
{"operator":"and","criteria":[the list of criteria to combine]}
```

```
{"operator":"or","criteria":[the list of criteria to combine]}
```

```
{"operator":"not","criteria":[criteria to negate]}
```

Sorting

Sorts the returned list of records. You can pass a list here and the sorting will be applied after each other. For example:

```
["cntn_id", "cntn_barCode"]
```

The above will sort the returned content first on ID (ascending) and then on barcode (ascending). To sort descending, prepend the field with a hyphen (-). For example:

```
["-cntn_id"]
```

Start and End Row

These two properties only have effect if used together. Allows a user to only query for a subset of the matched records.

Complete Request

A complete request should look like this:

```
{
"criteria": {"fieldName": "cntn_id", "operator": "equals", "value":
"example"},
"sortBy": ["cntn_createdOn"],
"startRow":10,
"endRow":20
}
```

...And fetches the 10th to 20th samples with ID 'example' sorted by the creation date.

Try it out!

Contrary to standard HTTP specifications, the online version of this manual does not support GET requests with a body. But you can try it out using the following curl command:

```
curl -u user:password -X GET -H 'Content-Type: application/json' \
    -d '{"criteria": {"fieldName": "cntp_name", "operator": "equals",
"value": "DNA"}}' \
    http://slims.customer.com/slimsrest/rest/ContentType/advanced
```

Parameters:

Type	Name	Description	Datatype	Example
Path	**Table** (*required*)	The table to look up	String	Content
Body	**RestFetchRequest** (*required*)	The request	Object	See below

RestFetchRequest Example:

```
{
    "criteria": {
        "additionalProp1": {},
        "additionalProp2": {},
        "additionalProp3": {}
    },
    "endRow": 0,
    "fetchDynamicValues": true,
    "sortBy": [
        "string"
    ],
    "startRow": 0
}
```

Responses:

- 200: OK - Rest Fetch Response
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Fetch Records by Primary Key

GET

```
REST Endpoint: /rest/{table}/{pks}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Get multiple records by their primary keys.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Table** (*required*)	The table to look up	String	Content
Path	**Pks** (*required*)	The primary key(s) of the record(s) to look up	String (Comma separated list of pks)	10,42

Example:

```
http://slims.customer.com/slimsrest/rest/Content/10,42
```

Responses:

- 200: OK - Rest Fetch Response
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Update Multiple Records

> **POST**
>
> ```
> REST Endpoint: /rest/{table}
> Consumes: application/json
> Produces: */*
> ```

Implementation Notes:

Used to update multiple records at once.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Table** (*required*)	The table to update	String	Content
Body	**Records** (*required*)	Records to update	List[Object]	See below

Example:

```
[{
    "cntn_pk" : "200",
    "cntn_cf_testSpecificForContentA" : "test2223",
    "cntn_mass": {"amount": "12", "unit_pk": "6", "unit_display":"g"},
    "cntn_collectionDate":"2017-07-01T11:12:00.000000"
},
{
    "cntn_pk" : "201",
    "cntn_cf_testSpecificForContentA" : "test"
}]
```

> (Date format: yyyy-MM-dd'T'HH:mm:ss.SSSSSS)

Responses:

- 200: OK - Rest Fetch Response
- 201: Created
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Update a Record by Pk

POST

```
REST Endpoint: /rest/{table}/{pk}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Updates a single record.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Table** (*required*)	The table to update	String	Content
Path	**Pk** (*required*)	The primary key of the record to update	Integer(int64)	1
Body	**RecordToUpdate** (*required*)	Values of the record to update	Object	See below

RecordToUpdate example:

```
{
   "cntn_cf_testSpecificForContentA" : "test2223",
   "cntn_mass": {"amount": "12", "unit_pk": "6", "unit_display":"g"}
}
```

Responses:

- 200: OK - Rest Fetch Response
- 201: Created
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Add a Record

PUT

```
REST Endpoint: /rest/{table}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Creates a new record.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Table** (*required*)	The table to add a record to	String	Content
Body	**RecordToAdd** (*required*)	Record to be added	Object	See below

RecordToAdd Example:

```
{
"cntn_fk_contentType":"11",
"cntn_status":"10",
"cntn_cf_testSpecificForContentA" : "test2223",
"cntn_mass": {"amount": "12", "unit_pk": "6", "unit_display":"g"}
}
```

Responses:

- 200: OK - Rest Fetch Response
- 201: Created
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Add Multiple Records on Same Table

PUT

```
REST Endpoint: /rest/{table}/multiple
Consumes: application/json
Produces: */*
```

Implementation Notes:

Add multiple records on the same table.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Table** (*required*)	The table to add records to	String	Content
Body	**RecordsToAdd** (*required*)	Records to be added	List[Object]	See below

RecordsToAdd Example:

```
[
  {
    "cntn_fk_contentType":"11",
    "cntn_status":"10",
    "cntn_cf_testSpecificForContentA" : "test2223",
    "cntn_mass": {"amount": "12", "unit_pk": "6", "unit_display":"g"}
  },
  {
    "cntn_fk_contentType":"11",
    "cntn_status":"11",
    "cntn_cf_testSpecificForContentA" : "test2223",
    "cntn_mass": {"amount": "15", "unit_pk": "6", "unit_display":"g"}
  },
  {
    "cntn_fk_contentType":"11",
    "cntn_status":"10",
    "cntn_cf_testSpecificForContentA" : "test2223",
    "cntn_mass": {"amount": "11", "unit_pk": "6", "unit_display":"g"}
  }
]
```

Responses:

- 200: OK - Rest Fetch Response
- 201: Created
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

CONTENT LINKED IN ELN

Unlink Content from an Experiment Run Step

DELETE

```
REST Endpoint:
/rest/eln/content/{usage}/{experimentRunStepPk}/{contentPk}
```

```
Consumes: application/json
Produces: */*
```

Implementation Notes:

Unlinks a content record from an experiment run step.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Usage** (*required*)	Input or output	String	Input
Path	**experimentRunStepPk** (*required*)	The primary key of the experiment run step	Integer(int64)	1
Path	**contentPk** (*required*)	The primary key of the content	Integer(int64)	4

Example:

```
http://slims.customer.com/slimsrest/rest/eln/content/input/1/4
```

Responses:

- 200: OK
- 204: No Content
- 401: Unauthorized
- 403: Forbidden

Fetch Content Linked to an Experiment Run Step

GET

```
REST Endpoint: /rest/eln/content/{usage}/{experimentRunStepPk}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Fetch the content that is linked to an experiment run step.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Usage** (*required*)	Input or output	String	Input

Type	Name	Description	Datatype	Example
Path	**experimentRunStepPk** (*required*)	The primary key of the experiment run step	Integer(int64)	6

Example:

```
http://slims.customer.com/slimsrest/rest/eln/content/input/6
```

Responses:

- 200: OK - Rest Fetch Response
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Link Content to an Experiment Run Step

POST

```
REST Endpoint:
/rest/eln/content/{usage}/{experimentRunStepPk}/{contentPk}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Used to link a content to an Experiment Run Step.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Usage** (*required*)	Input or output	String	Output
Path	**experimentRunStepPk** (*required*)	The primary key of the experiment run step	Integer (int64)	7
Path	**contentPk** (*required*)	The primary key of the content	Integer (int64)	76

Example:

```
http://slims.customer.com/slimsrest/rest/eln/content/input/7/76
```

Responses:

- 200: OK - Rest Fetch Response
- 201: Created
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

CONTENT MIX OPERATION

Create a Mix of Ingredients

PUT

```
REST Endpoint: /rest/actions/content/mix
Consumes: application/json
Produces: */*
```

Implementation Notes:

Create a mix of selected ingredients.

Parameters:

Type	Name	Description	Datatype	Example
Body	**mixData** (*required*)	Mix values and ingredient pks	{List[Integer (int64)], List[Object]}	See below

mixData Example:

```
{
  "ingredientPks": [
    0
  ],
  "mixValues": {
    "additionalProp1": {},
    "additionalProp2": {},
    "additionalProp3": {}
  }
}
```

Responses:

- 200: OK - Rest Fetch Response
- 201: Created

- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Electronic Lab Notebook (ELN)

Fetch Root of ELN Tree

GET

```
REST Endpoint: /rest/eln
Consumes: application/json
Produces: */*
```

Implementation Notes:

Fetch the root of the Electronic Lab Notebook tree.

Responses:

- 200: OK - Rest Fetch Response
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Fetch Children of a Record in ELN Tree

GET

```
REST Endpoint: /rest/eln/{table}/{pk}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Fetch the children of a record in the Electronic Lab Notebook tree.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Table** (*required*)	The table of the parent record	String	Project
Path	**Pk** (*required*)	The primary key of the parent record	Integer (int64)	1

Example:

```
http://slims.customer.com/slimsrest/rest/eln/Project/1
```

Responses:

- 200: OK - Rest Fetch Response
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Add a Simplified Block to an Experiment Run

POST

```
REST Endpoint: /rest/eln/ExperimentRun/{pk}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Add a simplified block to an experiment run.

Parameters:

Type	Name	Description	Datatype	Example
Path	**Pk** (*required*)	The primary key of the experiment run	Integer (int64)	4
Body	**blockValues** (*optional*)	The simplified block parameters (xprs_name, xpst_type should be given)	Object	See below

blockValues Example:

```
{
    "xprs_name": "",
    "xpst_type": "",
    "additionalProp": {}
}
```

Responses:

- 200: OK - Rest Fetch Response
- 201: Created
- 401: Unauthorized
- 403: Forbidden

- 404: Not Found

FILE REPOSITORY OPERATIONS

Download the Attachment

> **GET**

```
REST Endpoint: /rest/repo/{pk}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Download an attachment.

Parameters:

Type	Name	Description	Datatype	Default
Path	**Pk** (*required*)	pk of the attachment to download	Integer (int64)	7

Responses:

- 200: OK
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Upload a New Attachment

> **POST**

```
REST Endpoint: /rest/repo
Consumes: application/json
Produces: */*
```

Implementation Notes:

Upload an Attachment.

Parameters:

Type	Name	Description	Datatype	Example

Type	Name	Description	Datatype	Example
Body	**attachmentValues** (*required*)	Values of the attachment to add	Object	See below
	atln_recordPk (*optional*)	The primary key of the record to attach to	Integer (int64)	1
	atln_recordTable (*optional*)	The name of the table of the record to attach to	String	Content

attachmentValues Example:

```
{
  "atln_recordPk": 1,
  "atln_recordTable": "Content",
  "attm_name": "myname.txt",
  "contents": "This is the content of the file"
}
```

Responses:

- 200: OK - Rest Fetch Response
- 201: Created
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Plugin Operations

```
Plugin Operations are only relevant for SLIMS instances that use plugins!
```

Upload or Replace a Plugin

POST

```
REST Endpoint: /rest/plugin
Consumes: application/json
Produces: */*
```

Implementation Notes:

Upload or replace a SLIMS Plugin via REST

Parameters:

Type	Name	Description	Datatype	Example
Body	**pluginValues** (required)	Values for the plugin	Object	See below

pluginValues Example:

```
{
  "contents": "pluginContent",
  "name": "pluginName",
  "type": "SLIMS"
}
```

Responses:

- 200: OK - Rest Fetch Response
- 201: Created
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Queue

Fetch Record in Queue in a Certain Status

GET

```
REST Endpoint: /rest/queue/content/{queue}/{status}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Allows users to fetch the samples in a queue that are in a certain status. The possible statuses are: NOT_STARTED, IN_PROGRESS, APPROVAL, DONE, and CANCELLED. For example, you can fetch all content on the "Start" queue that are not in an experiment yet by querying /queue/content/Start/NOT_STARTED

Parameters:

Type	Name	Description	Datatype	Example
Path	**Queue** (*required*)	The queue to look in	String	Start
Path	**Status** (*required*)	The status of the elements in the queue	String	NOT_STARTED

Example:

```
http://slims.customer.com/slimsrest/rest/queue/content/Start/NOT_STARTED
```

Responses:

- 200: OK - Rest Fetch Response
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Health-Controller

Prometheus monitoring

POST

```
REST Endpoint: /rest/health/metrics/prometheus/{service}
Consumes: application/json
Produces: */*
```

Implementation Notes:

Scrapes metrics data for a Prometheus monitoring system

Parameters:

Type	Name	Description	Datatype	Example
Path	**service** (required)	Name of the service	String	

Responses:

- 200: OK - Rest Fetch Response
- 201: Created
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found
