1spatial
YOUR WORLD SMARTER

1Integrate

REST API Guide

Product version: v 2.8.1

Document version: v 1.33

Document date: 10/03/2021

Copyright 2021 1Spatial plc and its affiliates.

All rights reserved. Other trademarks are registered trademarks and the properties of their respective owners.

No part of this document or any information appertaining to its content may be used, stored, reproduced or transmitted in any form or by any means, including photocopying, recording, taping, information storage systems, without the prior permission of 1Spatial plc.

1Spatial
Tennyson House
Cambridge Business Park
Cambridge
CB4 0WZ
United Kingdom

Phone: +44 (0)1223 420414

Fax: +44 (0)1223 420044

Web: www.1spatial.com

Every effort has been made to ensure that the information contained in this document is accurate at the time of printing. However, the software described in this document is subject to continuous development and improvement. 1Spatial plc reserves the right to change the specification of the software. 1Spatial plc accepts no liability for any loss or damage arising from use of any information contained in this document.

Contents

1 1Integrate REST API 5

Notation 5

Defaults 5

2 Authentication 6

3 Basic Operations 8

Folder Listing 8

Retrieving Resources 9

Creating and Updating Resources 9

Creating Folders or Updating Folder Metadata 9

Deleting Resources 10

Cloning Resources 10

4 Resources 12

Common 12

5 Sessions 13

Session Properties 15

Session Extents 16

Session Control 18

Check Session Status 18

Session Summary Results 19

Tasks 21

OpenData 21

BuildTopology 24

CheckRules 25

ApplyActions 25

ApplyActionMap 26

CopyTo 26

Commit 28

Pause 28

6 Data Stores 29

Data store Resource Properties 30

Data store Creation from File Upload 33

Providing Export Mapping Files 36

Data Store Creation from Credentials 36

Credentials Properties 39

MapInfo Tab (GDAL) 39

Oracle 40
Esri Shape 42
Esri File Geodatabase (FILEGDB) 45
PostGIS 46
Other Data Store Formats 50
Password Properties 50
Automatic Schema Derivation 50
Import Schema Derivation 50
Export Schema Derivation 51
Schema Mapping 51
Test and Refresh Mapping 53
User Defined Classes 54
CopyTo Task Output File Download 57

7 Rules 58

8 Actions 60

Action Maps 61

9 Results 62

Task Summary 62
Detailed Non-conformance or Error Reports 63
Summarised Error Reports 64
Summarised Folder Results 64

1 1Integrate REST API

The REST API provides a simple integration point for 1Integrate. This allows resources (datastores, rules, actions, action maps, sessions) within the application to be created, edited and deleted in a similar manner to the main user interface. There are also endpoints for controlling a session (run, pause, stop), and getting session results (non-conformances, errors).

The REST API has been designed to be used with any client language or http client library.

The API requires authorisation headers and utilises a number of different request types such as POST and PUT, so responses cannot be tested using a web browser.

If you try and request a resource such as `http://[server]:[port]/1Integrate/rest/datastores` from a web browser, then the response will be an **HTTP 401 (unauthorised)** error.

Note: Not all browsers will display the fact that an error has occurred.

To test the URL for the API connection it is recommended that you use your chosen REST client.

Note: This document assumes knowledge of 1Integrate concepts; the concepts and capabilities are not described in detail within this document. For more information see the 1Integrate WebHelp.

Notation

Within this document, the following colour-coding is used to demonstrate API requests and responses:

```
Green boxes represent example requests
```

```
Grey boxes represent example properties that can be passed in the request
```

```
Purple boxes represent example responses
```

Defaults

Unless stated otherwise, both the Accept and Content-Type headers for all requests must be set to `application/json`.

2 Authentication

All calls to the rest service must be authenticated by using a JSON Web Token.

Tokens can be generated via a POST to the token service using an appropriate username and password. Note that the username must have the rswsuser role in order to access the API, an example request and example authorization header is show below:

```
POST http://[server]:[port]/1Integrate/rest/token
```

```
{"username":"user1", "password":"password1"}
```

This will return an *Authorization* header:

```
Authorization:
eyJraWQiOiJrMSIsImFsZyI6IlJTMjU2In0.eyJpc3MiOiIxU3BhdGl
hbCIsImV4cCI6MTUwMTI1Mjk4OCwianRpIjoiSjRXcGNYZVNtQzJ3aF
VZMU9femhhdyIsImlhdCI6MTUwMTI0NTc4OCwic3ViIjoiMVNwYXRpY
WwiLCJyb2xlcyI6WyJyc19hZG1pbnMiLCJyc191c2VycyIsInJzd3N1
c2VyIl0sInJlbWVtYmVyIjpudWxsfQ.foy1N1kCuQjk7zjgcqilJoxQ
xp6DQsO3FYrJs8Le79wQ3JPE6onTmz_
X6DQxfjVyL9r9SSIgfPxzTrUt-04PQFvbjsVr_pbCBhLYaDr_
luTnzQ0OrVZJEt9Avy2gRgvGmYhVycKOHpn0ZVQKwZAt_
hJLkLUczsUR2AulxCYxITDotXte3j5Vy7ZhQRcJ4Eq-
VNtSlRy6kYzlNAF-F_JpgEh5RNCDKIxyvbZj1R4jJSBWX_mOT7_
coFuqpSyyTZcUfUvCo5NqFEZ-y0sXHzEVwSHxA24iyiJHg_
U7NE9weEtcGgKzLI7vZgkxEJ8oNxw5VSxvbNNtEMlAtBcMf-9bBA
```

Any subsequent API calls must have an Authorization entry in the header with the value of 'Bearer' followed by a space, followed by the token returned in the Authorization header of the call to the token service, as in the following example:

```
Authorization: Bearer
eyJraWQiOiJrMSIsImFsZyI6IlJTMjU2In0.eyJpc3MiOiIxU3BhdGl
hbCIsImV4cCI6MTUwMTI1Mjk4OCwianRpIjoiSjRXcGNYZVNtQzJ3aF
VZMU9femhhdyIsImlhdCI6MTUwMTI0NTc4OCwic3ViIjoiMVNwYXRpY
WwiLCJyb2xlcyI6WyJyc19hZG1pbnMiLCJyc191c2VycyIsInJzd3N1
c2VyIl0sInJlbWVtYmVyIjpudWxsfQ.foy1N1kCuQjk7zjgcqilJoxQ
xp6DQsO3FYrJs8Le79wQ3JPE6onTmz_
X6DQxfjVyL9r9SSIgfPxzTrUt-04PQFvbjsVr_pbCBhLYaDr_
luTnzQ0OrVZJEt9Avy2gRgvGmYhVycKOHpn0ZVQKwZAt_
hJLkLUczsUR2AulxCYxITDotXte3j5Vy7ZhQRcJ4Eq-
VNtSlRy6kYzlNAF-F_JpgEh5RNCDKIxyvbZj1R4jJSBWX_mOT7_
coFuqpSyyTZcUfUvCo5NqFEZ-y0sXHzEVwSHxA24iyiJHg_
U7NE9weEtcGgKzLI7vZgkxEJ8oNxw5VSxvbNNtEMlAtBcMf-9bBA
```

The token needs to be added to the header with a 'Bearer' keyword, for example in Python:

```
url = 'http://localhost:18080/1integrate/rest/%

tokenResponse = requests.post(url % 'token', json=
```

```
{'username':'<user>','password': '<password>'}
tokenResponse.raise_for_status()
token = 'Bearer' + tokenResponse.headers'Authorization'
      #Create a requests session to avoid having to specify headers on each
call.
      requestSession = requests.Session()
      requestSession.headers.update( {'Authorization': token, 'Accept':
'Application/JSON',
      'Content-Type': 'Application/JSON'}
      )
```

An example in PowerShell script:

```
$Body = '{"username":"'+$Username+'", "password":"'+$Password+'"}'
$Response = Invoke-WebRequest -Uri
'http://<server>:8080/1Integrate/rest/token' -Method Post -Body $Body
# Read the token from the header
$Token = $Response.Headers.Authorization
# Now use the token returned in the header when sending requests
# Note, need to add 'Bearer ' before the token in the authorization header
$Headers = @{}
$Headers.Add("Authorization", "Bearer " + $Token)
```

By default the token will last 2 hours. When you use a token to access the 1Integrate REST API and the server detects that your token is about to expire, it will refresh your token and send the new token back with the response, again in the *Authorization* response header.

If a longer lasting token is required, a 2 week long token can be created using the following request:

```
POST http://[server]:
[port]/1Integrate/rest/token?rememberMe=true
```

3 Basic Operations

The API is structured following the folder structure of resources within the main application.

Folder Listing

A GET request to the path of a folder will retrieve the folder metadata and a listing of children of the folder.

```
GET http://[server]:[port]/1Integrate/rest/datastores
GET http://[server]:[port]/1Integrate/rest/rules
GET http://[server]:[port]/1Integrate/rest/actions
GET http://[server]:[port]/1Integrate/rest/actionmaps
GET http://[server]:[port]/1Integrate/rest/sessions
```

```
{
 "createdBy": "Radius Studio",
 "updated": 1487954201386,
 "created": 1487954201386,
 "contents": [
  {
   "isEmpty": false,
   "name": "Recycle Bin",
   "type": "folder"
  },
  {
   "name": "Empty Test",
   "type": "folder"
  },
  {
   "name": "New Session 1",
   "type": "paused_session"
  }
 ]
}
```

Retrieving Resources

Resources can be retrieved with a GET request to the full resource path.

```
GET http://[server]:[port]/1Integrate/rest/datastores/{path}
```

> **Note:** The easiest way to find the structure of the JSON for creating or updating items is to create them in the user interface and then GET them to inspect their structure.

The path to a resource is the folder structure from 1Integrate for the particular resource type. For example, for a Session called "A" that is within the folder structure "/X/Y/" then the path to that session is `http://[server]:[port]/1Integrate/rest/sessions/X/Y/A`.

> **Note:** Session results are accessed from the results resource, using the path to the relevant session, and not from the sessions resource.

Creating and Updating Resources

Resources are created or updated through PUT requests. These should contain the full JSON of the resource to be created.

If the parent folders in the request do not exist they will be created automatically.

For examples of each type of resource, see "Resources" on page 12.

Creating Folders or Updating Folder Metadata

Folders can be created, or folder metadata (description, comments) can be updated using a PUT request to the folder path, with a query parameter `folder=true`.

```
PUT http://[server]:[port]/1Integrate/rest/sessions/{path}?folder=true
```

```
{
 "description": "test description",
 "comments": "test comments"
}
```

```
{
 "comments": "test comments",
```

```
 "created": 1501238675855,
 "createdBy": "1Spatial",
 "description": "test description",
 "updated": 1501238675855,
 "updatedBy": "1Spatial"
}
```

Deleting Resources

Resources or folders can be deleted with a DELETE request.

Deleting a folder will recursively delete all items contained within the folder.

```
DELETE http://[server]:[port]/1Integrate/rest/sessions/{path}
```

Cloning Resources

Resources or folders can be cloned with a PUT request.

```
PUT http://[server]:[port]/1Integrate/rest/clone
```

```
[
{
"source" : [Item to be copied]
"destination" : [Fully qualified path to copy item or name]
}
]
```

Note:
"destination" is an optional property and can be defined as a path, a name or left blank:

- Defining a path will place a clone of the original resource in the specified location, the name of the original resource will be maintained.
- Defining a name alone will place the clone in the same location as the original resource.
- Leaving "destination" blank or omitting it entirely will mean the clone is placed in the same location as the original resource, with a name following the "Copy of [Name]" convention in the UI.

When requesting a single entity an example response would be:

```
[
```

```
{
"clone": "[cloned resource name/location]"
}
]
```

When requesting multiple entities, responses will return as follows:

```
[
{
"clone": "[location of clone if successful]",
"status": 200
},
{
"errorMessage": "[error message]",
"status": [status code of the error (e.g. 404 for not
found)]
]
}
```

4 Resources

The following types of resource can be managed within the REST API:

- Connections to datastores
- Rules
- Actions
- Action Maps
- Sessions (and Tasks)

Common

All resource types have the following common contents.

Note: Items may be omitted when they are default or not specified.

```
"createdBy": "1Spatial",
"updated": 1488809081647,
"created": 1488809081647,
"version": 1,
"updatedBy": "1Spatial"
"description": ""
"comments": ""
```

The creation and update timestamps and users are maintained automatically. They do not need to be specified for create or update operations, but will always be returned from the service.

The `version` property is used for optimistic locking to avoid concurrent users accidentally making conflicting edits. If multiple requests are made to update a resource with the same version then they will be rejected.

Note: If you get a response with status `409 Conflict` when trying to update a resource, this means someone else has updated the resource before you. Perform a GET to get the latest version of the resource and retry the update using that version.

5 Sessions

Sessions contain a list of tasks, with varying content depending on the kind of task(s) they contain (see "Tasks" on page 21).

Responses containing session resources will include the common resource properties (see "Common" on page 12), as well as other session-specific properties (see "Session Properties" on page 15).

Note: Session results are accessible from the results resource, not the sessions resource (see below).

Sessions can also be created or updated via `PUT` requests.

```
GET http://[server]:[port]/1Integrate/rest/sessions/{path}
```


```
{
 "comments": "Comments",
 "created": 1501237189979,
 "createdBy": "1Spatial",
 "description": "",
 "updated": 1501237290578,
 "updatedBy": "1Spatial",
 "tasks": [
  {
   "kind": "OpenData",
   "datastore": "/datastores/ds1",
   "classes": [
    {
     "name": "english"
    },
    {
     "disabled": true,
     "name": "luttuce"
    }
   ],
   "parameters": {},
   "topologyEnabled": true
  },
  {
```

```
    "kind": "CheckRules",
    "rules": [
     "/rules/rule1",
     "/rules/subfolder2/"
    ]
   },
   {
    "kind": "ApplyActionMap",
    "actionmap": "/action_maps/actionmap1"
   },
   {
    "kind": "ApplyActions",
    "actions": [
     "/actions/action1",
     "/actions/subfolder1/"
    ]
   },
   {
    "kind": "CopyTo",
    "datastore": "/datastores/ds1",
    "parameters": {}
   },
   {
    "kind": "BuildTopology",
    "model": "NETWORK",
    "snappingType": "SHARE_NODES",
    "tolerance": 0.042,
    "classes": [
     "english"
    ]
   },
   {
    "kind": "Pause"
   }
 ],
 "status": "NOT_STARTED",
 "version": 4,
 "extentAsBoundingBox": {
  "maxX": 10000,
```

```
  "maxY": 10000,
  "minX": 0,
  "minY": 0
 }
}
```

Session Properties

Property Name	Data Type	Optionality	Description
status	*(Read only)*	*(Read only)*	One of: ▶ NOT_STARTED ▶ RUNNING ▶ PAUSED ▶ FINISHED ▶ WAITING ▶ PAUSING ▶ REWINDING ▶ UNKNOWN ▶ STOPPING
tasks	Array of task objects (See "Tasks" on page 21)	Optional	Tasks to be run in the session.

Property Name	Data Type	Optionality	Description
runAsMultiplePartitions	Boolean	Optional (default: false)	True indicates the extent will be partitioned, and a separate session will be run for each partition. Only valid for 'Several Predefined Regions' sessions – see Session Extents section. Conversely, this must be false for all other extent types. Equivalent to UI checkbox 'Run as multiple partitions'.
extentBuffer	Double	Optional (default: 0)	A Buffer can be set to consider data in a region that is larger than the selection. The buffer value is in dataset units.
extentAsBoundingBox	See "Session Extents" below	Optional (See "Session Extents" below)	See "Session Extents" below
extentAsPolygon	String		
extentFromDatastore	See "Session Extents" below		

Session Extents

The extent to be used in a session can be determined using the same extent types available in the 1Integrate UI, by passing the properties

(`extentAsBoundingBox`, `extentAsPolygon` and `extentFromDatastore`).

> **Note:** The properties `extentAsBoundingBox`, `extentAsPolygon` and `extentFromDatastore` are mutually exclusive; requests using more than one of these properties will be refused.

Extent Type	How to achieve this in the REST API
All Data	Do not provide any extent properties in the request.

Bounding Box	Provide an extent in terms of minimum and maximum X and Y values using the `extentAsBoundingBox` property.
	`"extentAsBoundingBox":`
	`{`
	`"maxX": 10,`
	`"maxY": 10,`
	`"minX": 0,`
	`"minY": 0`
	`}`

One Pre-defined Region	Provide an `extentFromDatastore` property with `attributeName` populated.
	`"extentFromDatastore":`
	`{`
	`"attributeName": "ID",`
	`"attributeValue": "1",`
	`"className": "ROADS",`
	`"datastore": "/datastores/TestDatastore"`
	`}`
	These values correspond to the UI fields as follows:
	▶ Table is `className`
	▶ Where is `attributeName`
	▶ Equals is `attributeValue`

Extent Type	How to achieve this in the REST API
Several Pre-defined Regions	Provide an `extentFromDatastore` property without an `attributeName` populated. Requires the additional property `runAsMultiplePartitions` to be set to `true`. `"extentFromDatastore": { "className": "ROADS", "datastore": "/datastores/TestDatastore" }, "runAsMultiplePartitions": true` These fields correspond to the UI fields: ▶ Table is `className`
Polygon	Provide an `extendAsPolygon` property. This is Well-Known text in a string, and must be a polygon. `"extentAsPolygon": "POLYGON((0 0, 0 10, 10 10, 0 0)) "`

Session Control

Sessions can be controlled via `POST` requests.

```
POST http://[server]:[port]/1Integrate/rest/sessions/{path}?action=play
POST http://[server]:[port]/1Integrate/rest/sessions/{path}?action=pause
POST http://[server]:[port]/1Integrate/rest/sessions/{path}?action=stop
POST http://[server]:[port]/1Integrate/rest/sessions/{path}?action=rewind&taskIndex=
```

The response for these requests will just be status code 202.

Check Session Status

The status and results for a session are available from results resource that is parallel to the session resource. The same session path is used to access the results for the session. For example, if a session resource is at:

```
http://[server]:[port]/1Integrate/rest/sessions/Folder1/MySession
```

Then, the results are available from:

```
http://[server]:
[port]/1Integrate/rest/results/Folder1/MySession
```

You can check the status of a session using a `GET` request.

```
GET http://[server]:[port]/1Integrate/rest/results/
{path}?detail=status
```

```
{
 "status": "PAUSED"
}
```

See "Session Properties" on page 15 for a list of possible values of the `status` property.

Session Summary Results

Return the summary report of a session with all its tasks.

> **Note:** `taskIdentifier` is a string used to uniquely identify the task within a session. `customLabel` is optional string that can be assigned to a task to make it more recognisable to users.

```
GET http://[server]:[port]/1Integrate/rest/results/
{path}
```

```
{
 "status": "PAUSED",
 "tasks": [
  {
   ";": 0,
   "count": 0,
   "customLabel": "Example Label 1",
   "duration": 1257,
   "errors": 0,
   "kind": "OpenDataTask",
   "processed": 3500,
   "reportCount": 0,
   "started": 1493119229050,
   "status": "FINISHED",
   "taskIdentifier": "4d7b4474-8095-4322-a554-
70abef37ac90"
  },
  {
   ";": 0,
   "count": 0,
```

```
    "customLabel": "Example Label 2",
    "duration": 73,
    "errors": 0,
    "kind": "PauseTask",
    "processed": 0,
    "reportCount": 0,
    "started": 1493119230325,
    "status": "FINISHED"
    "taskIdentifier": "12dca560-45fc-4619-8c4e-
693b34a5959a"
   },
   {
    "count": 246,
    "duration": 484,
    "errors": 0,
    "kind": "CheckRulesTask",
    "processed": 500,
    "reportCount": 246,
    "started": "1493119232968",
    "status": "FINISHED",
    "taskIdentifier": "e3d5a60b-f454-92b9-
7c2cba568e65",
    "total": 500
   },
   {
    "count": 0,
    "duration": 16,
    "errors": 0,
    "kind": "PauseTask",
    "processed": 0,
    "count": 0,
    "started": 1493119233533,
    "status": "PAUSED",
    "taskIdentifier": "2ad248e5-722274be9-95a9-
dd7b00c273f8",
    "total": 0
   }
  ]
}
```

Tasks

Tasks are managed through sessions (see "Sessions" on page 13), and are placed in a sequence to determine the order in which they should be performed.

The `kind` property is common to all tasks types, and determines other configuration options for the task.

Note: `taskIdentifier` is a string used to uniquely identify the task within a session. `customLabel` is an optional string that can be assigned to a task to make it more recognisable to users. `id` is a string that uniquely identifies the task for the session (only visible from "session" GET requests, not in "results".

The following types of task can be included in a session:

- OpenData
- BuildTopology
- CheckRules
- ApplyActions
- ApplyActionMap
- CopyTo
- Commit
- Pause

Note: Once a task has run, you can return the summary of its results using a GET request (see "Task Summary" on page 62).

OpenData

```
{
 "kind": "OpenData",
 "datastore": "/datastores/path/to/MyDatastore",
 "restrictToOpenedDataExtent": "[true/false]"
 "openedDataBufferDistance": "[XX Dataset Units]"
 "classes" :
 [
  {
   "disabled": true,
   "name": "ROAD"
  },
```

```
    {
      "name": "RAILWAY"
    }
  ],
  "parameters":
  {
    "someUnusedParameter": 0
  },
  "topologyEnabled": true;
}
```

Property Name	Data Type	Optionality	Description
datastore	String	Mandatory	Resource path of the datastore to connect to.
restrictToOpenedDataExtent	Boolean	Optional	Enable or disable to restrict the opening of data to classes selected in an Open Data task
openedDataBufferDistance	Integer	Optional	A buffer in dataset units around the extent opened by `restrictToOpenedDataExtent`

Property Name	Data Type	Optionality	Description

classes	Array of objects	Optional	Enable or disable specific classes during load.
			Not providing this property will load from *all* classes as configured in the datastore.
			Each element of the array takes the form:
			`{ "disabled": true, "name": "className" }`
			`disabled` defaults to `false`, so specifying the name of a class is enough to enable it. Both of the below are equivalent:
			`{ "name": "className" }`
			`{ "disabled": false, "name": "className" }`
			If a class is configured in the datastore but not the `classes` array, it is effectively disabled.

parameters	Object	Optional	Used for some niche functionality, such as connecting to a specific database version or workspace.
topologyEnabled	Boolean	Optional (default: false)	Equivalent to the UI checkbox **Enable Topology**.

BuildTopology

```
{
 "kind": "BuildTopology",
 "model": "NETWORK",
 "snappingType": "SHARE_NODES",
 "tolerance": 0.5
}
```

Property Name	Data Type	Optionality	Description
model	String	Mandatory	Valid inputs are `NETWORK` or `PLANAR`. Equivalent to UI field **Model**.
snappingType	String	Mandatory	If model is NETWORK, can be one of: `SHARE_NODES`, `NODES_SPLIT_EDGES` or `EDGES_SPLIT_EDGES`. If model is PLANAR, must be `EDGES_SPLIT_EDGES`. Equivalent to UI field **Snapping type**.
tolerance	Double	Optional (default: 0)	Topological tolerance to use during structuring. A value of 0 will instead derive a tolerance from loaded data.
classes	Array of Strings	Optional	Specify classes to be topologically structured. Omitting this parameter will not structure any classes.

CheckRules

```
{
 "kind": "CheckRules",
 "rules":
 [
  "/rules/path/to/MyRule",
  "/rules/path/to/MyOtherRule",
  "/rules/path/to/MyRuleFolder"
 ],
 "filterRule": "/rules/path/to/MyRule"
}
```

Property Name	Data Type	Optionality	Description
rules	Array of Strings	Optional	Array of rule and/or rule folder resource paths.
filterRule	String	Optional	Specify a filter rule.

ApplyActions

```
{
 "kind": "ApplyActions",
 "actions":
 [
  "/actions/path/to/MyAction",
  "/actions/path/to/MyOtherAction",
  "/actions/path/to/MyActionFolder"
 ],
 "filterRule": "/rules/path/to/MyRule"
}
```

Property Name	Data Type	Optionality	Description
actions	Array of Strings	Optional	Array of action and/or action folder resource paths.
filterRule	String	Optional	Specify a filter rule.

ApplyActionMap

```
{
 "kind": "ApplyActionMap",
 "actionmap": "/action_maps/path/to/MyActionMap",
 "filterRule": "/rules/path/to/MyRule"
}
```

Property Name	Data Type	Optionality	Description
actionmap	String	Mandatory	Resource path of an action map.
filterRule	String	Optional	Specify a filter rule.

CopyTo

Note: CopyTo properties are the same as the OpenData properties, but without `topologyEnabled`.

```
{
 "kind": "CopyTo",
 "datastore": "/datastores/path/to/MyDatastore",
 "classes" :
 [
  {
   "disabled": true,
   "name": "ROAD"
  },
  {
   "name": "RAILWAY"
  }
 ],
 "parameters":
 {
  "someUnusedParameter": 0
 }
}
```

Property Name	Data Type	Optionality	Description
datastore	String	Mandatory	Resource path of the datastore to connect to.

classes	Array of objects	Optional	Enable or disable specific classes during write.
			Not providing this property will write to *all* classes as configured in the datastore.
			Each element of the array takes the form:
			`{ "disabled": true, "name": "className" }`
			`disabled` defaults to `false`, so specifying the name of a class is enough to enable it. Both of the below are equivalent:
			`{ "name": "className" }`
			`{ "disabled": false, "name": "className" }`
			If a class is configured in the datastore but not the `classes` array, it is effectively disabled.

parameters	Object	Optional	Used for some niche functionality, such as connecting to a specific database version or workspace.

Commit

> **Note:** Commit properties are the same as the CopyTo properties, but without `classes`.

```
{
 "kind": "Commit",
 "datastore": "/datastores/path/to/MyDatastore",
 "parameters":
 {
  "someUnusedParameter": 0
 }
}
```

Property Name	Data Type	Optionality	Description
datastore	String	Mandatory	Resource path of the datastore to connect to.
parameters	Object	Optional	Used for some niche functionality, such as connecting to a specific database version or workspace.

Pause

A Pause task is used to pause a session once it is running. It has no configurable properties.

```
{
 "kind": "Pause"
}
```

6 Data Stores

The full specification of a data store within 1Integrate is often a complex structure encapsulating an entire data model. The REST API provides a number of convenient methods for constructing a data store.

Responses containing datastore resources will include the common resource properties (see "Common" on page 12), as well as other datastore-specific properties (see "Data store Resource Properties" on the next page).

Data Stores can also be created or updated via `PUT` requests.

```
GET http://[server]:[port]/1Integrate/rest/datastores/
{path}
```

```
"createdBy": "1Spatial",
"updated": 1488809081647,
"created": 1488809081647,
"version": 1,
"updatedBy": "1Spatial",
"description": "",
"comments": ""
```

Data store Resource Properties

Property Name	Data Type	Optionality	Description
importType	String	Mandatory	Defines the datastore type to use for data reading. Can take the following values: ▶ Autodesk AutoCAD DWG/DXF ▶ Bentley MicroStation Design (V8) ▶ Comma Separated Value (CSV) ▶ Esri ArcGIS Server ▶ Esri Enterprise Geodatabase ▶ Esri File Geodatabase (GDAL) ▶ Esri Geodatabase File (FILEGDB) ▶ Esri Shape ▶ ESRI Shape (GDAL) ▶ MapInfo Tab (GDAL) ▶ Microsoft SQL Server Spatial ▶ PostGIS ▶ Oracle

Property Name	Data Type	Optionality	Description
			Note: Be careful to use the exact spelling and capitalisation as given .
importCredentials	Credentials object	Optional (default: empty object)	Valid credentials depend on `importType`. **Note:** All values are Strings, even if the value is really numeric or Boolean.
importClasses	Array of Class objects	Optional (default: null)	Import schema and mappings.
exportType	String	Optional (default: “New Export Credentials”)	Defines the datastore to use for data writing. Equivalent to the UI drop-down box **Data Store Type** on the Output Details tab.
exportCredentials	Credentials object	Optional (default: null)	Valid credentials depend on `exportType`. **Note:** All values are Strings, even if the value is really numeric or Boolean.

Property Name	Data Type	Optionality	Description
exportClasses	Array of Class objects	Optional (default: null)	Export schema and mappings.
userDefinedClasses	Array of Class objects	Optional (default: null)	User-defined schema and mappings.

Data store Creation from File Upload

Files can be uploaded to a datastore with a POST operation supplied with `Content-Type: application/octet-stream`, and the file (usually .zip) in the body.

Note: Any format defined by a single file (e.g. CSV or DWG) can be uploaded without being zipped (but zipped files are also supported). Any format defined by multiple files or folders (e.g Esri Shapefile, Esri FGDB) must *always* be uploaded as a single zip file.

```
POST http://[server]:[port]/1Integrate/rest/datastores/{path}
```

If the file is uploaded successfully, its schema will be populated and returned in the `importClasses` section. If the correct `exportType` and `exportCredentials` are provided, then the input schema will be copied and reversed to create the `exportClasses`, which will also be returned.


```
{
 "created": 1501246885833,
 "createdBy": "1Spatial",
 "updated": 1501246885833,
 "updatedBy": "1Spatial",
 "importType": "Esri Shape",
 "importCredentials": {
  "Coordinate Reference System": "",
  "_connection_use": "import",
  "Clip to envelope": "false",
  "Allow invalid geometries": "true",
  "Encoding": "",
  "Reverse coordinate axis order (y,x)": "false",
  "Trim preceding spaces": "true",
  "Dissolve holes": "true",
```

```
  "Import FME Log File": "",
  "Convert attribute names to upper case": "false",
  "Exposed attributes": "",
  "Fix ring direction and inclusion errors": "true",
  "Source Files (.shp)": "Glaisnock.zip",
  "Treat measures as elevation": "false"
},
"importClasses": [
  {
   "attrs": [
    {
     "dataType": "fme_varchar",
     "mappedType": "String",
     "name": "ID",
     "params": {
      "length": "254"
     }
    },
    {
     "dataType": "fme_geometry{0}",
     "mappedType": "Geometry",
     "name": "geometry",
     "params": {
      "dimension_count": "2",
      "complex": "false",
      "type": "line",
      "unit_length": "1.0",
      "space": "British National Grid (ORD SURV GB)",
      "srid": "FME=_BritishNatGrid_0,EPSG=27700"
     }
    }
   ],
   "name": "INFSEWER"
  }
],
"exportType": "MapInfo TAB (MFAL)",
"exportCredentials": {
  "Coordinate Reference System": "",
  "_connection_use": "export",
```

```
  "Destination Files (.tab)": "",
  "Write region centroids": "false",
  "Export FME Log File": "",
  "Filename prefix": "",
  "Encoding": "",
  "Reverse coordinate axis order (y,x)": "false",
  "Bounds ([xmin] [ymin] [xmax] [ymax])": ""
 },
 "exportClasses": [
  {
   "attrs": [
    {
     "dataType": "fme_varchar",
     "name": "ID",
     "params": {
      "length": "254"
     }
    },
    {
     "dataType": "fme_geometry{0}",
     "name": "geometry",
     "params": {
      "dimension_count": "2",
      "complex": "false",
      "type": "line",
      "unit_length": "1.0",
      "space": "British National Grid (ORD SURV GB)",
      "srid": "FME=_BritishNatGrid_0,EPSG=27700"
     }
    }
   ],
   "name": "INFSEWER"
  }
 ],
 "version": 4
}
```

The following properties are common to all "File Upload" datastore types (see "Credentials Properties" on page 39 for individual datastore credential properties):

Property Name	Data Type	Optionality	Description
Coordinate Reference System	String		Defines the coordinate system of the source data. An empty string indicates this should be inferred from the data. See the 1Integrate WebHelp for more details on the format to specify the coordinate system.
Source Files (…)	File path	Mandatory (and must be non-empty) before uploading data; optional afterwards, as long as no more data is to be uploaded.	Formats vary by datastore (e.g. “Source Files (.shp)”. This is an internally used label for the uploaded data which is placed into the repository. The UI uses the path of the uploaded file as the label. Re-using the same label tells the datastore that it can re-use the already uploaded data.

Providing Export Mapping Files

You may be required to provide a file as an Input Parameter value for the export mapping of a datastore (e.g if you are using BigQuery).

```
POST http://[DatabasePath]/exportFile
```

Data Store Creation from Credentials

Note: If credentials are supplied without a schema, 1Integrate can automatically connect to the datasource, fetch the schema and generate a default mapping (see "Automatic Schema Derivation" on page 50). This works for any combination of import credentials and/or export credentials.

Specific credentials to be supplied depend on the type of datastore required (see "Credentials Properties" on page 39).

```
PUT http://[server]:[port]/1Integrate/rest/datastores/
{path}
```

```
{
 "importType": "Esri Shape",
 "importCredentials": {
  "Coordinate Reference System": "",
  "_connection_use": "import",
  "Clip to envelope": "false",
  "Allow invalid geometries": "true",
  "Encoding": "",
  "Reverse coordinate axis order (y,x)": "false",
  "Trim preceding spaces": "true",
  "Dissolve holes": "true",
  "Import FME Log File": "",
  "Convert attribute names to upper case": "false",
  "Exposed attributes": "",
  "Fix ring direction and inclusion errors": "true",
  "Source Files (.shp)": "Glaisnock.zip",
  "Treat measures as elevation": "false"
 },
 "exportType": "MapInfo TAB (MFAL)",
 "exportCredentials":{
  "Coordinate Reference System": "",
  "_connection_use": "export",
  "Destination Files (.tab)": "",
  "Write region centroids": "false",
  "Export FME Log File": "",
  "Filename prefix": "",
  "Encoding": "",
  "Reverse coordinate axis order (y,x)": "false",
  "Bounds ([xmin] [ymin] [xmax] [ymax])": ""
 }
}
```

```
{
 "created": 1501246885833,
 "createdBy": "1Spatial",
 "updated": 1501246885833,
 "updatedBy": "1Spatial",
 "importType": "Esri Shape",
```

```
 "importCredentials": {
  "Coordinate Reference System": "",
  "_connection_use": "import",
  "Clip to envelope": "false",
  "Allow invalid geometries": "true",
  "Encoding": "",
  "Reverse coordinate axis order (y,x)": "false",
  "Trim preceding spaces": "true",
  "Dissolve holes": "true",
  "Import FME Log File": "",
  "Convert attribute names to upper case": "false",
  "Exposed attributes": "",
  "Fix ring direction and inclusion errors": "true",
  "Source Files (.shp)": "Glaisnock.zip",
  "Treat measures as elevation": "false"
 },
 "exportType": "MapInfo TAB (MFAL)",
 "exportCredentials": {
  "Coordinate Reference System": "",
  "_connection_use": "export",
  "Destination Files (.tab)": "",
  "Write region centroids": "false",
  "Export FME Log File": "",
  "Filename prefix": "",
  "Encoding": "",
  "Reverse coordinate axis order (y,x)": "false",
  "Bounds ([xmin] [ymin] [xmax] [ymax])": ""
 },
 "version": 1
}
```

Credentials Properties

- All properties are optional unless stated otherwise.
- All optional Boolean properties default to false unless stated otherwise.
- All properties are represented as Strings in responses, but can be provided as either Strings or the real type of the property in requests.

Note: See the 1Integrate WebHelp for more guidance on datastore import and export properties.

MapInfo Tab (GDAL)

Take care to spell the datastore `importType` property as follows:

```
"importType": "MapInfo Tab (GDAL)"
```

Import

Property Name	Data Type	Optionality	Description
Coordinate Reference System	String	Mandatory	An empty string indicates this should be inferred from the data.

Property Name	Data Type	Optionality	Description
Source Files (.tab,.dat,.map,.id,.ind,.mif,.mid)	String	Mandatory (and must be non-empty) before uploading data; optional afterwards, as long as no more data is to be uploaded.	This is an internally used label for the uploaded data which is placed into the repository. The UI uses the path of the uploaded file as the label. Re-using the same label tells the datastore that it can re-use the already uploaded data.
Allow invalid geometries	Boolean		

Export

Property Name	Data Type	Optionality	Description
Destination Files (.tab,.dat,.map,.id,.ind,.mif,.mid)	String		

Oracle

Take care to spell the datastore `importType` property as follows:

```
"importType": "Oracle"
```

Configuration of an Oracle datastore is complicated. Every parameter is individually optional, however there are some mandatory combinations.

One of the following connection configurations must be satisfied:

- JNDI Location
- Net Service Name, Username and Password
- Service Name, Host, Port, Username and Password

The above is the priority order in which a connection is attempted.

If you supply more than one type of configuration, only the first prioritised method will be used. For example, if JNDI Location and the Service Name parameters are provided but connection via JNDI fails, the request will fail without attempting to use the Service Name connection mode.

Import

Property Name	Data Type	Optionality	Description
Username	String		
Password	String (Password)		See "Password Properties" on page 50.
Net Service Name	String		
Host	String		
Port	Integer		
Service Name	String		
JNDI Location	String		
Schema Name	String		
Allow Invalid Geometries	Boolean		Note the capitalisation.
Scale for Coordinate Data	Integer		

Export

Property Name	Data Type	Optionality	Description
Username	String		
Password	String (Password)		See "Password Properties" on page 50.
Net Service Name	String		
Host	String		
Port	Integer		

Property Name	Data Type	Optionality	Description
Service Name	String		
JNDI Location	String		
Schema Name	String		

Esri Shape

Note: Esri Shape datastores can be used with either GDAL or FME connections. Some of the following properties are only relevant to FME connections.

Take care to spell the datastore `importType` property as follows:

For FME connections:

```
"importType": "Esri Shape"
```

For GDAL connections:

```
"importType": "ESRI Shape (GDAL)"
```

Import

Property Name	Data Type	Optionality	Description
Coordinate Reference System	String	Mandatory	An empty string indicates this should be inferred from the data.
Source Files (.shp)	String	Mandatory and must be non-empty before uploading data; optional afterwards, as long as no more data is to be uploaded.	This is an internally used label for the uploaded data which is placed into the repository. The UI uses the path of the uploaded file as the label. Re-using the same label tells the datastore that it can re-use the already uploaded data.
Allow invalid geometries	Boolean		

Property Name	Data Type	Optionality	Description
The following parameters are only necessary for FME connections (not GDAL):			
Import FME Log File	String		
Fix ring direction and inclusion errors	Boolean		
Reverse coordinate axis order (y,x)	Boolean		
Convert attribute names to upper case	Boolean		
Treat measures as elevation	Boolean		
Encoding	String	Mandatory	
Clip to envelope	Boolean		
Dissolve holes	Boolean		
Trim preceding spaces	Boolean		
Exposed attributes	String	Mandatory	

Export

Property Name	Data Type	Optionality	Description
Destination Files (.shp)	String	Mandatory and must be non-empty before uploading data; optional afterwards, as long as no more data is to be uploaded.	
The following parameters are only necessary for FME connections (not GDAL):			
Coordinate Reference System	String	Mandatory	An empty string indicates this should be inferred from the data.
Export FME Log File	String		
Reverse coordinate axis order (y,x)	Boolean		
Convert attribute names to upper case	Boolean		
Treat measures as elevation	Boolean		
Encoding	String	Mandatory	
Strict compatibility	Boolean		
Surface and solid storage	String		

Esri File Geodatabase (FILEGDB)

Note: Esri File Geodatabase can be used with either GDAL or FME connections. Some of the following properties are only relevant to FME connections.

Take care to spell the datastore `importType` property as follows:

For FME connections:

```
"importType": "Esri Geodatabase File (FILEGDB)"
```

For GDAL connections:

```
"importType": "Esri File Geodatabase (GDAL)"
```

Import

Property Name	Data Type	Optionality	Description
Coordinate Reference System	String	Mandatory	An empty string indicates this should be inferred from the data.
Source Files (.gdb)	String	Mandatory and must be non-empty before uploading data; optional afterwards, as long as no more data is to be uploaded.	This is an internally used label for the uploaded data which is placed into the repository. The UI uses the path of the uploaded file as the label. Re-using the same label tells the datastore that it can re-use the already uploaded data.
Allow invalid geometries	Boolean		
The following parameters are only necessary for FME connections (not GDAL):			
Import FME Log File	String		
Fix ring direction and inclusion errors	Boolean		

Property Name	Data Type	Optionality	Description
Reverse coordinate axis order (y,x)	Boolean		
Clip to envelope	Boolean		
Exposed attributes	String	Mandatory	

Export

Property Name	Data Type	Optionality	Description
Destination Files (.gdb)	String	Mandatory and must be non-empty before uploading data; optional afterwards, as long as no more data is to be uploaded.	
The following parameters are only necessary for FME connections (not GDAL):			
Coordinate Reference System	String	Mandatory	An empty string indicates this should be inferred from the data.
Export FME Log File	String		
Reverse coordinate axis order (y,x)	Boolean		

PostGIS

Take care to spell the datastore `importType` property as follows:

```
"importType": "PostGIS"
```

Import

Property Name	Data Type	Optionality	Description
Coordinate Reference System	String	Mandatory	An empty string indicates this should be inferred from the data.
Source Database	String	Mandatory	
Import FME Log File	String		
Allow invalid geometries	Boolean		
Fix ring direction and inclusion errors	Boolean		
Reverse coordinate axis order (y,x)	Boolean		
Command timeout	Integer	Mandatory	
Retrieve all schemas	Boolean	Mandatory	Must be set to true to do anything useful with the datastore.
Import host name	String	Mandatory	
Import port number	Integer	Mandatory	
Import username	String	Mandatory	
Import password	String (Password)	See "Password Properties" on page 50.	
WHERE clause	String	Mandatory	
Clip to envelope	Boolean		

Property Name	Data Type	Optionality	Description
Read cache size	Integer	Mandatory	
Assume one SRID per column	Boolean		
Exposed attributes	String	Mandatory	

Export

Property Name	Data Type	Optionality	Description
Coordinate Reference System	String	Mandatory	An empty string indicates this should be inferred from the data.
Source Database	String	Mandatory	
Export FME Log File	String		
Reverse coordinate axis order (y,x)	Boolean		
Command timeout	Integer	Mandatory	
Export host name	String	Mandatory	
Export port number	Integer	Mandatory	
Export username	String	Mandatory	
Export password	String (Password)	See "Password Properties" on page 50.	
Spatial type is geography (default: geometry)	Boolean		
Spatial column name	String	Mandatory	

Property Name	Data Type	Optionality	Description
Orient polygons	Boolean		

Other Data Store Formats

The following formats are also supported by the REST API. Contact 1Spatial for further details.

- Comma Separated Value (CSV)

```
"importType": "Comma Separated Value (CSV)"
```

- Autodesk AutoCAD DWG/DXF

```
"importType": "Autodesk AutoCAD DWG/DXF"
```

- Bentley MicroStation Design (V8)

```
"importType": "Bentley MicroStation Design (V8)"
```

- Microsoft SQL Server Spatial

```
"importType": "Microsoft SQL Server Spatial"
```

- Esri Enterprise Geodatabase

```
"importType": "Esri Enterprise Geodatabase"
```

Password Properties

Some credential properties are considered Passwords.

These:

- Will never appear in responses.
- Are optional in requests.

Note: If a password property has previously been set, and is not supplied with a request to update a resource, the existing password will be used.

Automatic Schema Derivation

Import Schema Derivation

Upon a `PUT` request, or a file upload to a datastore, the import schema will be derived from the import credentials if *all* of the following are true:

- Import credentials have been supplied.
- No import schema has been supplied.
- Import schema can be obtained from the supplied credentials.

Export Schema Derivation

Upon a `PUT` request, or a file upload to a datastore, the export schema will be derived from the import schema and user-defined schema if *all* of the following are true:

- Import schema exists (was either supplied or derived as above).
- Export credentials were supplied.
- No export schema has been supplied.

Schema Mapping

Schema mapping may be optionally supplied within the `importClasses` or `exportClasses` lists in a datastore.

Schema mapping elements are omitted by default for clarity but may be added to any class or attribute definition.

Valid options are:

- `mappedName` – custom name for the class or attribute
- `indexed` – if the attribute should be indexed
- `reported` – if the attribute should be included in any reports
- `disabled` – if the attribute should be displayed in the user interface, but not loaded into the system when running sessions

Note: The response you get back from this update request is almost identical to the request, but with different timestamps and version.

```
PUT http://[server]:[port]/1Integrate/rest/datastores/{path}
```

```
{
 "importType": "RandomMockDataStore",
 "importCredentials": {
  "ClassCount": "7",
  "InstanceCount": "50",
  "StringAttributeCount": "12",
  "LongAttributeCount": "5",
  "Seed": "2",
  "DoubleAttributeCount": "3",
  "IntAttributeCount": "4"
 },
 "importClasses": [
```

```
  {
   "attrs": [
    {
     "indexed": true,
     "reported": true,
     "dataType": "int",
     "mappedType": "Integer",
     "disabled": true,
     "name": "force-int"
    },
    {
     "dataType": "geometry",
     "mappedType": "Geometry",
     "name": "geometry"
    }
   ],
   "name": "angora",
   "mappedName": "newclass"
  }
 ],
 "exportType": "New Export Credentials"
}
```

```
{
 "created": 1501168989408,
 "createdBy": "1Spatial",
 "updated": 1501249854206,
 "updatedBy": "1Spatial",
 "importType": "RandomMockDataStore",
 "importCredentials": {
  "ClassCount": "7",
  "InstanceCount": "50",
  "StringAttributeCount": "12",
  "LongAttributeCount": "5",
  "Seed": "2",
  "DoubleAttributeCount": "3",
  "IntAttributeCount": "4"
 },
 "importClasses": [
  {
```

```
    "attrs": [
     {
      "dataType": "int",
      "disabled": true,
      "indexed": true,
      "mappedType": "Integer",
      "name": "force-int",
      "reported": true
     },
     {
      "dataType": "geometry",
      "mappedType": "Geometry",
      "name": "geometry"
     }
    ],
    "mappedName": "newclass",
    "name": "angora"
   }
 ],
 "exportType": "New Export Credentials",
 "version": 3
}
```


Test and Refresh Mapping

You can test your input and export mapping using your REST client.

Test Input Mapping:

To test input mapping, make the following request:

```
GET http://[host]:[port]/1Integrate/rest/datastores/
[datastore_path]/import/test
```

This will return a “success” flag, as well as an optional “message” detailing any errors.


```
{
"message": "Connection Failed: [message]",
"success": false
}
```


Test Output Mapping:

To test output mapping, make the following request:

```
GET http://[host]:8080/1Integrate/rest/datastores/
[datastore_path]/export/test
```

This will return a “success” flag, as well as an optional “message” detailing any errors.

```
{
"message": "Connection Failed: [message]",
"success": false
}
```

refresh mapping:

To refresh mapping, make the following request:

> **Note:** When refreshing via REST, all classes and attributes are automatically enabled.

```
PUT http://[host]:8080/1Integrate/rest/datastores/
[datastore_path]/refresh
```

```
{
"import": [true|false],
"export": [true|false]
}
```

This will return a “success” flag, as well as an optional “message” detailing any errors.

```
{
"message": "Connection Failed: [message]",
"success": false
}
```

User Defined Classes

You can add temporary classes that are only used in the rules/actions to a datastore.

Note again that the response is almost identical to the request, just with different timestamps and version.

> **Note:** The response is almost identical to the request, but with different timestamps and version.

```
PUT http://[server]:[port]/1Integrate/rest/datastores/
{path}
```

```
{
 "importType": "RandomMockDataStore",
```

```
"importCredentials": {
 "ClassCount": "7",
 "InstanceCount": "50",
 "StringAttributeCount": "12",
 "LongAttributeCount": "5",
 "Seed": "2",
 "DoubleAttributeCount": "3",
 "IntAttributeCount": "4"
},
"importClasses": [
 {
  "attrs": [
   {
    "indexed": true,
    "reported": true,
    "dataType": "int",
    "mappedType": "Integer",
    "disabled": true,
    "name": "force-int"
   },
   {
    "dataType": "geometry",
    "mappedType": "Geometry",
    "name": "geometry"
   }
  ],
  "name": "angora",
  "mappedName": "newclass"
 }
],
"userDefinedClasses": [
 {
  "name": "AddedClass",
  "attrs": [
   {
    "name": "geometry",
    "dataType": "Geometry"
   },
   {
```

```
      "name": "AddedAttribute",
      "dataType": "Boolean"
     }
    ]
   }
 ],
 "exportType": "New Export Credentials"
}
```



```
{
 "created": 1501168989408,
 "createdBy": "1Spatial",
 "updated": 1501249854206,
 "updatedBy": "1Spatial",
 "importType": "RandomMockDataStore",
 "importCredentials": {
  "ClassCount": "7",
  "InstanceCount": "50",
  "StringAttributeCount": "12",
  "LongAttributeCount": "5",
  "Seed": "2",
  "DoubleAttributeCount": "3",
  "IntAttributeCount": "4"
 },
 "importClasses": [
  {
   "attrs": [
    {
     "dataType": "int",
     "disabled": true,
     "indexed": true,
     "mappedType": "Integer",
     "name": "force-int",
     "reported": true
    },
    {
     "dataType": "geometry",
     "mappedType": "Geometry",
     "name": "geometry"
    }
```

```
    ],
    "mappedName": "newclass",
    "name": "angora"
   }
 ], ],
 "userDefinedClasses": [
  {
   "name": "AddedClass",
   "attrs": [
    {
     "name": "geometry",
     "dataType": "Geometry"
    },
    {
     "name": "AddedAttribute",
     "dataType": "Boolean"
    }
   ]
  }
 ],
 "exportType": "New Export Credentials",
 "version": 4
}
```

CopyTo Task Output File Download

The resulting data from a CopyTo task can be downloaded using a GET request, along with the path to the data store and `/outputFile`:

GET http://[server]:[port]/1Integrate/rest/datastores/Test/outputFile

This request will output a file with a default name `data` with no extension, to set a filename use the request:

http://[server]:[port]/1Integrate/rest/datastores/Test/outputFile?fileName=[filename].[extension]

Note: This request will produce `application/octet-stream` content. An Accept: HTTP Header is not required, but if included ensure it states `Accept:application/octet-stream`, any other content will not work.

7 Rules

Rules are encoded in a simplified JSON form as used by the 1Integrate user interface.

Note: The format of rules is subject to change in future releases, therefore it is not recommended to develop significant dependencies on this structure.

Below is an example of a rule response from a `GET` request.

```
GET http://[server]:[port]/1Integrate/rest/rules/{path}
```

```
{
 "template": false,
 "rootTerm": {
  "predicates": [
   {
    "predicates": [
     {
      "predicates": [
       {
        "attributes": {
         "objRef": "",
         "classRef": "mandolin",
         "propName": "expansion-long"
        },
        "kind": "DynamicValue"
       },
       {
        "kind": "LessRelation"
       },
       {
        "attributes": {
         "value": "10",
         "datatype": "integer"
        },
        "kind": "StaticValue"
       }
      ],
```

```
        "kind": "RelationalPredicate"
       }
      ],
      "attributes": {
       "objLabel": "",
       "classLabel": "mandolin"
      },
      "kind": "RootPredicate"
     }
   ],
   "kind": "Rule"
 },
 "createdBy": "1Spatial",
 "updated": 1488809081761,
 "created": 1488809081761,
 "version": 1,
 "updatedBy": "1Spatial"
}
```

8 Actions

As with rules, actions are encoded in a simplified JSON form as used by the 1Integrate user interface.

> **Note:** The format of actions is subject to change in future releases, therefore it is not recommended to develop significant dependencies on this structure.

Below is an example of an action response from a `GET` request.

```
GET http://[server]:[port]/1Integrate/rest/actions/
{path}
```

```
{
 "created": 1501160588903,
 "createdBy": "1Spatial",
 "updated": 1501160613182,
 "updatedBy": "1Spatial",
 "rootTerm": {
  "kind": "Action",
  "predicates": [
   {
    "kind": "RootOperation",
    "predicates": [
     {
      "kind": "AssignmentOperation",
      "predicates": [
       {
        "kind": "DynamicValue",
        "attributes": {
         "objRef": "",
         "propName": "test",
         "classRef": ""
        }
       },
       {
        "kind": "StaticValue",
        "attributes": {
         "datatype": "integer",
```

```
            "value": "0"
           }
          }
         ]
        }
       ],
       "attributes": {
        "classLabel": "",
        "objLabel": ""
       }
      }
    ]
  },
  "template": false,
  "version": 2
}
```

Action Maps

Action maps are lists of rule and action pairings.

```
GET http://[server]:[port]/1Integrate/rest/actionmaps/{path}
```

```
{
  "pairs": [
    {
      "rule": "/rules/data test",
      "action": "/actions/New Action 1"
    },
    {
      "rule": "/rules/data test",
      "action": "/actions/action folder/"
    }
  ]
}
```

9 Results

Note: Session results can only be accessed if a session is paused or finished. Once a session has been stopped or reset, results are removed from the database and no longer accessible.
Rewinding over a specific task will also delete the results of that task.

Task Summary

The status and results for a session are available from a results resource that is parallel to the session resource.

The same session path is used to access the results for that session. For example, if a session resource is at:

```
http://[server]:[port]/1Integrate/rest/sessions/Folder1/MySession
```

The results are available from:

```
http://[server]:
[port]/1Integrate/rest/results/Folder1/MySession
```

Return the summary of a specific task:

```
GET http://[server]:[port]/1Integrate/rest/results/
{path}?detail=task&taskIdentifier=
```


```
{
 "kind": "ApplyActionTask",
 "started": 1501166280848,
 "duration": 100,
 "actions": [
  {
   "count": 0,
   "errors": 43,
   "path": "/actions/errorAction1",
   "processed": 43,
   "total": 0
  }
 ],
 "classes": [
  {
   "count": 0,
   "errors": 43,
```

```
    "name": "main__water_distribution_location",
    "processed": 43,
    "total": 43
   }
 ],
 "count": 0,
 "errors": 43,
 "processed": 43,
 "reportCount": 43,
 "total": 43,
 "status": "FINISHED"
}
```


Detailed Non-conformance or Error Reports

Returns a detailed non-conformance report or an error report for a task.

```
GET http://[server]:[port]/1Integrate/rest/results/
{path}?detail=report&taskIdentifier=&start=0&count=1000
```


```
[
 {
  "featureId": "1004896203",
  "className": "pipe",
  "attributes": {
   "foxglove-int": {
    "value": "1004896203"
   },
   "geometry": {
    "y0": "242.0",
    "y1": "257.0",
    "x0": "77.0",
    "x1": "85.0"
   }
  },
  "nonconformances": [
   {
    "description": "____pipe.identifier is less than
10____",
    "path": "/rules/rule1"
   }
  ],
```

```
  "gothicId": "384"
 }
]
```

Summarised Error Reports

Returns the aggregated error report of a session (if `path` points to a session), or the aggregated error report of a folder of sessions (if `path` points to a folder).

> **Note:** Sub-directories are not supported

```
GET http://[server]:[port]/1Integrate/rest/results/{path}?detail=aggregatedErrors
```

```
[
 {
  "cause":
"com.onespatial.radius.studio.spatialengine.api.error.IntegrationException",
  "index": 1,
  "instancesByClass": {
   "main__water_distribution_location": 43
  },
  "instancesBySession": {
   "errorSession1": 43
  },
  "message": "No value of this name has been defined in the schema",
  "nativeStackTrace":
"com.onespatial.gothic.util.WrappedGothicException",
  "stackTrace":
"com.onespatial.radius.studio.spatialengine.api.error.IntegrationException:
com.onespatial.gothic.util.WrappedGothicException:
NOSUCHVAL: No value of this name has been defined in the schema",
  "tasktype": "ApplyActionTask"
 }
]
```

Summarised Folder Results

Gives you the aggregated summary report of all sessions in a folder.

> **Note:** Sub-directories are not supported

```
GET http://[server]:[port]/1Integrate/rest/results/
{path}?detail=amalgamatedResults
```


```
[
 {
  "status": "PAUSED",
  "tasks": [
   {
    "kind": "OpenDataTask",
    "started": 1501166280195,
    "duration": 647,
    "classes": [
     {
      "count": 0,
      "errors": 0,
      "name": "main__water_distribution_location",
      "processed": 43,
      "total": 0
     }
    ],
    "count": 0,
    "errors": 0,
    "processed": 43,
    "reportCount": 0,
    "total": 0,
    "status": "FINISHED"
   },
   {
    "kind": "ApplyActionTask",
    "started": 1501166280848,
    "duration": 100,
    "actions": [
     {
      "count": 0,
      "errors": 43,
      "path": "/actions/errorAction1",
      "processed": 43,
      "total": 0
     }
    ],
```

```
    "classes": [
     {
      "count": 0,
      "errors": 43,
      "name": "main__water_distribution_location",
      "processed": 43,
      "total": 43
     }
    ],
    "count": 0,
    "errors": 43,
    "processed": 43,
    "reportCount": 43,
    "total": 43,
    "status": "FINISHED"
   },
   {
    "kind": "PauseTask",
    "started": 1501166280988,
    "duration": 29,
    "count": 0,
    "errors": 0,
    "processed": 0,
    "reportCount": 0,
    "total": 0,
    "status": "PAUSED"
   }
  ]
},
{
 "status": "PAUSED",
 "tasks": [
  {
   "kind": "OpenDataTask",
   "started": 1501166281495,
   "duration": 637,
   "classes": [
    {
     "count": 0,
```

```
      "errors": 0,
      "name": "main__water_distribution_location",
      "processed": 43,
      "total": 0
     }
    ],
    "count": 0,
    "errors": 0,
    "processed": 43,
    "reportCount": 0,
    "total": 0,
    "status": "FINISHED"
   },
   {
    "kind": "ApplyActionTask",
    "started": 1501166282136,
    "duration": 102,
    "actions": [
     {
      "count": 0,
      "errors": 43,
      "path": "/actions/errorAction2",
      "processed": 43,
      "total": 0
     }
    ],
    "classes": [
     {
      "count": 0,
      "errors": 43,
      "name": "main__water_distribution_location",
      "processed": 43,
      "total": 43
     }
    ],
    "count": 0,
    "errors": 43,
    "processed": 43,
    "reportCount": 43,
```

```
    "total": 43,
    "status": "FINISHED"
   },
   {
    "kind": "PauseTask",
    "started": 1501166282278,
    "duration": 26,
    "count": 0,
    "errors": 0,
    "processed": 0,
    "reportCount": 0,
    "total": 0,
    "status": "PAUSED"
   }
  ]
 }
]
```