

1 Integrate for ArcGIS

User Guide

Product version: v 2.1

Document version: v 1.6.2

Document date: 24/06/2019



Copyright 2019 1Spatial plc and its affiliates.

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

US Patent Number 9542416 B2 (2017-01-10)

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 Interface	5
Rule Author Interface	5
Rule Author Interface Components	6
Rule Author Hierarchy	9
1Integrate for ArcGIS Widgets and Add-ins	10
Web AppBuilder Widget	10
ArcMap Add-in	11
ArcGIS Pro Add-in	13
2 Map Services and Schemas	15
Map Services	15
Schemas	17
Getting a Schema	18
3 Rule Sets	20
Creating Rulesets	20
Publishing Rulesets	21
Downloading and Uploading Rulesets	21
Moving, Copying and Deleting Rulesets	23
Packages	24
Creating Packages	24
Moving, Copying and Deleting Packages	24
Free Rulesets	25
Uploading and Publishing a Free Ruleset	25
Using a Free Ruleset	27
4 Rules and Actions	28
Quick Rules and Quick Actions	28
Creating Quick Rules and Quick Actions	28
Assigning Actions to a Quick Rule	30
Upgrading Quick Rules and Quick Actions	31
Writing Rules and Actions	32
Creating Rules	32
Creating Actions	33
Assigning Actions to a Rule	34
Parent and Child Objects	35
Geometries	36
Conditions	36

Relationships	42
Values	51
Operations	58
Object Labels	62
Aggregate Functions	62
Built-in Functions	65
Built-in Operations	131
Auto-Actions	136
5 Validating and Enhancing Data	137
Using the Web AppBuilder Widget	137
Running Rules	138
Viewing Validation Results	139
Using the ArcMap Add-in	141
Running Rules	142
Viewing Validation Results	142
6 Extensions	144
Adding Extensions	144
Network Graphs	145
Connecting Network Graphs	146
Validating Network Graphs	149
Positional Data Shifting	149
Using Shift Vectors	150
Register the Known Shifts	151
Record Geometric Constraints	152
Apply Shift Vectors	153
Example: Shifting Pipes	153
7 Issue Management	161
Configure an Issue Layer	162
Create an Issue Management Quick Action	162
Worked Example: Manhole covers with an undefined cover shape	163
8 Backup and Restore	170
Backup	170
Restore	170

1 Interface

There are two main interfaces within 1Integrate for ArcGIS:

- ▶ "Rule Author Interface" below
- ▶ "1Integrate for ArcGIS Widgets and Add-ins" on page 10

Validation and enhancement of data is performed using the 1Spatial Add-ins and widgets for a variety of platforms:

- ▶ [Web AppBuilder for ArcGIS](#)
- ▶ [ArcMap](#)
- ▶ [ArcGIS Pro](#)

Rule Author Interface

The Rule Author is an interface used for managing Rule Sets, Packages, Rules and Actions.

Navigation is performed primarily through the Menu on the left-hand side of the screen, and the Toolbar at the top of the screen.

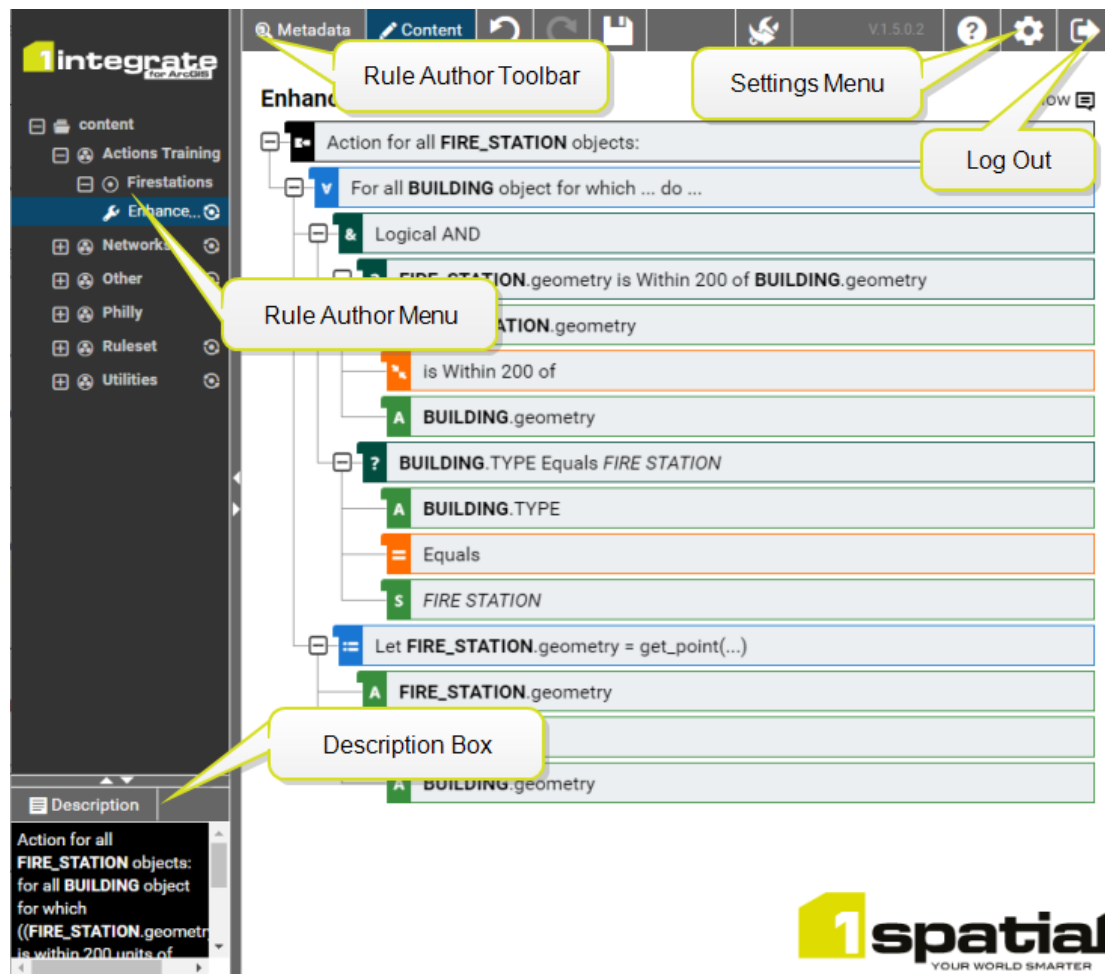


Figure 1-1: Rule Author interface

Rule Author Interface Components

Rule Author Toolbar

The Toolbar at the top of the Rule Author interface contains buttons dependent on the item currently opened from the Rule Author Menu.

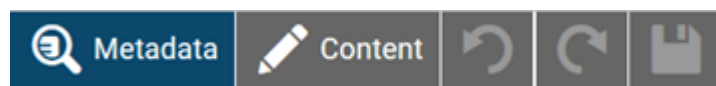


Figure 1-2: Rule Author Toolbar

It contains buttons such as:

- ▶ **Metadata** - add information about the selected item, such as title and description.
- ▶ **Content** (enabled when a rule or action is selected) - edit the business logic of the selected item.

- ▶ **Undo/Redo** (enabled when a rule or action is selected) - allows you to undo or redo the last action.
- ▶ **Save** - save any changes made.

Rule Author Menu

The Menu on the left of the Rule Author interface displays all created rule sets, under the main "content" level folder.

This is where all rule sets, packages, rules and actions are created.

- ▶ Click on items within this menu to view or edit their contents.
- ▶ Right-click on items to perform other actions such as renaming, uploading, downloading and adding child objects.

Contents Window

This main section of the page displays content selected from the menu or toolbar.

It is mainly used for creating and editing Rules and Actions, displaying the structure of the component you have currently selected.

Description Box

When creating or editing a rule, this box provides a description for the currently selected component.

Settings Menu

Clicking the settings icon opens the settings menu.

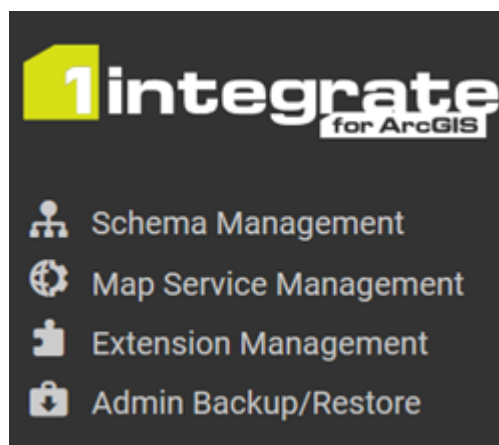


Figure 1-3: *Settings Menu*

From here, the following items can be controlled:

- ▶ **Schema Management** - displays all ArcGIS Server map service schemas currently registered with 1Integrate for ArcGIS (see "Schemas" on page 17).

Schemas are used for rule authoring, to ensure the correct feature class names and attributes are used in the rules.

Within this interface you can refresh the schema (if it has changed within ArcGIS Server), or download the schema so that it can be used for rule authoring in another instance of 1Integrate for ArcGIS.

- ▶ **Map Service Management** - here schemas can be registered and unregistered, and initially captured from ArcGIS Server (see "Map Services" on page 15).

Schemas only have to be registered once, and are downloaded into the Rule Author once "Get Schema" has been selected.

- ▶ **Extension Management** - allows built-in extensions to be uploaded or removed (see "Extensions" on page 144).

Built-in extensions extend the spatial processing capabilities of 1Integrate for ArcGIS, providing functionality beyond the core product. They are currently managed by 1Spatial Consultancy on a customer specific basis.

- ▶ **Admin Backup/Restore** - allows the entire Rule Author Environment (Rule Sets, Packages, Rules and Actions) to be backed up or restored from a single file (see "Backup and Restore" on page 170).

Launch Help

This button launches the 1Integrate for ArcGIS Help System.

Version number







This displays the version number of the Rule Author currently running.

Logout

Click this button to log out of the Rule Author interface.

Rule Author Hierarchy

1Integrate for ArcGIS uses the following hierarchy to organise items within the Rule Author:

Icon	Item	Description
	Rule Set	<p>The top level entity.</p> <p>A Rule Set is "published" to a map service within ArcGIS Server.</p>
	Rule Package	<p>A logical grouping of rules that exist within the Rule Set.</p> <p>Any number of rules or actions can be created within a rule package. Typically groups of rules or actions that are related are grouped together within a Rule Package.</p>
 	Rule / Quick Rule	<p>A rule that has been created within a Rule Package.</p> <ul style="list-style-type: none"> ▶ A Quick Rule is created using a pre-defined template that the user just has to fill in the feature name and any values. ▶ A Rule is created using the full rule authoring capability to provide a more flexible and comprehensive way to define a business rule.
 	Action / Quick Action	<p>An action that has been created within a Rule Package.</p> <p>A Quick Action or Action can be either a <i>reporting</i> action (e.g. where to locate a reporting pin to identify a non-conforming feature), or an <i>enhancement</i> action to manipulate data in some way (e.g. to correct, enhance or create data).</p> <ul style="list-style-type: none"> ▶ A Quick Action is created using a pre-defined template that the user just has to fill in the feature name and any values. ▶ An Action is created using the full rule authoring capability to provide a more flexible and comprehensive way to define an action.

This hierarchy is demonstrated in the following example:

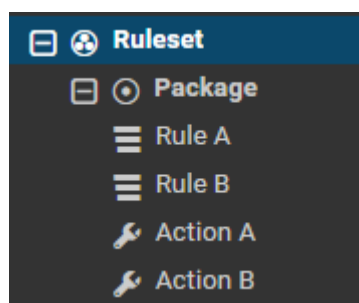


Figure 1-4: An example **Rule Set**, containing a **Package**, containing **Rules** and **Actions**.

1Integrate for ArcGIS Widgets and Add-ins

1Integrate for ArcGIS is available as a widget or Add-in for a variety of platforms, each with a slightly different interface:

- ▶ [Web AppBuilder for ArcGIS](#)
- ▶ [ArcMap](#)
- ▶ [ArcGIS Pro](#)

Web AppBuilder Widget

The Web AppBuilder for ArcGIS can be configured to use the 1Integrate for ArcGIS Widget (see the *1Integrate for ArcGIS Installation Guide*).

It is accessed by clicking on the widget icon in the top-right corner of the interface.



Figure 1-5: *Widget icon*

Opening the widget displays all rulesets that have been published for the current dataset.

Rules are split into Validation and Enhancement tabs, depending on the type of action associated with each rule (Report or Enhance). Within each tab, Rules are displayed within packages, as organised in the Rule Author.

See "Using the Web AppBuilder Widget" on page 137 for details on how to run these Validation and Enhancement rules.

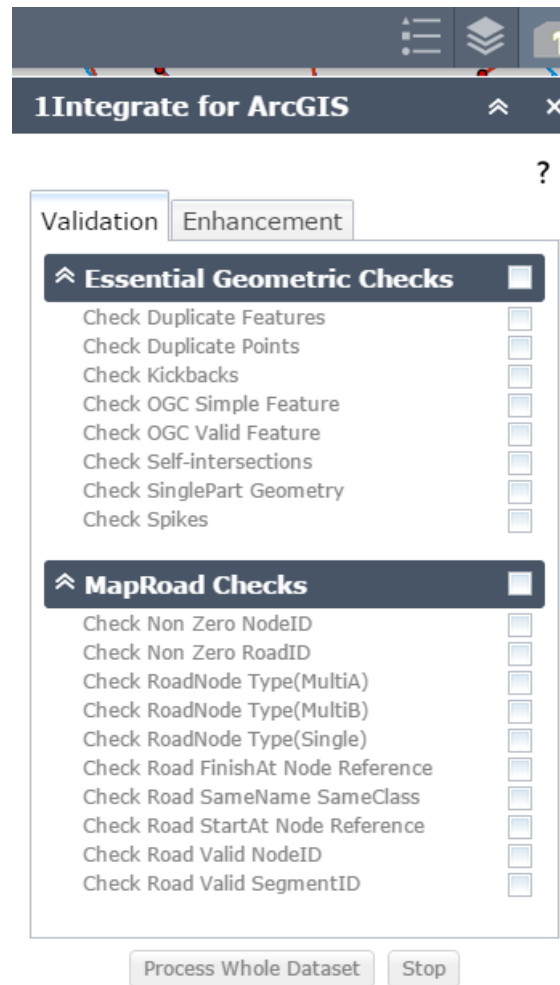


Figure 1-6: Web AppBuilder widget interface

ArcMap Add-in

The 1Integrate for ArcGIS Add-in can be installed for ArcMap (see the *1Integrate for ArcGIS Installation Guide*).

It is opened by clicking on the **1Integrate for ArcGIS** button within the ArcMap toolbar.

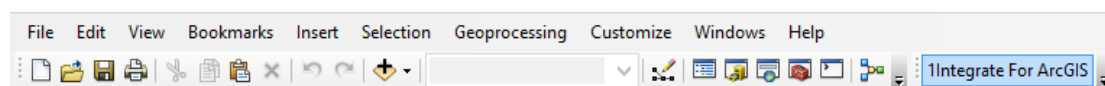


Figure 1-7: ArcMap menu



Note: Initially, the Add-in will appear empty. You must click **Fetch Rules** to display the rulesets that have been published for the current dataset.

Rules are split into Validation and Enhancement tabs, depending on the type of action associated with each rule (Report or Enhance).

Within each tab, Rules are displayed within packages, as organised in the Rule Author.

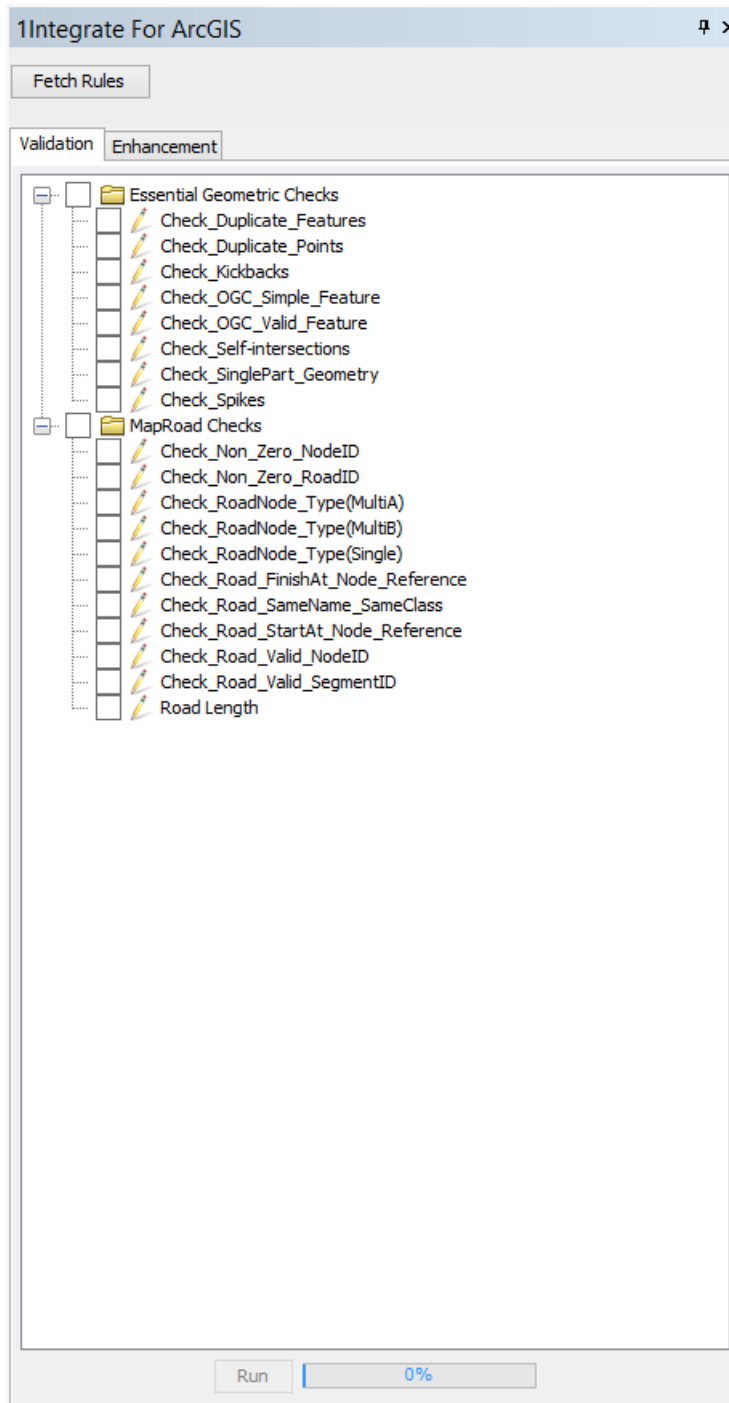


Figure 1-8: *ArcMap Add-in interface*

ArcGIS Pro Add-in

The 1Integrate for ArcGIS Add-in can be installed for ArcGIS Pro (see the *1Integrate for ArcGIS Installation Guide*).

It is opened by navigating to **Add-In > 1Integrate for ArcGIS** within the ArcGIS Pro ribbon menu.

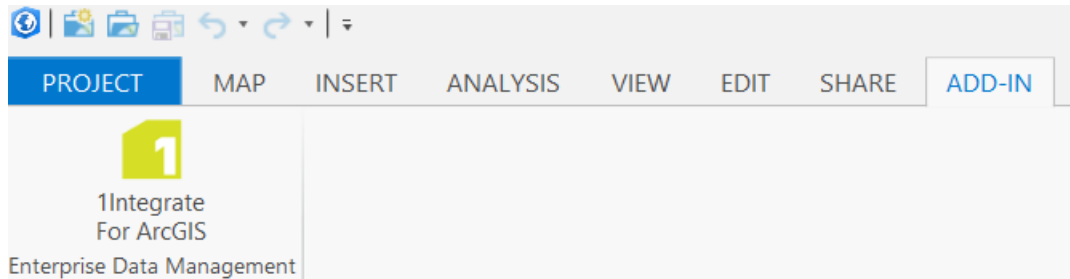


Figure 1-9: *ArcGIS Pro ribbon menu*

Rules are split into Validation and Enhancement tabs, depending on the type of action associated with each rule (Report or Enhance).

Within each tab, Rules are displayed within packages, as organised in the Rule Author.

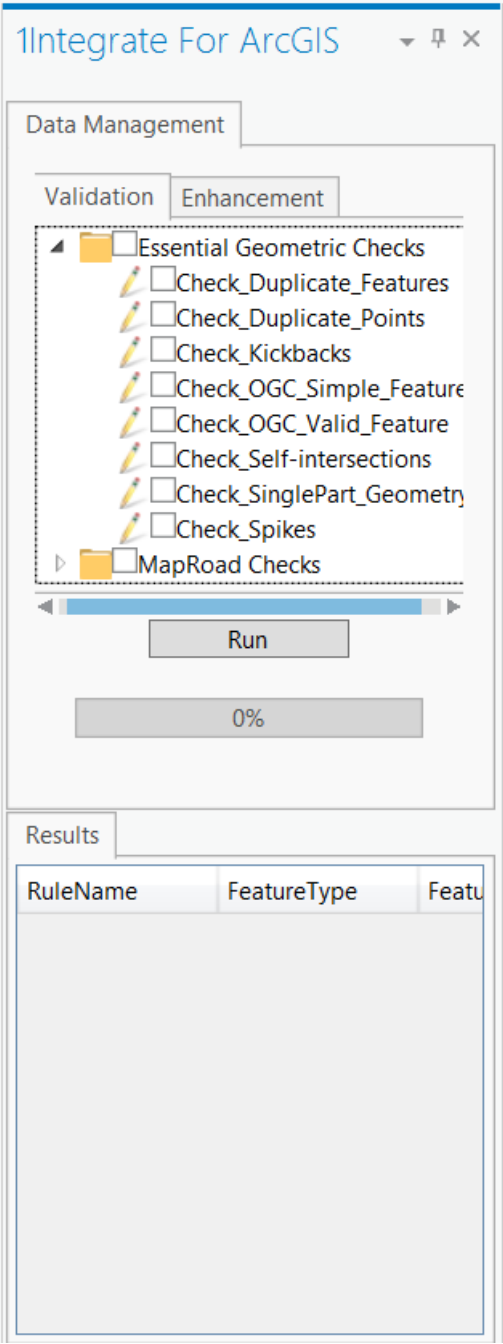


Figure 1-10: *ArcGIS Pro Add-in interface*

2 Map Services and Schemas

Map services must be registered in the 1Integrate for ArcGIS Rule Author so that rules and actions can be written for them.

Schemas are used for rule authoring, to ensure the correct feature class names and attributes are used.

Map Services

Map Service Management can be found in the **Settings** menu in the Rule Author interface.



Figure 2-1: *Settings menu icon*

Map Server Management allows you to connect to the map services within ArcGIS Server that have been configured to use the 1Integrate for ArcGIS extension.

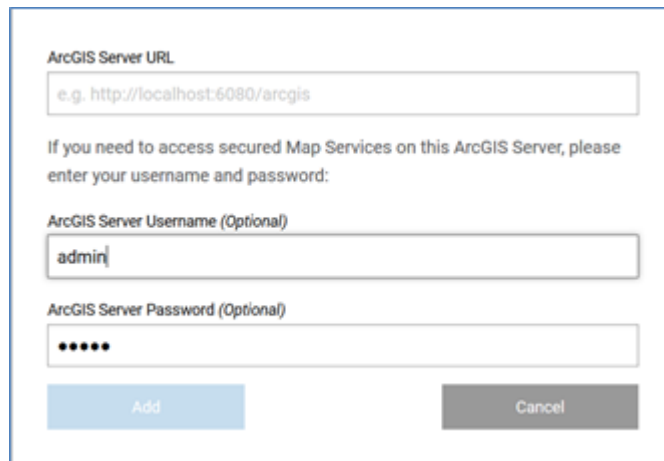
Schemas for these map services can then be selected for a Rule Set, and used to help build rules and actions within the Rule Author (see "Schemas" on page 17).



Note: Multiple ArcGIS Server instances can be added within the Map Service Management interface.

Add An ArcGIS Server to Map Service Management:

1. Navigate to **Settings > Map Service Management**.
2. Select **Add Server**.
3. Enter the details for your ArcGIS Server and click **Add**.



ArcGIS Server URL

e.g. http://localhost:6080/arcgis

If you need to access secured Map Services on this ArcGIS Server, please enter your username and password:

ArcGIS Server Username (Optional)

admin

ArcGIS Server Password (Optional)

•••••

Add Cancel

Figure 2-2: Enter details to connect to ArcGIS Server


- Once successfully connected, the Map Service Management page will display your ArcGIS Server and any map services within it that are configured to use 1Integrate for ArcGIS.

Map Service Management

caml051 ▼	
Service Name	Get Schema
Philly	

Figure 2-3: List of available map services for the selected ArcGIS Server

Download the Schema for a Map Service:

- Navigate to **Settings > Map Service Management**.
- Select the required ArcGIS Server from the drop-down menu, then find the required Map Service.
- Click on the  icon in the Get Schema column.
- A message will confirm the schema has been downloaded.

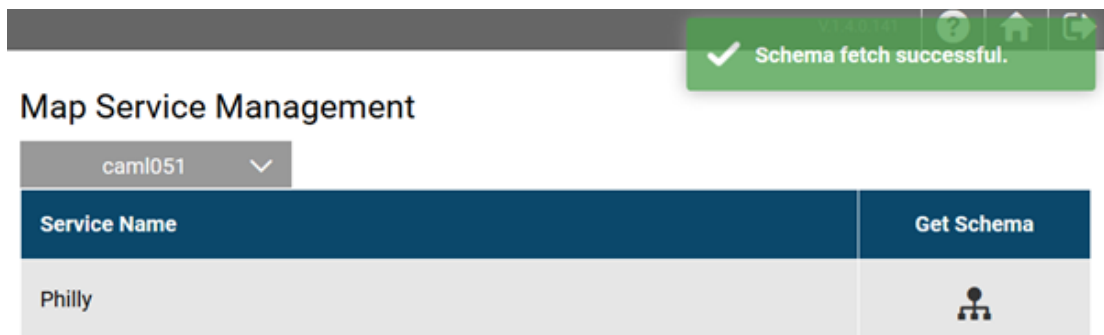


Figure 2-4: Schema successfully downloaded

Schemas

In order to use particular feature classes within your Rules, you must select a schema for the Ruleset.



Note: Data schemas containing more than one layer with the same name (but in different places in the layer tree) are not supported.

Schemas are managed in the Schema Management page, accessed via the settings menu.



Figure 2-5: Settings menu icon

Currently uploaded schemas are visible at the top of the Schema Management page. They can be refreshed, downloaded or deleted.

New schemas can be uploaded via the Schema Upload tool.

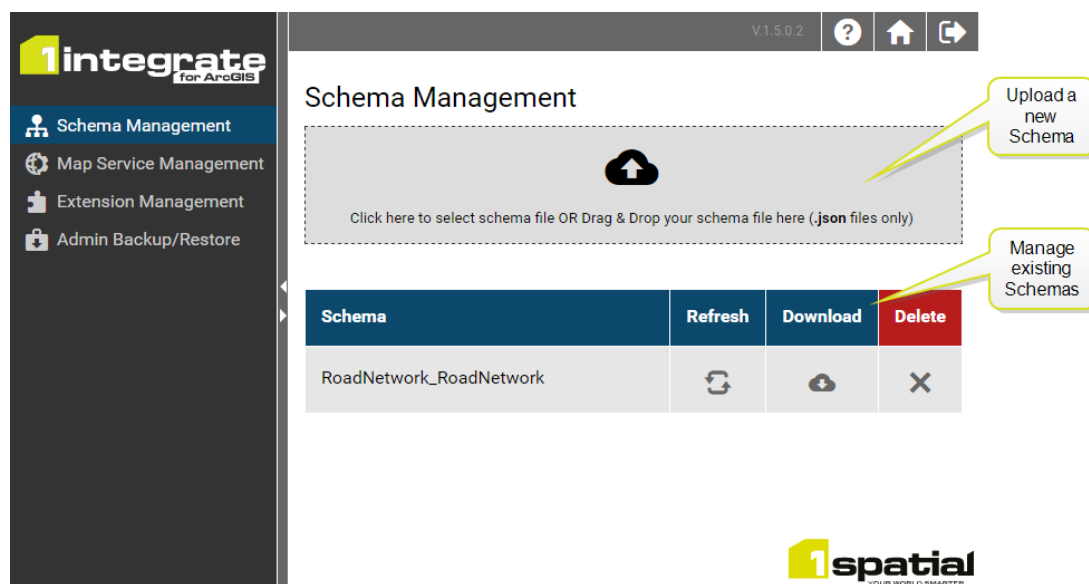


Figure 2-6: Schema Management page

If you do not have any schemas available, you must upload one before selecting it for a Ruleset.

Upload a new Schema:

1. Navigate to **Settings > Schema Management**.
2. Click the Upload button to browse to your schema file (**.json**) or drag and drop the file directly onto the interface.

Select a Schema for a Ruleset:

1. Select the required Ruleset.
2. Click **Select a schema**, and choose your schema from the drop-down menu.
3. Click the **Save** button.

A confirmation message is displayed once the Ruleset has been updated successfully.

Getting a Schema

1Integrate for ArcGIS Server Edition allows you to download a schema from a ArcGIS Map Service. See "Download the Schema for a Map Service" on page 16.

1Integrate for ArcGIS Desktop Edition allows you to apply data validation and enhancement rules to data help in shapefiles and file geodatabases.

To help you write rules and actions using the same schema as your data, there are some tools to help get the data schema from within ArcMap or ArcGIS Pro into the Rule Author.

Get a Schema in ArcMap or ArcGIS Pro:

1. Open your shapefile or file geodatabase data within ArcMap or ArcGIS Pro.
2. Open the 1Integrate for ArcGIS Add-in.
3. Click the **Get Schema** button.

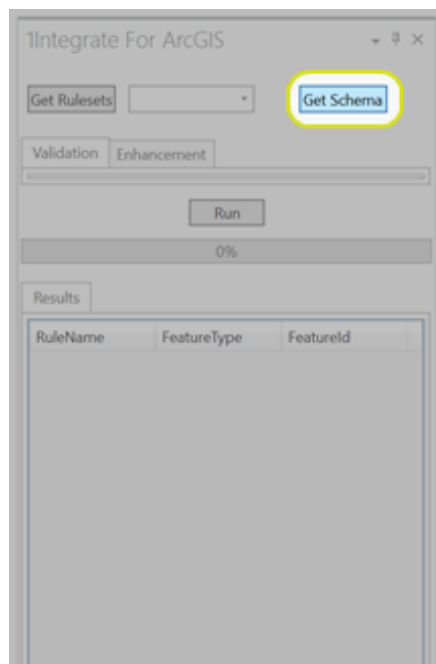


Figure 2-7: *Get Schema for the currently loaded data*

4. The schema for the data that is currently loaded within ArcMap or ArcGIS Pro will be downloaded to a **.json** file.

This can now be uploaded in the **Schema Management** page in the Rule Author (see "Upload a new Schema" on the previous page).

3 Rule Sets

A Rule Set is made up of one or more [Packages](#) containing Rules, Quick Rules, Actions and Quick Actions. Rule Sets are used to manage, publish, backup and restore these Packages.

All Rule Sets are created within the top level **content** folder.

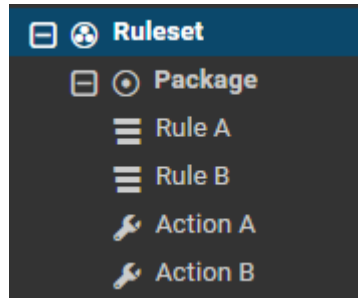


Figure 3-1: An example **Rule Set**, containing a **Package**, containing **Rules** and **Actions**.



Note: In order for rulesets to be available within application Add-ins or the web app widget, they must be published (see "Publishing Rulesets" on the next page) and associated with a data service (see the *1 Integrate for ArcGIS Installation Guide*).



Note: In order to use particular feature classes within your Rules, you must select a schema for the Ruleset (see "Schemas" on page 17).

Creating Rulesets

In order to create your own rules or packages, you must first create a ruleset in which to store them.

Create a New Ruleset:

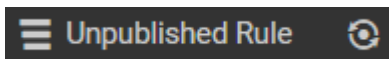
1. Log in to the Rule Author.
2. Right-click on the **content** folder and select **New Ruleset**.
3. Enter a name and click **Create**.

The new Ruleset is created and added to the content folder.

Publishing Rulesets

Published rulesets can be accessed by datasets, and are visible within the application Add-ins or the web app widget.

Any unpublished rules within a ruleset will be marked with an icon.



Publish a ruleset:

1. Right-click on the required ruleset.
2. Click **Publish**.

A confirmation message is displayed once the ruleset has been successfully published.




Figure 3-2: *Confirmation that ruleset is published*


 **Note:** If the Ruleset fails validation, an error will be displayed.

Downloading and Uploading Rulesets

Rulesets (and the Rules and Actions contained within them) can be downloaded or uploaded as a backup file (**.rules**).

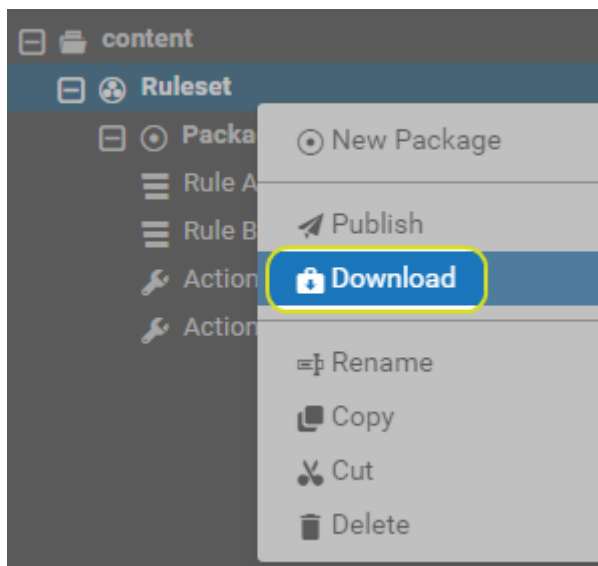
A **.rules** file can be extracted from one Rule Author and then uploaded to another.

 **Note:** The [free ruleset](#) is a **.rules** backup file, and is uploaded in the same way.

 **Note:** It is also possible to upload XML data. Please contact 1Spatial for further information.

Download a Ruleset:

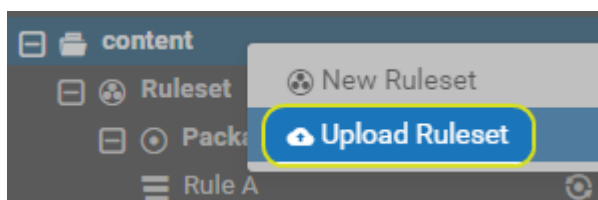
1. Right-click on the required ruleset.
2. Select **Download**.

Figure 3-3: *Downloading a ruleset*

3. The backup file is downloaded to the default web downloads location on your computer.

Upload a Ruleset:

1. Log in to the Rule Author.
2. Right-click on the **contents** folder in the sidebar and click **Upload Ruleset**.

Figure 3-4: *Uploading a ruleset*

3. Enter a name for your ruleset.

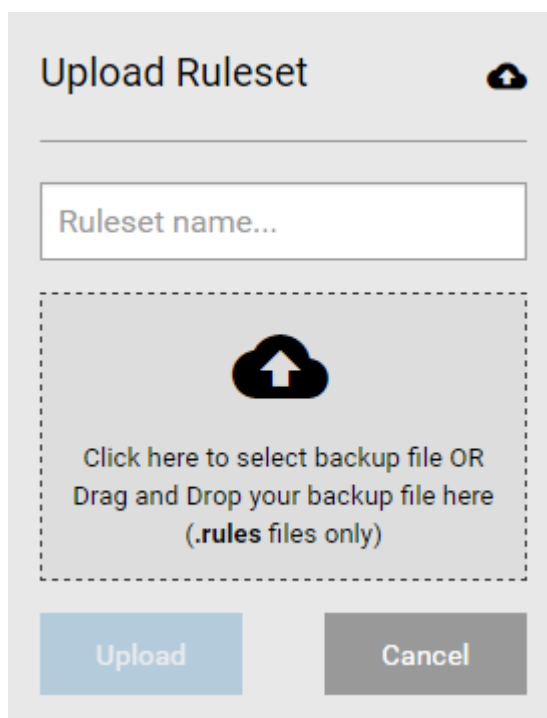


Figure 3-5: Select a ruleset to upload

4. Either drag and drop your **.rules** file onto the **Upload Ruleset** box, or click on the box and navigate to your file.
5. Click **Upload**.

A confirmation message is displayed once the ruleset has been successfully uploaded.



Note: An error message may display if the corresponding schema cannot be found. See "Schemas" on page 17 for information on how to upload a corresponding schema.

Moving, Copying and Deleting Rulesets

Rulesets can be moved to a new destination folder, copied, or deleted.

Move a Ruleset:

1. Right-click on the Ruleset and select **Cut**.
2. Right-click on the destination folder and select **Paste**.
3. Enter a new name for the Ruleset and click **OK**.

Copy a Ruleset:

1. Right-click on the Ruleset and select **Copy**.
2. Right-click on the destination folder and select **Paste**.
3. Enter a new name for the Ruleset and click **OK**.

Delete a Ruleset:

1. Right-click on the Ruleset and select **Delete**.
2. Click **Confirm Delete** to permanently delete the Ruleset.

Packages

Packages contain Rules, Quick Rules, Actions and Quick Actions.

They are contained within [Rulesets](#).

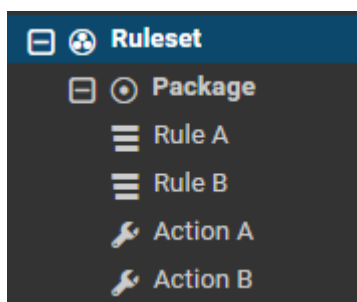


Figure 3-6: An example **Rule Set**, containing a **Package**, containing **Rules** and **Actions**.

Once a Ruleset is published, Packages appear within the application Add-ins and widgets as collapsible grouping levels of Rules.

Creating Packages

Create a Package:

1. Log in to the Rule Author.
2. Right-click on a Ruleset and select **New Package**.
3. Enter a name and click **Create**.

The new Package is created and added to the Ruleset.

Moving, Copying and Deleting Packages

Packages can be moved to a new destination folder, copied, or deleted.

Move a Package:

1. Right-click on the Package and select **Cut**.
2. Right-click on the destination folder and select **Paste**.
3. Enter a new name for the Package and click **OK**.

Copy a Package:

1. Right-click on the Package and select **Copy**.
2. Right-click on the destination folder and select **Paste**.
3. Enter a new name for the Package and click **OK**.

Delete a Package:

1. Right-click on the Package and select **Delete**.
2. Click **Confirm Delete** to permanently delete the Package.

Free Rulesets

A free ruleset is provided as a backup (**.rules**) file so that you can quickly get started with 1Integrate for ArcGIS.

Uploading and Publishing a Free Ruleset

In order to use a free ruleset with a data service, it must first be uploaded and then published.

Upload a Free Ruleset:

1. Log in to the Rule Author.
2. Right-click on the **contents** folder in the sidebar and click **Upload Ruleset**.

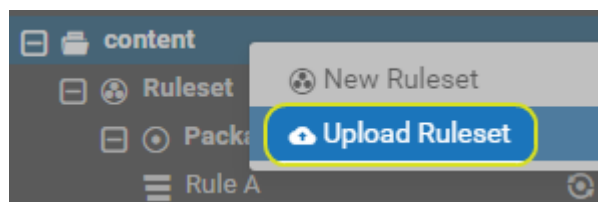


Figure 3-7: *Uploading a ruleset*

3. Enter a name for your ruleset.

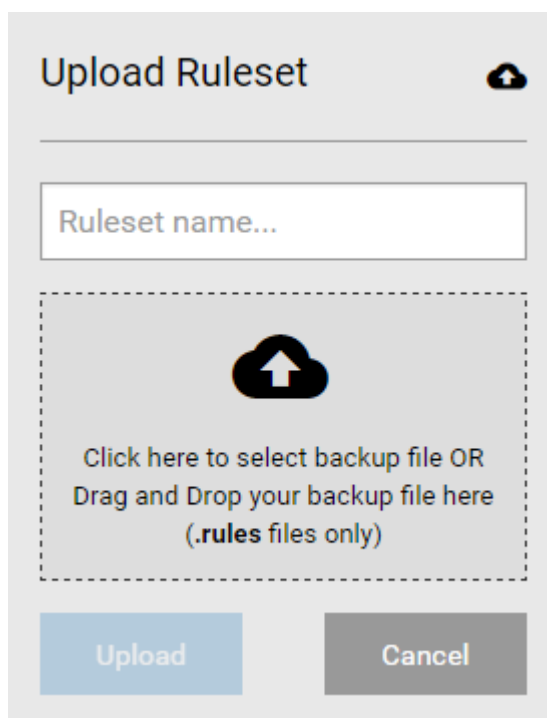


Figure 3-8: Select a ruleset to upload

4. Either drag and drop your **.rules** file onto the **Upload Ruleset** box, or click on the box and navigate to your file.
5. Click **Upload**.

A confirmation message is displayed once the ruleset has been successfully uploaded.



Note: An error message may display if the corresponding schema cannot be found. See "Schemas" on page 17 for information on how to upload a corresponding schema.

Once uploaded, your ruleset must be published before it can be used in the application Add-ins or the web app widget.

Publish a Free Ruleset:

1. Right-click on the required ruleset.
2. Click **Publish**.

A confirmation message is displayed once the ruleset has been successfully published.



Figure 3-9: *Confirmation that ruleset is published*

Note: If the Ruleset fails validation, an error will be displayed.



Note: Ensure that you have configured your data service to use the correct ruleset. See the *1Integrate for ArcGIS Installation Guide*.

Using a Free Ruleset

Once uploaded and published, your free ruleset will be available within your data service.

Open your data service via your web or desktop application, and open the 1Integrate for ArcGIS Add-in or widget.

Your free ruleset will be visible and can be used immediately (see "Validating and Enhancing Data" on page 137).

4 Rules and Actions



Note: Before creating a Rule, you must first create a [Ruleset](#) and a [Package](#) within which it can be stored.

Rules identify a group of features based on sets of logical expressions.

Actions either *Report* or *Enhance* any features identified by their associated Rule.

Rules and Actions can either be written from scratch ("Writing Rules and Actions" on page 32), or from simple templates ("Quick Rules and Quick Actions" below).

Quick Rules and Quick Actions

A **Quick Rule** is a template rule, designed to cover common data validation tasks so that you can quickly create some basic rules. They can be adapted by changing classes or values within their template.

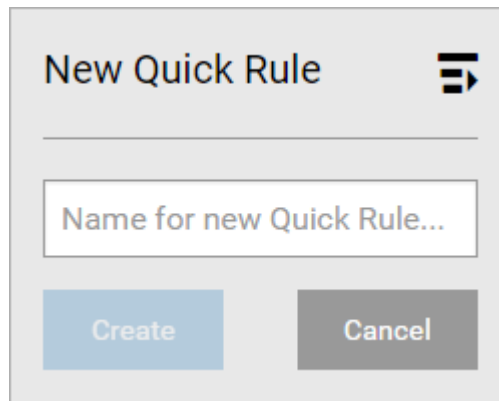
A **Quick Action** is a template action, which can be used to report upon or modify any data defined by a rule. They can be adapted by changing classes or values within their template.

Creating Quick Rules and Quick Actions

Quick Rules and Quick Actions behave in exactly the same way as any other rules and actions, they are just simpler to create.

Create a Quick Rule:

1. Right-click on a package and select **New Quick Rule**.
2. Enter a name for the new Quick Rule and click **Create**.



The dialog box titled "New Quick Rule" features a hamburger menu icon in the top right corner. Below the title is a text input field with the placeholder text "Name for new Quick Rule...". At the bottom of the dialog are two buttons: a blue "Create" button and a grey "Cancel" button.

Figure 4-1: Naming a new Quick Rule

3. Select a type of Quick Rule from the groups displayed:

- ▶ Object Geometry
- ▶ Object Attributes
- ▶ Spatial Relationship
 - ▶ Object Must
 - ▶ Object Must Not

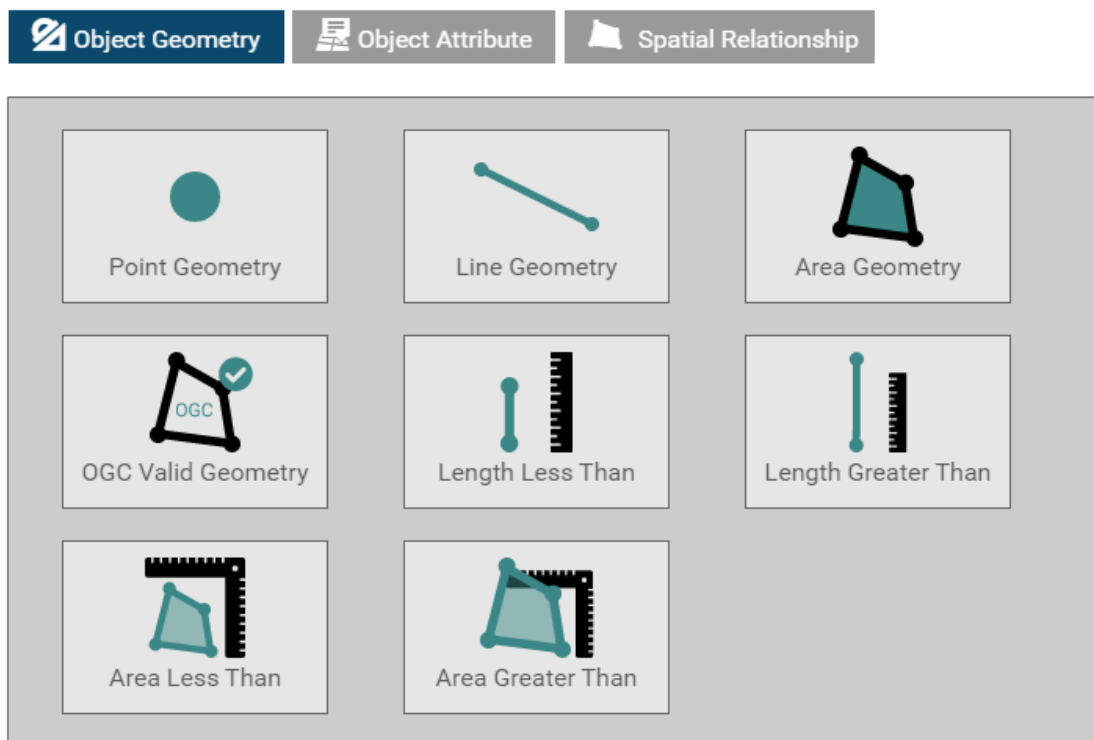


Figure 4-2: Quick Rule categories

4. Select a rule within the group.
5. Depending on the rule selected, define any classes or values required

by the Quick Rule.

6. Click the **Save** button.



Note: A newly created Quick Rule must have an [action assigned to it](#), and then be [published](#) before it can be used in a dataset.

Create a Quick Action:

1. Right-click on a package and select **New Quick Action**.
2. Enter a name for the new Quick Action and click **Create**.
3. Select a type of Quick Action from the groups displayed:
 - ▶ Report Line Features
 - ▶ Geometric Report

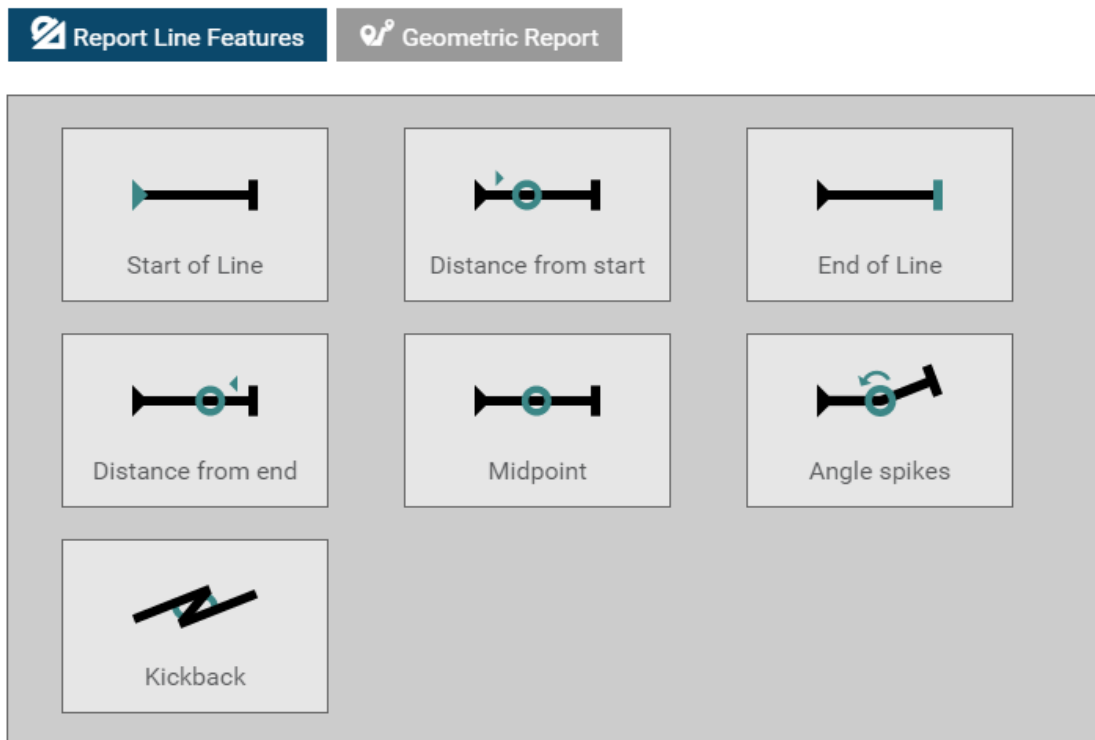


Figure 4-3: *Quick Action categories*

4. Depending on the rule selected, define any classes or values required by the Quick Action.
5. Click the **Save** button.

Assigning Actions to a Quick Rule

Any type of action (quick action or action) can be assigned to any type of rule (quick rule or rule).

If you assign an action to a rule, the action can be applied to any objects that do not conform to the rule.

Once published, it can then be used in the 1Integrate for ArcGIS Add-in or widget.

- ▶ If a reporting action is used, it will appear in the Validation tab, under the *Rule's* name.
- ▶ If an enhancement action is used, it will appear in the Enhancement tab, under the *Action's* name.

The following procedure is the same if you are working with rules, quick rules, actions or quick actions.

Assign an Action to a Quick Rule:

1. Select the rule to be edited.
2. Click **Report Action** or **Enhance Action**, depending on the type of action you wish to assign to the rule.



Figure 4-4: Assign Reporting Action (left) or Enhancement Action (right)

3. Select an action from the available list. You can search for actions by typing in the box.
4. Click the **Save** button.

A confirmation message displays, once the rule has saved successfully.

Upgrading Quick Rules and Quick Actions

Any existing quick rule or quick action can easily be converted to a normal rule or action. This allows the rule to be edited and developed with more precision.

Upgrade a Quick Rule to a Rule:

1. Right-click on the quick rule within the ruleset navigation pane.
2. Select **Make Rule**.

The quick rule will be upgraded, and will now appear with a rule icon.

Upgrade a Quick Action to an Action:

1. Right-click on the quick action within the ruleset navigation pane.
2. Select **Make Action**.

The quick action will be upgraded, and will now appear with an action icon.

Writing Rules and Actions

Rules and Actions are created from a combination of the following basic components:

Condition	A condition is a high-level logical test that defines the syntax for a rule. It defines the sequence in which tests are performed.
Relationship	A relationship compares any two values and returns a result of true or false.
Value	A value is a constant, a reference to objects or a calculation.

Conditions control the flow of tests performed on a feature, and the structure of a rule (or action) defines the sequence in which each condition is tested.

Most conditions are tested one after the other, from the top of the rule to the bottom. The exception is "IF...THEN...ELSE", where either one condition is tested or another, depending on an initial test.

Creating Rules

Create a Rule:

1. Right-click on the package in which to create your new rule, and click **New Rule**.
2. Enter a name for the rule and click **Create**.

A message confirms that your new, empty, rule has been created.

3. Click on the **Metadata** tab, and enter a **Title** and **Description** for the rule. These can act as useful reminders when editing the rule in future.
4. Within the Content tab, double-click **Check for all objects that...** to select a class.

In the **Class** field, enter a label to identify classes to be processed (e.g. "BUILDING"). If the class label is omitted, all classes are queried.

Optionally, enter a **Name** for the feature.

Click the tick to proceed.



Note: If your ruleset uses a schema, you can only select classes and properties defined in that schema.

5. Double-click **Condition** and select a condition from the drop-down menu.

Depending on the condition selected, a number of values, conditions or relationships will be created as child items.

6. Double-click on each **Value**, **Condition** or **Relationship** to define it.
7. Click the **Save** button.

A confirmation message displays, once the rule has saved successfully.



Note: A newly created rule must have an [action assigned to it](#), and then be [published](#) before it can be used in a dataset.

Creating Actions

Actions are created in a similar way to rules. The only difference in structure is that Actions contain an **Operation**, which dictates the primary function that the action should take (e.g. report on values or delete a feature).

Create an Action:

1. Right-click on the package in which to create your new action, and click **New Action**.
2. Enter a name for the action and click **Create**.

A message confirms that your new, empty, action has been created.

3. Click on the **Metadata** tab, and enter a **Title** and **Description** for the action. These can act as useful reminders when editing the action in future.
4. Within the Content tab, double-click **Check for all objects that...** to select a class.

In the **Class** field, enter a label to identify classes to be processed (e.g. "BUILDING"). If the class label is omitted, all classes are queried.

Optionally, enter a **Name** for the feature.

Click the tick to proceed.



Note: If your ruleset uses a schema, you can only select classes and properties defined in that schema.

5. Double-click **Operation** and select an operation from the drop-down menu.

Depending on the operation selected, a number of values, conditions or relationships will be created as child items.

6. Double-click on each **Value**, **Condition** or **Relationship** to define it.
7. Click the **Save** button.

A confirmation message displays, once the action has saved successfully.

Assigning Actions to a Rule

If you assign an action to a rule, the action can be applied to any objects that do not conform to the rule.

Once published, it can then be used in the 1Integrate for ArcGIS Add-in or widget.

- ▶ If a reporting action is used, it will appear in the Validation tab, under the *Rule's* name.
- ▶ If an enhancement action is used, it will appear in the Enhancement tab, under the *Action's* name.

The following procedure is the same if you are working with rules, quick rules, actions or quick actions.

Assign an Action to a Rule:

1. Select the rule to be edited.
2. Click **Report Action** or **Enhance Action**, depending on the type of action you wish to assign to the rule.



Figure 4-5: Assign Reporting Action (left) or Enhancement Action (right)

3. Select an action from the available list. You can search for actions by typing in the box.
4. Click the **Save** button.

A confirmation message displays, once the rule has saved successfully.

Parent and Child Objects

Each component of a rule is displayed in a hierarchy of parent and child objects.

The "root" of the rule is displayed at the top, and can be considered the top-level parent object. Conditions are child objects of this root.

Child objects are required for some components (such as conditions) and will typically be added automatically when required.

Child or sibling objects can also be added manually by right-clicking on a component.

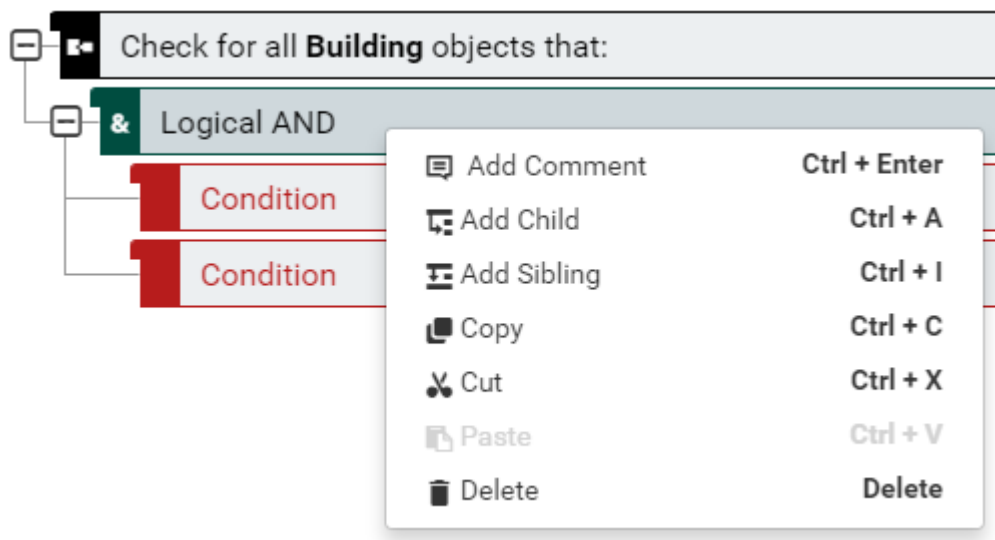


Figure 4-6: *Add a Child or Sibling to a rule component*

Geometries

There are three types of geometry within 1Integrate for ArcGIS: **points**, **lines** and **areas**.

Each can be defined in terms of its dimensions, and if it has an interior, exterior and boundary.

Geometry	Dimensions	Interior	Exterior	Boundary	Description
Point	0	✓	✓	✗	A point is a 0-dimensional geometry that has an interior and an exterior, but no boundary.
Line	1	✓	✓	✓	A line is a 1-dimensional geometry that has an interior, an exterior, and a boundary.
				✗	If the line is a closed ring, there is no boundary (as the boundary is at the end points of a line).
Area	2	✓	✓	✓	An area is a 2-dimensional geometry that has an interior, an exterior, and a boundary.



Note: For multi-part geometries, the interior or boundary is the union of the interiors or boundaries of all parts.

Conditions

A condition is a high-level, logical test that defines the syntax for a rule or action.

A condition can be:

- ▶ A comparison
- ▶ A logical operator
- ▶ A looping construct
- ▶ A collection

Types of Conditions:

Condition	Description	Parameters	Child Nodes
Comparison	<p>A comparison compares two values using a relationship and returns true or false to indicate whether the values fulfil the relationship.</p> <p>The data types of values that can be used depends on the relationship.</p>	None	<ul style="list-style-type: none"> ▶ Value ▶ Relationship ▶ Value
Reference	<p>Test for a reference between two objects. Specify the two objects and the name of the reference between them. The condition is satisfied if the first object refers to the second object using the given reference.</p>	<ul style="list-style-type: none"> ▶ Class label 1 ▶ Object label 1 (optional) ▶ Reference ▶ Class label 2 ▶ Object label 2 (optional) 	None

Condition	Description	Parameters	Child Nodes
Test in Range	<p>Test that a value lies in the range between two other values.</p> <p>Specify whether or not the end points of the range should be included in the comparison (i.e. > or \geq).</p> <p>The values can be integers, reals or strings.</p>	<ul style="list-style-type: none"> ▶ Include upper endpoint (yes or no) ▶ Include lower endpoint (yes or no) 	<ul style="list-style-type: none"> ▶ Value (to test) ▶ Value (range minimum) ▶ Value (range maximum)
AND	<p>Logical AND condition.</p> <p>Used to combine two or more other conditions (supplied as child objects) and check that they are all true.</p>	None	<ul style="list-style-type: none"> ▶ Condition1 ▶ Condition2 ▶ Condition3 ▶ ... ▶ ConditionN
OR	<p>Logical OR condition.</p> <p>Used to combine two or more other conditions (supplied as child objects) and check of any of them are true.</p>	None	<ul style="list-style-type: none"> ▶ Condition1 ▶ Condition2 ▶ Condition3 ▶ ... ▶ ConditionN
XOR	<p>Logical XOR condition.</p> <p>Used to combine two or more conditions (as child nodes) and return true only if exactly one (but no more than one) of the conditions is true, or false otherwise.</p>	None	<ul style="list-style-type: none"> ▶ Condition1 ▶ Condition2 ▶ Condition3 ▶ ... ▶ ConditionN

Condition	Description	Parameters	Child Nodes
NOT	<p>Logical NOT condition.</p> <p>Used to invert the result of another condition (supplied as a child object).</p> <p>For example, if the child condition is true, this condition returns false and vice-versa.</p>	None	<ul style="list-style-type: none"> ▶ Condition
IF... THEN... ELSE	<p>This must have two or three child conditions.</p> <p>The first condition is checked and if it holds then the overall result is the result of checking the second condition.</p> <p>Otherwise, if the first condition does not hold, the overall result is the result of checking the third condition, or true if the third condition is omitted.</p>	None	<ul style="list-style-type: none"> ▶ Condition1 ▶ Condition2 ▶ Condition3 (optional)

Condition	Description	Parameters	Child Nodes
Existence	<p>This may be used to check for the existence (or absence) of related objects satisfying a condition specified in a clause.</p> <p>The objects in the clause are identified by the class and object label pair. You specify a qualifier ("at least", "at most", or "exactly"), a number of objects and another class name.</p> <p>The condition checks that there exists at least one object for which the child condition holds.</p> <p>To check for the absence of objects satisfying a condition, look for exactly 0 objects.</p>	<ul style="list-style-type: none"> ▶ Qualifier ▶ Integer ▶ Class Label ▶ Object Label (optional) 	<ul style="list-style-type: none"> ▶ Condition
For All	<p>Check that all related objects satisfy some other condition.</p> <p>This condition must have two child conditions. The first condition <i>finds</i> a set of objects, the second condition then <i>checks</i> that these objects meet certain requirements.</p>	<ul style="list-style-type: none"> ▶ Class label ▶ Object label (optional) 	<ul style="list-style-type: none"> ▶ Condition (find) ▶ Condition (check)

Condition	Description	Parameters	Child Nodes
Existence in Collection	Checks whether the required number of elements exist in an object collection or an array that match the condition in the clause.	<ul style="list-style-type: none"> ▶ Qualifier ▶ Integer ▶ Class label ▶ Object label (optional) 	<ul style="list-style-type: none"> ▶ Value (collection name) ▶ Condition
For All in Collection	<p>Checks all objects or elements in a collection satisfy a condition.</p> <p>The condition requires a value which should be an array or a collection of objects, and a condition to check.</p> <p>For a collection of simple types, or objects of unspecified class, the class label may be omitted. Otherwise the class label specifies the type of object in the collection.</p>	<ul style="list-style-type: none"> ▶ Class label ▶ Object label 	<ul style="list-style-type: none"> ▶ Value ▶ Condition

Relationships

A relationship compares any two values and returns a result of true or false.

They can either compare scalar values or spatial geometries.

Scalar Relationships

A scalar relationship compares any two values (boolean, integers, real numbers, dates or strings).

The result is evaluated as true or false, returning true if the values have the specified relationship.

Relationships include equal, not equal, less than, greater than etc., as well as relationships specifically for string: contains, begins with, ends with or matches a regular expression.

Scalar Relationship	Description
Equal	The two values are the same.
Not Equal	The two values are different.
Less Than	The first value is less than the second value.
Less Than or Equal	The first value is less than the second value, or the two values are the same.
Greater Than	The first value is greater than the second value.
Greater Than or Equal	The first value is greater than the second value, or the two values are the same.
Begins With	For string values, test whether the first string begins with the second string.
Ends With	For string values, test whether the first string ends with the second string.
Contains	For string values, test whether the first string contains the second string.
Regular Expression	For string values, check whether the first string matches the wild card string or Perl regular expression in the second string.

Spatial Relationships



A spatial relationship is a type of relationship only applied to geometry values. It returns true if the geometries have the specified relationship.



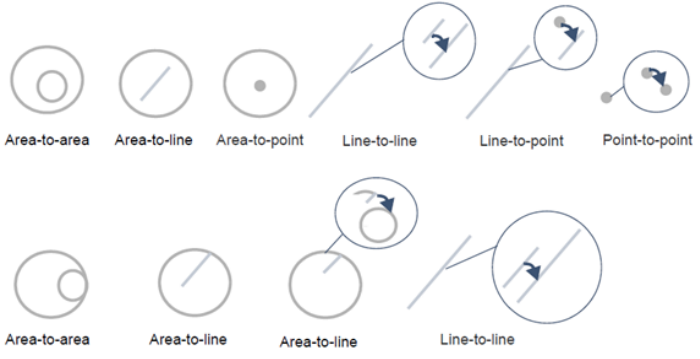

Any type of geometry can be passed into a spatial relationship check, but only the geometry types relevant to this spatial relationship will be tested. For example, if comparing areas, line geometries will not be tested.












Note: Some spatial relationships can be considered subsets of others. For example, Covered By is a subset of Within (if a geometry is Covered By another geometry, it is also Within that geometry).


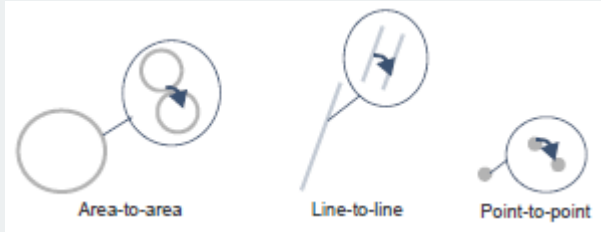
The detailed description of each spatial relationship uses the terms interior, boundary, exterior. See "Geometries" on page 36 for more information on these terms.




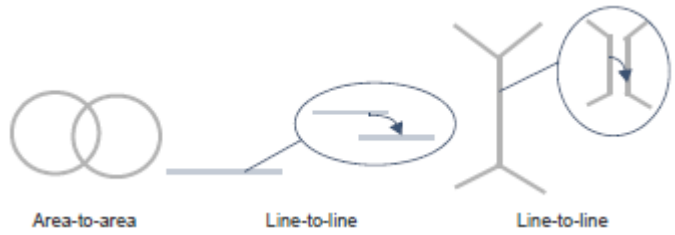
Spatial Relationship	Symbol	Description
Beyond		<p>Beyond checks that the geometry is at least a certain distance away from another (specified in dataset units).</p> <p>The order of the geometries does not matter.</p> <div>  <p>Note: Beyond is the opposite of Within Distance.</p> </div>


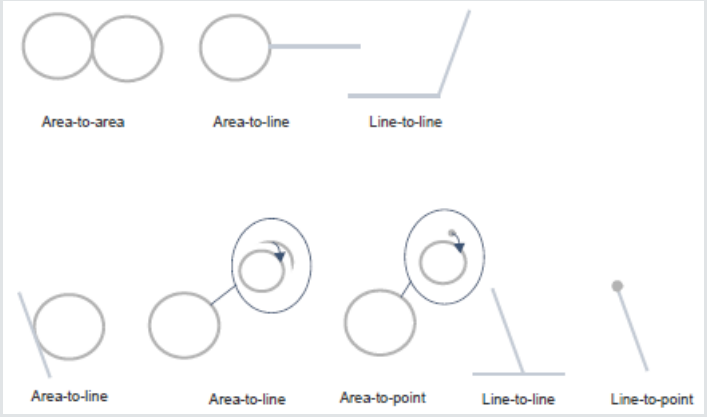
Spatial Relationship	Symbol	Description
Contains		<p>Contain means that the other geometry is completely inside this line or polygon (one object's boundary and interior are inside another).</p> <div>  Note: Their boundaries <i>are</i> allowed to intersect, so it includes the Cover and Equal relationships. </div> <p>For lines the boundary means the end points, and for polygons the boundary means all inner and outer rings.</p> <p>The types of geometries that can contain another are:</p> <ul style="list-style-type: none"> ▶ Area <i>contains</i> area ▶ Area <i>contains</i> line ▶ Area <i>contains</i> point ▶ Line <i>contains</i> line ▶ Line <i>contains</i> point ▶ Point <i>contains</i> point <div>  </div> <p>Figure 4-7: <i>Contains geometries</i></p> <div>  Note: Contains is the reciprocal of Within. </div>





Spatial Relationship	Symbol	Description
Covered By		<p>Covered By is a special case of Within.</p> <p>It is only true when this geometry is contained within another, <i>and</i> their boundaries intersect (so they share some boundary).</p> <p>For lines the boundary means the end points, and for polygons the boundary means all inner and outer boundary rings.</p> <p>The types of geometries which can be covered by another are:</p> <ul style="list-style-type: none"> ▶ Area <i>covered by</i> area ▶ Line <i>covered by</i> area ▶ Line <i>covered by</i> line <div>  Note: Covered by is the reciprocal of Covers. </div>
Covers		<p>Covers is a special case of Contains.</p> <p>It is only true when the other geometry is contained within this one, <i>and</i> their boundaries intersect (so they share some boundary).</p> <p>For lines the boundary means the end points, and for polygons the boundary means all inner and outer boundary rings.</p> <p>The types of geometries which can cover another are:</p> <ul style="list-style-type: none"> ▶ Area <i>covers</i> area ▶ Area <i>covers</i> line ▶ Line <i>covers</i> line <div>  Note: Covers is the reciprocal of Covered By. </div>

Spatial Relationship	Symbol	Description
Cross		<p>Two geometries Cross when a line runs across the boundary of an area, or the interior of another line.</p> <div>  Note: At least one of the geometries <i>must</i> be a line. </div> <p>The types of geometries which can cross are:</p> <ul style="list-style-type: none"> ▶ Line and line A line crosses another line if only their interiors intersect at single points (and at the same height). ▶ Area and line A line crosses a polygon geometry if its interior intersects the polygon's interior and either of the line's endpoints are outside the polygon. <p>The order of the geometries does not matter.</p> <div>  <p>Area-to-line Area-to-line Line-to-line</p> </div> <p>Figure 4-8: <i>Cross geometries</i></p>
Disjoint		<p>Disjoint means that the geometries do not intersect at all (neither the boundaries nor the interiors intersect).</p> <p>The order of the geometries does not matter.</p> <div>  Note: Disjoint is the opposite of Intersect. </div>

Spatial Relationship	Symbol	Description
Equal		<p>Two geometries are Equal when they are identical (they have the same interior, boundary and exterior).</p> <p>Geometries do not need to have the same number of vertices (as long as they follow the same path, within geometric tolerance), and lines do not need to have the same direction.</p> <p>Polygon boundaries do <i>not</i> have to use the same start/end vertex in order to be equal.</p> <p>The order of the geometries does not matter.</p> <p>The types of geometries which can be equal are:</p> <ul style="list-style-type: none"> ▶ Area and area ▶ Line and line ▶ Point and point <div data-bbox="673 1019 1275 1249">  <p>Area-to-area Line-to-line Point-to-point</p> </div> <p>Figure 4-9: <i>Equal geometries</i></p>

Spatial Relationship	Symbol	Description
Intersect		<p>Intersect means any sort of spatial interaction. It checks that either the boundaries or interiors of two geometries intersect in any way, using any of the following relationships:</p> <ul style="list-style-type: none"> ▶ Equal ▶ Touches ▶ Overlap ▶ Cross ▶ Within ▶ Contains <p>The order of the geometries does not matter.</p> <div>  Note: Intersect is the opposite of Disjoint. </div>
Overlap		<p>Two geometries Overlap when two lines or two areas partly overlay each other. Overlap means geometries are <i>partly</i> inside and <i>partly</i> outside each other (so does not include being contained or being equal).</p> <p>The types of geometries that can overlap are:</p> <ul style="list-style-type: none"> ▶ Area and area A polygon overlaps another polygon if it shares some but not all of its area. ▶ Line and line A line overlaps another line if it shares some but not all of its length. <p>The order of the geometries does not matter.</p> <div>  </div> <p>Figure 4-10: Overlap geometries</p>

Spatial Relationship	Symbol	Description
Touches		<p>Touch means that the boundaries intersect but the interiors do not.</p> <p>For lines the boundary means the end points, and for polygons the boundary means all inner and outer rings.</p> <p>Adjacent polygons or end-to-end lines touch.</p> <p>The types of geometries that can touch are:</p> <ul style="list-style-type: none"> ▶ Area and area ▶ Area and line ▶ Area and point ▶ Line and line ▶ Line and point <p>The order of the geometries does not matter.</p> <div data-bbox="622 996 1332 1411">  </div> <p>Figure 4-11: <i>Touch geometries</i></p>

Spatial Relationship	Symbol	Description
Within		<p>Within means this geometry is completely inside the other, which must be a line or polygon.</p> <p>Their boundaries <i>are</i> allowed to intersect, so it includes the Covered by and Equal relationships.</p> <p>For lines the boundary means the end points, and for polygons the boundary means all inner and outer rings.</p> <p>The types of geometries which can be within another are:</p> <ul style="list-style-type: none"> ▶ Area <i>within</i> area ▶ Line <i>within</i> area ▶ Line <i>within</i> line ▶ Point <i>within</i> area ▶ Point <i>within</i> line ▶ Point <i>within</i> point <div data-bbox="620 1072 1326 1473"> </div> <p>Figure 4-12: <i>Within geometries</i></p> <div data-bbox="620 1576 1326 1659">  Note: Within is the reciprocal of Contains. </div>
Within Distance		<p>Within Distance checks that the geometries approach within a specified minimum distance of each other (specified in dataset units).</p> <p>The order of the geometries does not matter.</p> <div data-bbox="620 1883 1326 2007">  Note: Within Distance is the opposite of Beyond. </div>

Values


A value is a constant, a reference to objects or a calculation. Values are compared using relationships (see "Relationships" on page 42).

Types of Values:

Value	Description	Parameters	Child Nodes
Aggregate Value	<p>Calculates a single value by combining others, using functions such as Count, Sum, Average, max, min or geometric union.</p> <p>This value should have a condition to test and zero or more child values.</p> <p>This value is computed by aggregating the values over any objects that are satisfied by the condition.</p> <p>There are a number of different types of aggregate functions. Detailed information on the child values required are provided in a tool-tip within 1Integrate for ArcGIS.</p>	<ul style="list-style-type: none"> ► Function 	<ul style="list-style-type: none"> ► Condition ► Values


Value	Description	Parameters	Child Nodes
Attribute Value	<p>A attribute value evaluated by reading the value of an attribute from an object.</p> <p>You must specify the class of the object and the attribute to read.</p> <p>You may optionally select the name of an object identified in an earlier part of the rule, if it is necessary to distinguish between different objects of the same class.</p>	<ul style="list-style-type: none"> ▶ Class label ▶ Object label ▶ Attribute name 	None
Built-in Function Value	<p>A value which is computed by applying the specified built-in function to one or more parameters.</p> <p>There are typically one or more child values of this element, to specify the parameters which will be passed to the function.</p> <p>When a function has been selected from the list of all possible built-in functions, a help icon provides a tool tip with information about the parameters required by this function.</p> <p>See "Built-in Functions" on page 65.</p>	<ul style="list-style-type: none"> ▶ Function name 	<ul style="list-style-type: none"> ▶ Value 1 ▶ Value n (optional)

Value	Description	Parameters	Child Nodes
Class Name	Returns the name of the class of an object. The object is specified by the class label or object label pair.	<ul style="list-style-type: none"> ▶ Class label ▶ Object label 	None
Conditional Value	<p>This value should have a condition to test and two child values to choose between.</p> <p>If the condition is true, the result is the first child value. If the condition is false, the result is the second child value.</p>	None	<ul style="list-style-type: none"> ▶ Condition ▶ Value 1 ▶ Value 2
Difference	This value should have two child values (integers or real numbers). The result is obtained by subtracting the second value from the first.	None	<ul style="list-style-type: none"> ▶ Value 1 ▶ Value 2
Division	This value should have two child values (integers or real numbers). The result is obtained by dividing the first value by the second.	None	<ul style="list-style-type: none"> ▶ Value 1 ▶ Value 2
Negated Value	<p>This value should have one child value (an integer or a real number), containing a sign that will be inverted.</p> <p>For example, -1 becomes $+1$ and vice-versa.</p>	None	<ul style="list-style-type: none"> ▶ Value

Value	Description	Parameters	Child Nodes
Null Value	<p>A Null value always evaluates as null.</p> <p>This can be used to verify if an object attribute value or the result of a calculation is null.</p> <div> Note: Searching on a null value is not valid for arrays.</div>	None	None

Value	Description	Parameters	Child Nodes
Object or Element Value	<p>An object or an element from a collection.</p> <p>These are used in two ways:</p> <p>A. To access a whole object directly, e.g. to compare objects to check whether they are or the same object or not, or to pass an object in to a built-in function or store it as a temporary value. In this case, users specify the class and, if required, the name alias of the object.</p> <p>B. To access an element when looping through a collection or array. One example is to use a “For All in Collection” or “Loop over a Collection” over a geometry in order to access each part of the geometry. Within the loop, access the element using this Object or Element value, leaving the class name blank but selecting the name specified by the loop.</p>	<ul style="list-style-type: none"> ▶ Class label ▶ Object label 	None

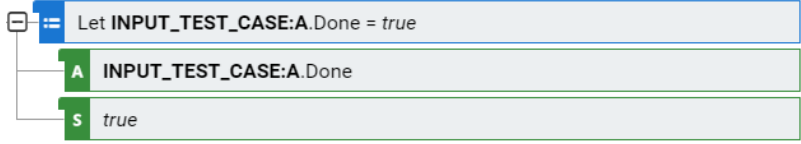
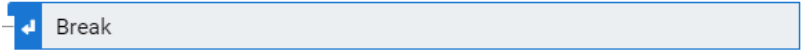
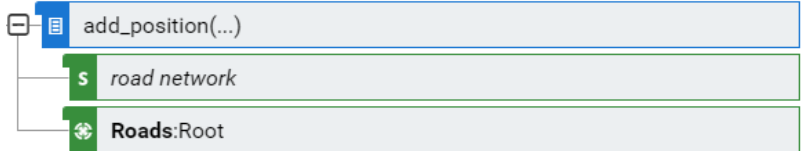
Value	Description	Parameters	Child Nodes
Product	This value should have two child values (integers or real numbers). The result is obtained by multiplying together all child values.	None	<ul style="list-style-type: none"> ▶ Value 1 ▶ Value 2
Remainder	This value should have two child values (integers or real numbers). The result is obtained by dividing the first value by the second, and taking the remainder.	None	<ul style="list-style-type: none"> ▶ Value 1 ▶ Value 2
Static Value	<p>A Static Value is fixed, and does not change.</p> <p>This can be used on either side of a comparison condition or as part of a more complicated expression.</p> <p>It can be a boolean, an integer, a real (floating point) or a string (text or timestamp). You must specify both the type of value, and the value itself.</p>	<ul style="list-style-type: none"> ▶ Datatype 	None
Sum	<p>This value should have at least two child values (integers or real numbers). The result is the sum of all child values.</p> <p>Alternatively, the values can be string types, in which case they are joined together.</p>	None	<ul style="list-style-type: none"> ▶ Value 1 ▶ Value 2 ▶ Value n (optional)

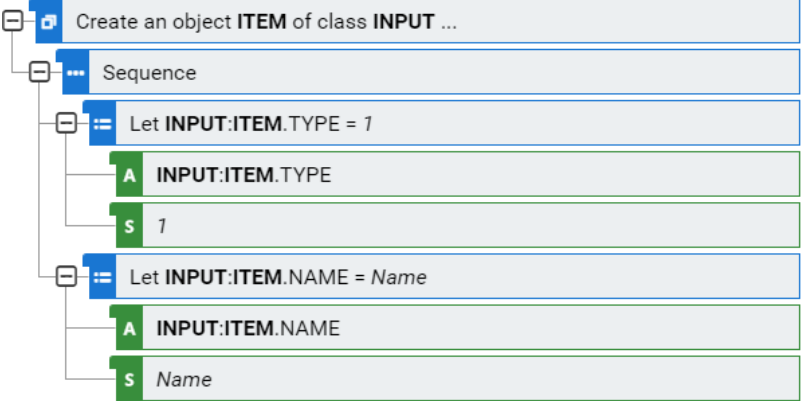

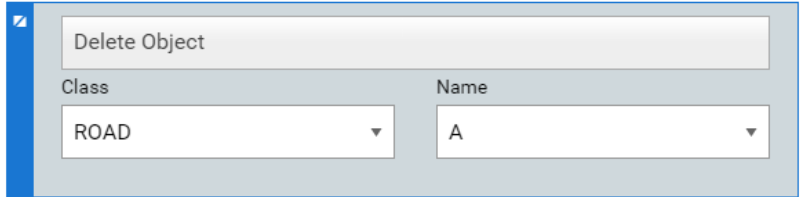
Value	Description	Parameters	Child Nodes
Temporary Value	<p>A temporary value is initially null and can be used to hold temporary results during an action.</p> <div>  Note: A temporary value can be used in an action but not in a rule. </div> <p>The value can have either local or global <i>scope</i>. If it has local scope, the value will be reset to null before the action is applied to each object. If it has global scope, the value will be reset to null only at the start of the whole task.</p> <p>A temporary value can be any type, but this type is determined by the value it is given.</p>	<ul style="list-style-type: none"> ▶ Scope ▶ Value 	None

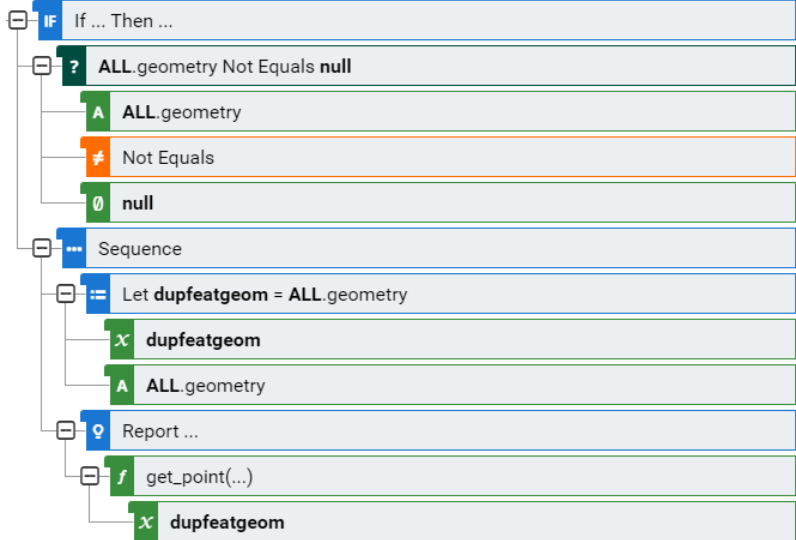

Operations

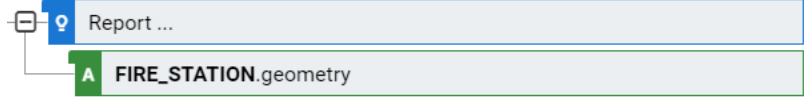
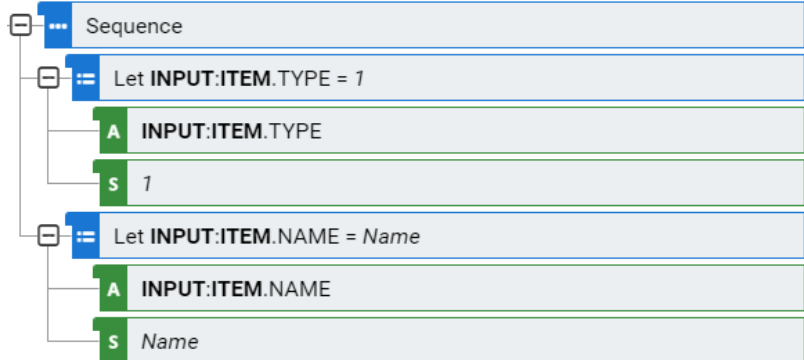


An operation is a high level component of an Action, which determines the primary function of the Action.

Types of Operations:

Operation	Description
Assignment	<p>Assigns a new value to an existing value.</p> <p>Only temporary values and dynamic values can be assigned a new value.</p> <p>For example, assign a new geometry or attribute to a feature. This is used widely to fix or integrate or transform data by updating it.</p>  <p>Figure 4-13: <i>Example Assignment operation</i></p>
Break	<p>When used inside of a loop this will cause the loop to terminate. Otherwise the current action will be terminated.</p>  <p>Figure 4-14: <i>Example Break operation</i></p>
Built-in Operation	<p>Performs a specific task on the data, determined by the built-in function that is selected.</p> <p>If the function requires parameters, these must be added as child values.</p>  <p>Figure 4-15: <i>Example Built-in operation</i></p>

Operation	Description
Create Object	<p>Creates a new object in the class specified.</p> <p>This is used widely to create reporting features representing the exact locations of problems or when inferring new data or creating an enhanced copy of the data.</p>  <p>Figure 4-16: <i>Example Create operation</i></p>
Delete Object	<p>Deletes an existing object.</p> <p>You can specify the class of object to be deleted and its name, if it has been given one.</p> <div data-bbox="496 1182 1337 1350">  Note: Usually you should aim to delete the primary object that the rule is iterating over to avoid later iterating over a feature that has been deleted. </div>  <p>Figure 4-17: <i>Example Delete operation</i></p>

Operation	Description
If...Then...Else	<p>As opposed to the IF...THEN...ELSE condition, which decides which condition to check, this operation is used to decide which operation to perform.</p> <p>The first part is the condition to check, the second part is the operation to perform if the condition is true and the (optional) third part is the operation to perform if the condition is false.</p> 
Loop Over a Collection	<p>Step over a specified geometry or collection and perform an operation on each part.</p> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p> Note: Geometries can be treated like collections and stepping over each part allows each part of a single or multi geometry to be processed individually.</p> </div> <p>Collections are usually created as attribute values by specific bespoke datastores to represent things like multi-cardinality lists. Use the "Object or Element" value to access the current part within the loop.</p>
Loop Over Objects	<p>Step over a specified set of objects in the cache and perform an operation on each one.</p>

Operation	Description
Report on Values	<p>Used to output a point geometry to be used as the pin location in the report.</p>  <p>Figure 4-19: <i>Example Report operation</i></p>
Sequence	<p>Used to perform a number of operations one after the other in the specified order.</p>  <p>Figure 4-20: <i>Example Sequence operation</i></p>
While Loop	<p>A While Loop must consist of two child elements: a test condition and an operation. The operation will be executed repeatedly while the test condition holds.</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p> Note: You must be careful to avoid infinite loops by ensuring that the condition will eventually be met.</p> </div>  <p>Figure 4-21: <i>Example While Loop operation</i></p>

Object Labels

An object label uniquely identifies objects in the same class that are tested against each other in clauses and subclauses of a rule.

Example: Fire station object labels A and B

```
Check for fire_station objects A that there are at
least 5 fire_station objects B for which fire_
station:B.geometry is within 10000m of fire_
station:A.geometry
```

The object labels `A` and `B` distinguish between the root object and objects found in the subclause, both of which have the same class label `fire_station` and come from the same class.

Aggregate Functions

Aggregate functions calculate a single result from one or more input values, specified as the result of a sub-condition.

Types of Aggregate Functions:

Function	Description	Parameter(s)
all_distinct	Returns true if all the input values are distinct (different).	<ul style="list-style-type: none"> ▶ Any value to test. ▶ <i>(optional)</i> Any additional values to test.
all_same	Returns true if all the input values are identical.	<ul style="list-style-type: none"> ▶ Any value to test. ▶ <i>(optional)</i> Any additional values to test.
as_list	<p>Returns a list containing all the values passed in.</p> <p>Items which may be added to a list are: objects, strings, integers, booleans, reals, and geometries (2D, 3D or Measured).</p> <p>A list may not contain elements of different types.</p> <p>The list may contain duplicates.</p>	<ul style="list-style-type: none"> ▶ Any item to be added to the list. ▶ <i>(optional)</i> Any additional items to add to the list.

Function	Description	Parameter(s)
avg	Returns the average (mean) of the input values.	▶ A numerical value.
count	Counts the number of objects or the number of non-null parameter values. If no parameters are passed, this function counts the number of objects traversed. Otherwise it counts the number of objects for which any of the parameters has a non-null value.	▶ <i>(optional)</i> Any values to count.
count_distinct	Counts the number of distinct groups of the specified value or values.	▶ Any value to count. ▶ <i>(optional)</i> Any additional values to count.
intersection	Returns the intersection of the input geometries.	▶ Any geometry.
max	Returns the maximum value of the parameter over each object traversed. The values may be either numerical values, booleans or strings. Boolean values are regarded as being equal to either 1 for true or 0 for false. Strings are compared lexicographically. If numbers are compared to strings, they are converted to strings before being compared.	▶ A numerical, boolean or string value.

Function	Description	Parameter(s)
mbr	<p>Returns the MBR (minimum bounding rectangle) of a set of geometries.</p> <p>The final result is the smallest rectangle, with sides parallel to the X and Y axes, containing all the geometry values passed in.</p>	<p>► Any geometry.</p>
min	<p>Returns the minimum value of the parameter over each object traversed.</p> <p>The values may be either numerical values, booleans or strings.</p> <p>Boolean values are regarded as being equal to either 1 for true or 0 for false.</p> <p>Strings are compared lexicographically. If numbers are compared to strings, they are converted to strings before being compared.</p>	<p>► A numerical, boolean or string value.</p>
sum	<p>Returns the sum of the parameter value over each object traversed.</p> <p>The values may be either numerical values, booleans or strings.</p> <p>Boolean values are regarded as being equal to either 1 for true or 0 for false.</p> <p>Strings are added together using concatenation. If numbers are added to strings, they are converted to strings first.</p>	<p>► A numerical, boolean or string value.</p>
union	<p>Returns the union of the input geometries.</p>	<p>► Any geometry.</p>


Built-in Functions

A built-in function returns a value calculated from one or more value parameters.

The following Built-in functions can be used within 1Integrate for ArcGIS.


Geometric Functions

Function	Description	Parameter(s)
angle	<p>Angles are calculated between two line segments, which do not need to converge or touch.</p> <p>Returns an acute angle in either degrees or radians.</p> <p>A negative value is returned for errors.</p>	<ul style="list-style-type: none"> ▶ A two point line segment. ▶ A two point line segment. ▶ <i>(optional)</i> A boolean flag set to true for degrees. Default is false, for radians.

Function	Description	Parameter(s)
area	<p>Returns the area of a geometry in dataset units.</p> <p>If the geometry is a point or a line the result will be 0.</p> <p>If the geometry is a complex geometry, the result will be the sum of the areas of each simple area in the complex geometry.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<p>► The input geometry.</p>


Function	Description	Parameter(s)
boundary	<p>Returns the boundary of a line or area.</p> <ul style="list-style-type: none"> ▶ The boundary of a line is a complex point geometry containing the line end points. ▶ The boundary of an area is a geometry containing all the rings in the area. This will be a simple line if the area has only an outer ring, otherwise it will be a complex line. <p>This function supports 3D and 2D geometries.</p> <p>For point geometries, a Null value is returned.</p>	<ul style="list-style-type: none"> ▶ The geometry for which the boundary is required.


Function	Description	Parameter(s)
buffer	Returns an area geometry representing a buffer zone of a fixed distance from the boundary of a geometry.	<ul style="list-style-type: none"> ▶ The geometry to buffer. ▶ The distance from the input geometry that the buffer zone will extend to. ▶ <i>(optional)</i> The length of the line segments to use to build up the buffer zone. ▶ <i>(optional)</i> The square end proportion. The square end proportion determines how far from the end of the line to create the square end. A negative value produces rounded corners and a positive value produces a buffer with square end. ▶ <i>(optional)</i> The mitre truncation proportion. A negative mitre truncation proportion value produces rounded corners. A zero value produces a true bevel which would cut off the corner with a straight line. A value greater than zero would extend the bevel away from the geometry by a proportion of the buffer distance.


Function	Description	Parameter(s)
convex_hull	<p>Returns the smallest convex area geometry that contains the input geometry.</p> <p>The return value will be a simple area geometry with no inner rings.</p> <div> Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions.</div>	<p>► The input geometry.</p>


Function	Description	Parameter(s)
count_inner_rings	<p>Counts the clockwise digitised rings of a simple or complex geometry.</p> <ul style="list-style-type: none"> ▶ For a simple area this is the number of inner rings. ▶ For a complex geometry this is the sum of inner ring counts for all the simple area components. ▶ For a point or line geometry this is 0. ▶ If the object passed was not a geometry, returns null. 	<ul style="list-style-type: none"> ▶ The input geometry.
count_parts	<p>Returns the number of parts in a geometry.</p> <ul style="list-style-type: none"> ▶ For a clear geometry this is 0. ▶ For a simple geometry this is 1. ▶ A complex geometry will have one or more parts. ▶ If the object passed was not a geometry, returns null. 	<ul style="list-style-type: none"> ▶ The input geometry.


Function	Description	Parameter(s)
count_vertices	<p>Returns the total number of vertices in a simple or complex geometry.</p> <p>The number of points in a ring does not include a duplicate point that closes the ring.</p> <ul style="list-style-type: none"> ▶ For an area geometry, the sum of the vertex counts for each of its rings will be returned. ▶ For a complex geometry, the sum of the vertex counts over each of its components will be returned. ▶ For anything other than a geometry an exception will be thrown. 	<ul style="list-style-type: none"> ▶ The input geometry.


Function	Description	Parameter(s)
create_geometry_from_wkt	<p>Creates a geometry from a geometric well-known text string (WKT).</p> <p>Supported types are:</p> <ul style="list-style-type: none"> ▶ Point ▶ MultiPoint ▶ LineString ▶ MultiLineString ▶ Polygon ▶ MultiPolygon <p>Empty geometries, or those which specify their dimension as Z, M or ZM, are not supported.</p>	<ul style="list-style-type: none"> ▶ A geometric WKT string.
difference	<p>Returns the difference between two geometries; all the parts of the first geometry that are not in the second geometry.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p> Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions.</p> </div>	<ul style="list-style-type: none"> ▶ The first geometry. ▶ The second geometry (to be subtracted from the first geometry).



Function	Description	Parameter(s)
difference_between_bearings	Returns an angle in degrees between 0 and 90, representing the difference between two bearings.	<ul style="list-style-type: none"> ▶ The first bearing: An angle in radians between $-\pi$ and π. ▶ The second bearing: An angle in radians between $-\pi$ and π.
difference_by_dimension	<p>A geometry equal to geometry1 AND NOT geometry2.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The required dimensionality of the result. <ul style="list-style-type: none"> ▶ 0 - for <code>GeomCombineResultType.POINTS</code> ▶ 1 - for <code>GeomCombineResultType.LINES</code> ▶ 2 - for <code>GeomCombineResultType.AREAS</code> ▶ The first geometry. ▶ The second geometry (to be subtracted from the first geometry).
dimension	Returns the dimension of the input geometry (0 to 2) or -1 if input is null.	<ul style="list-style-type: none"> ▶ A geometry.


Function	Description	Parameter(s)
distance	<p>Returns the distance in dataset units at the closest point of approach of two geometries.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ A geometry. ▶ Another geometry.
douglas_peucker	<p>Returns a simplified version of a geometry formed by applying the Douglas-Peucker smoothing algorithm to each piece of line-work in the geometry.</p> <p>The result will be a geometry which approximates the original geometry using fewer vertices.</p> <p>The line-work of the resulting geometry will lie entirely within the specified tolerance of the original geometry's line-work.</p>	<ul style="list-style-type: none"> ▶ The geometry to simplify. ▶ The required tolerance (must be a positive number).


Function	Description	Parameter(s)
drag_vertex	<p>Returns a geometry formed by moving a specified vertex on a simple line geometry to a new location.</p> <p>It uses a scale and rotate algorithm to try to preserve the shape of the geometry on either side of the vertex being moved.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ A simple line geometry. ▶ A simple point geometry identifying the vertex on the line to be moved. ▶ A simple point geometry identifying the new location for the vertex.
end_of	Returns a point geometry at the location of the end of a simple line geometry.	<ul style="list-style-type: none"> ▶ A simple line geometry.


Function	Description	Parameter(s)
ends_of	<p>Returns the endpoints of a simple line geometry.</p> <p>If the line is closed, the result will be a simple point geometry. Otherwise the result will be a complex geometry containing two points.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<p>► A simple line geometry.</p>


Function	Description	Parameter(s)
find_duplicates	<p>Returns a collection of point geometries, representing any duplicate points within a geometry.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The geometry to test. ▶ <i>(optional)</i> The tolerance (in dataset units) for points to be classed as duplicate. <p>If not specified uses the default geometric tolerance.</p>
find_kickbacks	<p>Returns a collection of point geometries, representing the locations of any kickbacks within a geometry.</p> <p>Kickbacks are a type of geometric error where a line segment changes direction twice by approximately 180 degrees (like a z shape) to repeat part of the line. They are usually detected with larger angles than spikes which is why there is a separate built-in to find them.</p>	<ul style="list-style-type: none"> ▶ The geometry to test. ▶ <i>(optional)</i> The maximum value for the sine of the angles in the kickback. <p>If omitted, this defaults to the sine of 1 degree.</p> <ul style="list-style-type: none"> ▶ <i>(optional)</i> The maximum width of the kickback. <p>If omitted, kickbacks with any width will be returned.</p>


Function	Description	Parameter(s)
find_self_intersections	Returns a multi-point geometry representing the points at which a line, or the rings of an area, self-intersect.	<ul style="list-style-type: none"> ▶ The geometry to test.
find_small_rings	<p>Returns a descriptor for the complex line geometry containing small rings.</p> <p>One or more of the following criteria may be used to determine if a ring is to be considered small:</p> <ul style="list-style-type: none"> ▶ The length of the diagonal if the ring's MBR. ▶ The area of the ring. ▶ The ratio of the ring's area to the square of its perimeter. <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The geometry to test. ▶ The maximum MBR diagonal length. ▶ <i>(optional)</i> The maximum ring area. ▶ <i>(optional)</i> The maximum area to square perimeter ratio. <div>  Note: If any of these parameters is negative it is ignored. </div>



Function	Description	Parameter(s)
find_spikes	<p>Returns a descriptor for the point complex geometry containing spike points.</p> <p>A spike is defined to be three consecutive points (A, B, C) such that:</p> <ol style="list-style-type: none"> 1. The distance AB is less than the distance BC. 2. The sine of the angle ABC is less than a maximum value which may be specified by the second parameter. 3. (Optionally) the distance AB is less than a maximum "length" value specified by the third parameter. 	<ul style="list-style-type: none"> ▶ The geometry to test. ▶ <i>(optional)</i> The maximum value for the sine of the angle in the spike (a real number in the range [0, 1]). <div>  Note: If omitted, this defaults to the sine of 1 degree (approximately 0.017). </div> <ul style="list-style-type: none"> ▶ <i>(optional)</i> The maximum length of the spike.



Function	Description	Parameter(s)
get_job_extent	<p>Obtains the extent of the current session.</p> <div>  Note: This does <i>not</i> calculate the minimum bounding rectangle (MBR) of the loaded data. </div> <p>This is calculated from the following (in order of priority):</p> <ul style="list-style-type: none"> ▶ If the session is partitioned, the extent of the partition. ▶ If an extent such as a bounding box or polygon has been specified for a session, the specified extent. ▶ If there are any extents present in the spatial metadata for the loaded data, the MBR of all such extents. ▶ A default, arbitrary extent (the whole world if data is in degrees, else a 10,000 width square). 	



Function	Description	Parameter(s)
get_point	<p>Returns a point guaranteed to lie on the specified geometry.</p> <p>The point is not necessarily near to the centre of the geometry, but is guaranteed to lie inside it.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<p>► Any geometry.</p>


Function	Description	Parameter(s)
get_x	<p>Returns the x co-ordinate of a point on a simple or complex geometry.</p> <p>For a point geometry the return will be the x co-ordinate of that geometry.</p> <p>For other types of geometry the return will be the x co-ordinate of an arbitrary point on the input geometry.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<p>► Any geometry.</p>



Function	Description	Parameter(s)
get_y	<p>Returns the y co-ordinate of a point on a simple or complex geometry.</p> <p>For a point geometry the return will be the y co-ordinate of that geometry.</p> <p>For other types of geometry the return will be the y co-ordinate of an arbitrary point on the input geometry.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<p>► Any geometry.</p>
has_duplicates	<p>Tests to see if a geometry has any consecutive coincident vertices.</p> <p>Returns a Boolean value, true if the geometry has any consecutive coincident vertices and false if it does not.</p>	<p>► Any geometry.</p>


Function	Description	Parameter(s)
has_kickbacks	<p>Checks whether a geometry has any kickbacks (a type of geometric error where a line segment changes direction twice by approximately 180 degrees to repeat part of the line).</p> <p>Kickbacks are also known as snap-backs.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The geometry to test. ▶ <i>(optional)</i> The maximum value for the sine of the angles in the kickback. <div>  Note: If omitted, this defaults to the sine of 1 degree. </div> ▶ <i>(optional)</i> The maximum width of the kickback.


Function	Description	Parameter(s)
has_small_rings	<p>Checks whether a geometry has any small rings (pig tails).</p> <p>Returns true if the geometry has small rings and false if it does not.</p> <p>One or more of the following criteria may be used to determine if a ring is to be considered small:</p> <ul style="list-style-type: none"> ▶ The length of the diagonal if the ring's MBR. ▶ The area of the ring. ▶ The ratio of the ring's area to the square of its perimeter. <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The geometry to test. ▶ The maximum MBR diagonal length. ▶ <i>(optional)</i> The maximum ring area. ▶ <i>(optional)</i> The maximum area to square perimeter ratio. <div>  Note: If any of these parameters is negative it is ignored. </div>


Function	Description	Parameter(s)
has_spikes	<p>Checks whether a geometry has any spikes.</p> <p>Returns true if the geometry has spikes and false if it does not.</p> <p>A spike is defined to be three consecutive points (A, B, C) such that:</p> <ol style="list-style-type: none"> 1. The distance AB is less than the distance BC. 2. The sine of the angle ABC is less than a maximum value which may be specified by the second parameter. 3. (Optionally) the distance AB is less than a maximum "length" value specified by the third parameter. <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The geometry to test. ▶ <i>(optional)</i> The maximum value for the sine of the angle in the spike (a real number in the range [0, 1]). <div>  Note: If omitted, this defaults to the sine of 1 degree (approximately 0.017). </div> <ul style="list-style-type: none"> ▶ <i>(optional)</i> The maximum length of the spike.

Function	Description	Parameter(s)
height	Returns the height of the nearest vertex for a 3D geometry.	<ul style="list-style-type: none"> ▶ A 3D geometry. ▶ <i>(optional)</i> A 2D geometry specifying the point on the 3D geometry whose height is to be returned. <div>  Note: If this parameter is omitted, then a vertex on the 3D geometry is chosen at random. </div>
inner_rings	<p>Returns the inner rings of a simple area geometry, or the inner rings of all areas in a complex geometry.</p> <p>If there is only one inner ring, that ring is returned as a simple line. If there are several, they are returned as components of a complex line geometry.</p> <p>Returns null if applied to a null geometry, a non-area geometry, an area geometry without inner rings or a complex geometry containing areas with no inner rings.</p> <p>The result has the same dimensionality (2D or 2.5D) as the input geometry.</p>	<ul style="list-style-type: none"> ▶ A simple or complex 2D or 2.5D geometry.

Function	Description	Parameter(s)
intersection	<p>Returns the intersection of two or more geometries - the parts in common between all of them.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ Any geometry. ▶ Another geometry to intersect with the first. ▶ <i>(optional)</i> Additional geometries to intersect.
intersection_by_dimension	<p>Returns a geometry equal to the intersection of all the input geometries, or null if the intersection is empty.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The required dimensionality of the result. <ul style="list-style-type: none"> ▶ 0 - for <code>GeomCombineResultType.POINTS</code> ▶ 1 - for <code>GeomCombineResultType.LINES</code> ▶ 2 - for <code>GeomCombineResultType.AREAS</code> ▶ Any geometry. ▶ Another geometry to intersect with the first. ▶ <i>(optional)</i> Additional geometries to intersect.


Function	Description	Parameter(s)
is_aligned	<p>Tests whether two line geometries are aligned.</p> <p>Returns true if the geometries are aligned and false otherwise.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ First simple line geometry (must not be null). ▶ Second simple line geometry (must not be null).


Function	Description	Parameter(s)
is_ boundary_ left	<p>Tests whether a point, line or area geometry is to the left of a line geometry with respect to the direction of the line geometry.</p> <p>Returns true if the boundary left relationship holds and false otherwise.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The line geometry. ▶ The area geometry.

Function	Description	Parameter(s)
is_ boundary_ right	<p>Tests whether a point, line or area geometry is to the right of a line geometry with respect to the direction of the line geometry.</p> <p>Returns true if the boundary right relationship holds and false otherwise.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The line geometry. ▶ The area geometry.
is_closed	<p>Tests whether a geometry is closed. Returns true if the geometry is a simple area or if it is a simple line with coincident end points.</p>	<ul style="list-style-type: none"> ▶ Any geometry.

Function	Description	Parameter(s)
is_downhill	<p>Tests whether a 3D line geometry is digitised in a downhill direction.</p> <p>Returns a Boolean value, true if the geometry is a 3D line that slopes downwards and false otherwise.</p> <p>Returns null if parameter 1 is not a 3D simple line geometry.</p> <p>An optional tolerance may be supplied to use when comparing the heights of vertices on the line. The line is considered to be downhill if each vertex is lower than all the preceding vertices, to within the specified tolerance.</p>	<ul style="list-style-type: none"> ▶ A 3D line geometry to test. ▶ <i>(optional)</i> A numerical value specifying the tolerance to apply to the line's z values.

Function	Description	Parameter(s)
is_level	<p>Tests whether a 3D line geometry has a constant height along its length.</p> <p>Returns a Boolean value, true if the geometry is a 3D line is level and false otherwise.</p> <p>Returns null if parameter 1 is not a 3D simple line geometry.</p> <p>An optional tolerance may be supplied for use when comparing the heights of vertices on the line. The line is considered to be level if the maximum height minus the minimum height along the line is less than or equal to the specified tolerance.</p>	<ul style="list-style-type: none"> ▶ A 3D line geometry to be tested. ▶ <i>(optional)</i> A numerical value specifying the tolerance to apply to the line's z values.
is_noded	<p>Tests if two line geometries intersect at common vertices.</p> <p>Returns true if the geometries are noded and false otherwise.</p>	<ul style="list-style-type: none"> ▶ The first line geometry. ▶ The second line geometry. ▶ <i>(optional)</i> An optional boolean value indicating for 3D geometries whether or not to check z values at intersections (defaults to true).


Function	Description	Parameter(s)
is_simple	<p>Tests to see if a geometry is simple according to the OGC definition.</p> <p>Returns a Boolean value, true if the geometry is simple according to the OGC definition and false if it is not.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ Any geometry.
is_uphill	<p>Returns true if a 3D line slopes upwards, with an optional tolerance for comparing the heights of vertices on the line.</p> <p>The line is considered to be uphill if each vertex is higher than all the preceding vertices, to within tolerance.</p>	<ul style="list-style-type: none"> ▶ A 3D line geometry to test. ▶ <i>(optional)</i> A numerical value specifying the tolerance to apply to the line's z values.


Function	Description	Parameter(s)
is_valid	<p>Tests to see if a geometry is valid according to the OGC definition.</p> <p>Returns a Boolean value, true if the geometry is valid according to the OGC definition and false if it is not.</p>	<ul style="list-style-type: none"> ▶ Any geometry.
line	<p>Returns a line geometry constructed from points supplied in parameters.</p> <div>  <p>Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions.</p> </div>	<ul style="list-style-type: none"> ▶ The start of the line as a simple point geometry. ▶ The second vertex of the line as a simple point geometry. ▶ <i>optional</i>) Subsequent vertices in the line as point geometries .
line_bearing_at_point	<p>Returns the angle of the line at a point between -pi and pi, measured in radians anti-clockwise from east (positive x axis).</p>	<ul style="list-style-type: none"> ▶ The line geometry for which to measure the angle. ▶ The point geometry for the point that is on or near the point on the line at which the angle is to be measured.


Function	Description	Parameter(s)
line_length	<p>Returns the length of a geometry in dataset units.</p> <p>The length of a point or an area is 0.</p> <p>The length of a complex geometry is the sum of the lengths of its component simple line geometries.</p>	<ul style="list-style-type: none"> ▶ Any geometry.
line_segment	<p>Returns a segment cut out of a simple line geometry, or null if the distance parameters are out of range.</p> <p>The start and end points of the segment are specified by distances along the line, measured from its start. A negative value indicates a distance from the line end instead of the start.</p> <p>With a 3D line geometry, distances are in plan; if the start or end of the segment is between two vertices with height values, the segment endvertex is given an interpolated height.</p>	<ul style="list-style-type: none"> ▶ Any geometry. ▶ A distance along the line for the segment start point. ▶ A distance along the line for the segment end point.
make_2d	<p>Returns a 2D geometry.</p> <p>If the argument is not valid, then null is returned.</p>	<ul style="list-style-type: none"> ▶ The geometry to convert to 2D.


Function	Description	Parameter(s)
make_3d	<p>Returns a 3D geometry.</p> <p>If the argument is not valid, then null is returned.</p>	<ul style="list-style-type: none"> ▶ The geometry to convert to 3D. ▶ (optional) The x-coordinate value for the first 3D point. ▶ (optional) The y-coordinate value for the first 3D point. ▶ (optional) The z-coordinate value for the first 3D point. ▶ (optional) The x-coordinate value for the second 3D point. ▶ (optional) The y-coordinate value for the second 3D point. ▶ (optional) The z-coordinate value for the second 3D point. ▶ (optional) The x-coordinate value for the third 3D point. ▶ (optional) The y-coordinate value for the third 3D point. ▶ (optional) The z-coordinate value for the third 3D point.



Function	Description	Parameter(s)
max_deflection_angle	<p>Returns the maximum deflection angle (in degrees) between adjacent line segments in a geometry.</p> <ul style="list-style-type: none"> ▶ For a point this is always 0. ▶ For an area it the maximum deflection in any of its rings. ▶ For a complex geometry it is the maximum deflection in any of its simple parts. ▶ For 3D geometries, the heights are ignored. ▶ For Measured geometries, the measures are ignored. 	<ul style="list-style-type: none"> ▶ The input geometry.
max_height	<p>Returns the maximum height of a 3D geometry.</p>	<ul style="list-style-type: none"> ▶ A 3D geometry.


Function	Description	Parameter(s)
mbr	<p>Returns the minimum bounding rectangle (MBR) of one or more geometries.</p> <p>The result is the smallest rectangle, with sides parallel to the X and Y axes, that contains all the input geometries.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<p>► Any geometry.</p>
min_height	<p>Returns the minimum height of a 3D geometry.</p>	<p>► A 3D geometry.</p>

Function	Description	Parameter(s)
min_intersection_angle	<p>Returns the minimum intersection angle between two geometries.</p> <p>Returns -1 if the two geometries do not intersect.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ First input geometry. ▶ Second input geometry.

Function	Description	Parameter(s)
move_vertex	<p>Returns a geometry formed by moving a vertex on a simple line geometry.</p> <p>This function will only move one vertex on the geometry and so may leave a spike. For smoother results, consider the drag_vertex function.</p> <div>  <p>Note: For heighted or measured geometries: If the new point has a height/measure then the height/measure from the point will be applied. If the new location point is 2D then the height/measure value on the original vertex is retained.</p> </div>	<ul style="list-style-type: none"> ▶ A point or line geometry. ▶ A simple point geometry identifying the vertex on the line to be moved. ▶ A simple point geometry identifying the new location for the vertex.


Function	Description	Parameter(s)
nearest_point	<p>Returns a geometry representing the nearest point(s) on a provided geometry from an originating point geometry. This will be complex if multiple points are equally close to the originating point.</p> <p>If neither parameter is a geometry an exception is thrown.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ An originating point. ▶ A destination geometry (to search), either point, line, area, or complex whose nearest point(s) to parameter 1 are returned. ▶ <i>(optional)</i> A Boolean value, true to return an arbitrary single nearest point or false to return all nearest points as a complex geometry. The default value is false.


Function	Description	Parameter(s)
nearest_vertex	<p>Returns a point geometry at the nearest vertex on a geometry, described by the nearest known X and Y positions.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The geometry on which to find the vertex. ▶ The nearest known X position. ▶ The nearest known Y position.
offset	<p>Offsets (translates) a geometry by a fixed distance without changing its shape, size or orientation.</p>	<ul style="list-style-type: none"> ▶ The geometry to offset. ▶ The distance to move the geometry in the x direction. ▶ The distance to move the geometry in the y direction. ▶ (<i>optional</i>) The distance to move the geometry in the z direction. <div>  Note: If a z offset is specified with a 2D geometry, the result will be a 3D geometry with all heights set to the z offset value. </div>



Function	Description	Parameter(s)
outer_ring	Returns the outer ring of a simple area geometry. The result will be a closed simple line geometry.	<ul style="list-style-type: none"> ▶ A simple or complex area geometry.
perimeter	<p>Returns the perimeter of an area geometry.</p> <p>If the geometry is a point or a line, the result will be 0.</p> <p>If the geometry is a complex geometry, the result will be the sum of the perimeters of each simple area geometry in the complex geometry.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ Any geometry.
point	Builds a 2D point (if passing x and y values) or a 2.5D point (if passing x, y and z values).	<ul style="list-style-type: none"> ▶ The x-coordinate value of the point. ▶ The y-coordinate value of the point. ▶ <i>(optional)</i> The z-coordinate value of the point.

Function	Description	Parameter(s)
point_along_line	Returns a point on a line geometry at a distance along it expressed as a proportion of the total length of the line.	<ul style="list-style-type: none"> ▶ A line geometry. ▶ Proportional distance along the line, a real number in the range [0,1]. <ul style="list-style-type: none"> ▶ value ≤ 0 locates the start point of the line. ▶ value ≥ 1 locates the end point of the line. ▶ $0 < \text{value} < 1$ locates a point within the line.
point_at_projection	Returns a new point, the specified bearing and distance from the origin point.	<ul style="list-style-type: none"> ▶ The origin 2D or 3D point from which to measure the location of the new point. ▶ An angle (in degrees anti-clockwise from east) representing the bearing to the new point from the origin point. ▶ A distance from the origin point (in dataset units) at which to create the new point.
point_on_line	<p>Finds a point on a line geometry, a given distance along it from its start or end point.</p> <p>Returns null if the distance parameter is out of range.</p> <p>With a 3D line, the height will be interpolated between vertices.</p>	<ul style="list-style-type: none"> ▶ The input geometry. ▶ Distance from the start of the line. A negative value indicates distance from the end of the line.




Function	Description	Parameter(s)
polygon	<p>Forms an area geometry from a simple closed line geometry, a complex line geometry containing one or more rings, a simple area, or a complex area geometry.</p> <p>Returns null if the rings do not form a valid area geometry.</p>	<ul style="list-style-type: none"> ▶ The input geometry.
proportion_along_line	<p>Finds the nearest position on a line to a given point, and calculates its distance along the line, expressed as a proportion of the total length of the line (a real number in the range between 0 and 1).</p>	<ul style="list-style-type: none"> ▶ A line geometry. ▶ A point geometry.
remove_duplicates	<p>Removes any duplicate vertices from a geometry.</p> <p>Any occurrence of consecutive, coincident vertices is replaced with a single vertex at the same location.</p>	<ul style="list-style-type: none"> ▶ Any geometry.



Function	Description	Parameter(s)
remove_kickbacks	<p>Removes any kickbacks (a type of geometric error where a line segment changes direction twice by approximately 180 degrees to repeat part of the line) from a geometry.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The input geometry. ▶ <i>(optional)</i> The maximum value for the sine of the angles in the kickback. If omitted, this defaults to the sine of 1 degree. ▶ <i>(optional)</i> The maximum width of the kickback.



Function	Description	Parameter(s)
remove_small_rings	<p>Removes any small rings from a geometry. Returns null if all of the rings are too small.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The input geometry. ▶ The maximum MBR diagonal length. ▶ <i>(optional)</i> The maximum ring area. ▶ <i>(optional)</i> The maximum area to square perimeter ratio.



Function	Description	Parameter(s)
remove_spikes	<p>Removes any spikes from a geometry.</p> <p>A spike is defined to be three consecutive points (A, B, C) such that:</p> <ol style="list-style-type: none"> 1. The distance AB is less than the distance BC. 2. The sine of the angle ABC is less than a maximum value which may be specified by the second parameter. 3. (Optionally) the distance AB is less than a maximum "length" value specified by the third parameter. <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The input geometry. ▶ <i>(optional)</i> The maximum value for the sine of the angle in the spike. <div>  Note: If omitted, this defaults to the sine of 1 degree. </div> <ul style="list-style-type: none"> ▶ <i>(optional)</i> The maximum length of the spike.

Function	Description	Parameter(s)
reverse_line	<p>Reverses the order of the vertices in a simple line geometry.</p> <p>Returns a copy of the line geometry, with the same vertices but running in the opposite direction.</p> <p>Returns null if parameter is not a simple line geometry.</p>	<ul style="list-style-type: none"> ▶ Any simple line geometry.
round_geom	<p>Returns a rounded geometry.</p>	<ul style="list-style-type: none"> ▶ A geometry to round. ▶ Precision to round to, as an integer. ▶ <i>(optional)</i> Whether or not to check if the geometry is valid after rounding. Default is true.
rule_hotspot_geometry	<p>A collection geometry containing the hotspots from this feature for the triggering Rule, or null.</p> <p>If an optional default is provided, will return the default instead of null.</p>	<ul style="list-style-type: none"> ▶ <i>(optional)</i> A default value if no hotspots are found. The default value must be geometric.
scale_and_rotate	<p>Returns a 2D copy of the input geometry scaled and rotated.</p>	<ul style="list-style-type: none"> ▶ The geometry to scale and rotate. It can be 2D or 3D, of any type. ▶ The Point geometry to scale and rotate about, must be a 2D or 3D simple point. ▶ The scale factor to use. ▶ The angle to rotate the object, in radians.

Function	Description	Parameter(s)
segment	<p>Returns a segment from a line as two consecutive vertices.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The line geometry from which the segment is taken ▶ The index of the vertex to extract, or 0 for the last segment. <div>  Note: A negative index will return the segment counting from the last segment. </div>
segments	<p>Obtains the segments from a geometry, returning as a Complex Line.</p> <p>If no segments are present, this geometry will be clear.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The geometry from which to take the segments.


Function	Description	Parameter(s)
start_of	Returns a point geometry at the location of the start of a simple line geometry.	<ul style="list-style-type: none"> ▶ A simple line geometry.
symmetric_difference	<p>Returns a geometry equal to geometry1 XOR geometry2.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The first geometry. ▶ The second geometry (to be XOR'd with the first geometry).
symmetric_difference_by_dimension	<p>Returns a geometry equal to geometry1 XOR geometry2.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The required dimensionality of the result. <ul style="list-style-type: none"> ▶ 0 - for <code>GeomCombineResultType.POINTS</code> ▶ 1 - for <code>GeomCombineResultType.LINES</code> ▶ 2 - for <code>GeomCombineResultType.AREAS</code> ▶ The first geometry. ▶ The second geometry (to be XOR'd with the first geometry).

Function	Description	Parameter(s)
true_distance	Returns the distance along the surface of the earth between the 2 points.	<ul style="list-style-type: none"> ▶ The longitude of point 1 in degrees. ▶ The latitude of point 1 in degrees. ▶ The longitude of point 2 in degrees. ▶ The latitude of point 2 in degrees. ▶ The earth spheroid model to use. <div>  Note: Allowed values are: 1: WGS-84 2: GRS-80 3: Airy (1830) 4: Int'l 1924 5: Clarke (1880) 6: GRS-67 </div>
union	Returns the union of two or more geometries. <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ Any geometry. ▶ Another geometry to combine with the first. ▶ (<i>optional</i>) Additional geometries to add to the union.




Function	Description	Parameter(s)
union_by_dimension	<p>Returns a geometry equal to the union of all the input geometries.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The required dimensionality of the result. <ul style="list-style-type: none"> ▶ 0 - for <code>GeomCombineResultType.POINTS</code> ▶ 1 - for <code>GeomCombineResultType.LINES</code> ▶ 2 - for <code>GeomCombineResultType.AREAS</code> ▶ Any geometry. ▶ Another geometry to combine with the first.
vertices	<p>Obtains the vertices of a geometry, returned as a Complex Point. If no vertices are present, this geometry will be clear.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The geometry from which to take the vertices.

Measured Geometry Functions

Function	Description	Parameter(s)
get_measure_maximum	<p>Returns the maximum measure value of a geometry.</p> <p>The maximum value may not necessarily be found at the start/end of the geometry.</p> <p>Measures can be null so the maximum may be null. If the input parameter is null, the value returned will be null.</p> <p>If the input geometry is not a measured geometry, an exception will be thrown.</p>	<p>► A measured geometry.</p>
get_measure_minimum	<p>Returns the minimum measure value of a geometry.</p> <p>The minimum value may not necessarily be found at the start/end of the geometry.</p> <p>Measures can be null so the minimum may be null. If the input parameter is null, the value returned will be null.</p> <p>If the input geometry is not a measured geometry, an exception will be thrown.</p>	<p>► A measured geometry.</p>
is_measured_line_monotonic_increasing	<p>Returns a boolean to indicate if the input measured geometry is a simple line and measured with values increasing in the digitised direction along its length.</p> <p>If the input geometry is not a simple measured line, an exception will be thrown.</p>	<p>► A measured line geometry.</p> <p>► <i>(optional)</i> A boolean to indicate if null measure values should be ignored.</p>

Function	Description	Parameter(s)
line_ segment_ from_ measures	<p>Returns the segment of linework between 2 particular measure values.</p> <p>The returned line geometry will be in direction based on the order of the specified measure values, rather than the geometric order of the input line.</p>	<ul style="list-style-type: none"> ▶ A measured simple line geometry. ▶ The value of the measure to signify the start point for the output segment. ▶ The value of the measure to signify the end point for the output segment. ▶ <i>(optional)</i> The tolerance for measures being outside the range present (in measure units). <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  Note: If not specified then the horizontal spatial tolerance will be used which may not be applicable for the units of the measures. </div>

Function	Description	Parameter(s)
measure_from_point	<p>Returns the measure value located at the nearest position on the input line to the point geometry.</p> <p>Linear interpolation between stored measure values (located at vertices) will be used to identify the exact measure value along the input line where the nearest location (in 2D plan form) to the point is found.</p> <p>If the measure value at the specified vertex is null or if either of the measure values adjacent to the point geometry are null, the return value will be null.</p> <p>If the input geometry is not a simple measured line, an exception will be thrown.</p>	<ul style="list-style-type: none"> ▶ A measured line geometry. ▶ The location from which the measure will be retrieved. <p>If the input geometry is a simple measured line, the location can be defined as either the nearest point on line to the specified point or as the 1-based integer index of the vertex within the line.</p> <p>If the input geometry is a complex measured line, the location must be defined as the nearest point on the line.</p>
measured_point	Builds a measured point from x, y and m values.	<ul style="list-style-type: none"> ▶ The x-coordinate value of the point. ▶ The y-coordinate value of the point. ▶ The measure value of the point. May be null.

Function	Description	Parameter(s)
point_from_measure	<p>Returns a point geometry located at the first position on the input line where the measure value is found.</p> <p>Linear interpolation between stored measure values (located at vertices) will be used to identify the exact position along the line where the specified measure value is found.</p> <p>If the input geometry is not a simple measured line, an exception will be thrown.</p> <p>If the measure value is not found in the input geometry, an exception will be thrown.</p>	<ul style="list-style-type: none"> ▶ A measured simple line geometry. ▶ The measure value to find (double). ▶ <i>(optional)</i> The tolerance for measures being outside the range present (in measure units). <div>  Note: If not specified then the horizontal spatial tolerance will be used which may not be applicable for the units of the measures. </div>
set_measure	<p>Returns a copy of the input geometry with a measure set at the specified location.</p> <div>  Note: This will overwrite any existing measure value at that location. </div> <p>Setting a null measure value is supported.</p>	<ul style="list-style-type: none"> ▶ A measured geometry. ▶ The vertex on which to set the measure. <div>  Note: This is defined as either the nearest vertex to the specified point or as the 1-based integer index of the vertex within the line. </div>

Conversion Functions

Function	Description	Parameter(s)
to_degrees	Converts an angle specified in radians to degrees.	<ul style="list-style-type: none"> ▶ A numerical value representing an angle in radians.

Function	Description	Parameter(s)
to_integer	<p>Converts a numerical, boolean or string value to an integer.</p> <p>Any decimal digits present in the number after the decimal point are removed.</p> <p>The boolean values true and false are converted into 1 and 0 respectively.</p>	<p>► A numerical, boolean, string or timestamp value.</p>
to_radians	<p>Converts an angle specified in degrees to radians.</p>	<p>► A numerical value representing an angle in degrees.</p>
to_real	<p>Converts a numerical, boolean or string value to a real (floating point) number.</p>	<p>► A numerical, boolean or string value.</p>
to_string	<p>Converts any value to a string.</p>	<p>► Any value.</p>


Mathematical Functions

Function	Description	Parameter(s)
abs	<p>Returns the absolute value of the input parameter.</p> <p>If the parameter is negative, it is returned with its sign reversed so that it becomes positive. Otherwise it is returned unchanged.</p>	<p>► Any numerical value.</p>
acos	<p>Returns the inverse cosine in radians (0.0 to pi).</p>	<p>► A numerical value in the range [-1,1].</p>
asin	<p>Returns the inverse sine in radians. The result will be an angle in the range between -pi/2 and pi/2.</p>	<p>► A numerical value in the range [-1,1].</p>
atan	<p>Returns the inverse tangent in radians. The result will be an angle in the range between -pi/2 and pi/2.</p>	<p>► Any numerical value.</p>

Function	Description	Parameter(s)
atan2	<p>Converts Cartesian x and y co-ordinates to polar co-ordinates and returns the polar angle, measured counter-clockwise from the positive x-axis.</p> <p>The result will be an angle in radians in the range between $-\pi$ and π.</p>	<ul style="list-style-type: none"> ▶ The x-coordinate value. ▶ The y-coordinate value.
ceil	Rounds a numerical value up to the nearest integer value greater than or equal to the input value.	▶ Any numerical value.
cos	Returns the cosine of an angle in radians (in the range $[-1,1]$).	▶ Any numerical value.
exp	Returns the inverse natural logarithm of a number (i.e Euler's number raised to the power of the input parameter, e^x).	▶ Any numerical value.
floor	Rounds a numerical value down to the nearest integer value less than or equal to the input value.	▶ Any numerical value.
is_infinite	<p>Tests if a number is infinite in magnitude.</p> <p>Returns true if the number is positive or negative infinity, and false otherwise.</p>	▶ Any numerical value.
is_NaN	<p>The special numerical value "not a number" results from certain mathematical operations such as dividing 0.0 by 0.0, which cannot be computed.</p> <p>Returns true if the number has the special value "not a number" and false if it is a valid (finite or infinite) number.</p>	▶ Any numerical value.
log	Returns the natural logarithm (base e) of a number.	▶ A numerical value greater than 0.

Function	Description	Parameter(s)
log10	Returns the logarithm to base 10 of a number.	<ul style="list-style-type: none"> ▶ A numerical value greater than 0.
max	Returns the largest of 2 or more values.	<ul style="list-style-type: none"> ▶ A numerical, boolean or string value. ▶ Another numerical, boolean or string value. ▶ <i>(optional)</i> Additional numerical, boolean or string values.
min	Returns the smallest of 2 or more values.	<ul style="list-style-type: none"> ▶ A numerical, boolean or string value. ▶ Another numerical, boolean or string value. ▶ <i>(optional)</i> Additional numerical, boolean or string values.
pow	Returns the value of one number raised to the power of another number (a^b).	<ul style="list-style-type: none"> ▶ Any numerical value (a). ▶ Any numerical value (b).
round	Returns a number rounded to the nearest integer value.	<ul style="list-style-type: none"> ▶ A numerical value (a). ▶ <i>(optional)</i> Number of decimal places to round to (b). If omitted, rounding is to the nearest integer.
sin	Returns the sine of an angle in radians (in the range [-1,1]).	<ul style="list-style-type: none"> ▶ A numerical value representing an angle in radians.
sqrt	Returns the (positive) square root of a number.	<ul style="list-style-type: none"> ▶ A numerical value greater than 0.
tan	Returns the tangent of an angle (specified in radians).	<ul style="list-style-type: none"> ▶ A numerical value representing an angle in radians.

Bit Manipulation Functions

Function	Description	Parameter(s)
bit_and	Returns the bitwise AND of two integers. Each bit in the binary expansion of the result will only be set if both of the corresponding bits in the two input numbers are also set.	<ul style="list-style-type: none"> ▶ An integer value. ▶ Another integer value.
bit_not	Returns the bitwise complement of an integer. Each bit in the binary expansion of the result will be the opposite of the corresponding bit in the input number.	<ul style="list-style-type: none"> ▶ An integer value.
bit_or	Returns the bitwise OR of two integers. Each bit in the binary expansion of the result will be set if either of the corresponding bits in the two input numbers is set.	<ul style="list-style-type: none"> ▶ An integer value. ▶ Another integer value.
bit_shift	Returns an integer value computed by shifting the binary bits of the input value. <div>  Note: Positive values shift it to the left and negative values shift it to the right. </div>	<ul style="list-style-type: none"> ▶ An integer value whose bits are to be shifted. ▶ An integer value specifying how far to shift the bits of the first parameter.
bit_xor	Returns the bitwise XOR of two integers. Each bit in the binary expansion of the result will be set if the corresponding bits in the two input numbers are different from one another.	<ul style="list-style-type: none"> ▶ An integer value. ▶ Another integer value.

String Functions

Function	Description	Parameter(s)
index_of	Searches in a string for occurrences of another string and returns the index of the first match found (where the first character is at index 0). Returns -1 if the string could not be found.	<ul style="list-style-type: none"> ▶ The string in which to search. ▶ The string to search for. ▶ <i>(optional)</i> The 0-based index in the first string to start the search from. If omitted, the search starts from the beginning of the first string.
jaro_winkler_similarity	Returns a double value representing the Jaro-Winkler similarity score between two words. If both strings are null or empty then they are considered an exact match.	<ul style="list-style-type: none"> ▶ The first word to compare. ▶ The second word to compare. ▶ <i>(optional)</i> Ignore case. If true (default), words are converted to upper case before comparison. ▶ <i>(optional)</i> Length of maximum common prefix in the range 0 to 4. If null default 4 is used. ▶ <i>(optional)</i> Scaling factor in the range 0 to 0.25 indicating how much the score is adjusted upwards for having common prefixes. If null, default 0.1 is used. ▶ <i>(optional)</i> Boost threshold to adjust the Jaro distance with the Winkler modification to give emphasis to common prefixes. If null default 0.7 is used.
length	Returns the length of a string (number of characters).	<ul style="list-style-type: none"> ▶ Any string.

Function	Description	Parameter(s)
levenshtein_distance	Returns an integer for the difference, 0 if identical.	<ul style="list-style-type: none"> ▶ The first string. ▶ The second string.
re_search	<p>Performs a regular expression search for a Java-style regular expression inside a string.</p> <p>If a match is found, the matching string (a substring of the first string) is returned. Otherwise null is returned.</p>	<ul style="list-style-type: none"> ▶ The string to search in. ▶ A string containing the regular expression to search for.
re_subs_all	Replaces all occurrences of patterns matching a Java-style regular expression in a string with the specified replacement string.	<ul style="list-style-type: none"> ▶ The string to search and replace in. ▶ A string containing the regular expression to search for. ▶ The replacement string, for each match found for the regular expression.
re_subs_first	Replaces the first occurrence of a pattern matching a Java-style regular expression in a string with the specified replacement string.	<ul style="list-style-type: none"> ▶ The string to search and replace in. ▶ A string containing the regular expression to search for. ▶ The replacement string, for the first match found for the regular expression.
soundex	Returns the soundex code for the word, or null if the word cannot be mapped to a soundex.	<ul style="list-style-type: none"> ▶ The word (string) to be tested.


Function	Description	Parameter(s)
substring	Returns substring of the input string between the specified character indexes (where the first character is at index 0).	<ul style="list-style-type: none"> ▶ The input string. ▶ The 0-based start index specifying where the substring should start. ▶ <i>(optional)</i> The 0-based end index specifying where the substring should end.
to_lowercase	Converts a string to lower case.	<ul style="list-style-type: none"> ▶ The input string.
to_uppercase	Converts a string to upper case (capital letters).	<ul style="list-style-type: none"> ▶ The input string.
trim	Returns a trimmed String, or null if a null object was passed in .	<ul style="list-style-type: none"> ▶ The input string.

Collection Functions




Function	Description	Parameter(s)
count	Returns the number of elements in a collection.	<ul style="list-style-type: none"> ▶ A collection or array.

Identity Functions


Function	Description	Parameter(s)
generate_uuid	Generates a (version 4) random UUID string in lower-case 8-4-4-4-12 hexadecimal format (e.g. 125d4167-c85b-43d4-b416-523625244020).	None

Function	Description	Parameter(s)
read_ sequence	Returns the next value in the sequence (as an integer).	<ul style="list-style-type: none"> ▶ The database to read the sequence from, specified by a JNDI location (as a string). ▶ The sequence name (as a string). ▶ <i>(optional)</i> The schema name, if different from the default (as a string).
rule_ name	<p>Returns the name (and optionally the path) of the currently running rule, or of the rule which triggered the current action if running an action map.</p> <div>  Note: Paths are returned without a type prefix, for example: <code>"/Production/Mandatory /Water Network/Pipe pressure must match the valve".</code> </div>	<ul style="list-style-type: none"> ▶ <i>(optional)</i> Set to "true" to return the full path and name of the rule. Otherwise just the name will be returned.


Lookup Functions




Function	Description	Parameter(s)
metadata_store_lookup	<p>Looks up the value of a key in a specific table within in a metadata store.</p> <p>The query will match all keys in the table that <i>equals</i> the input value. If there are multiple matches, the first value found in the table is returned.</p> <div>  Note: The returned value is the same as a Constant Value using the same metadata store, table and key. </div> <div>  Note: This built-in function is the reverse of metadata_store_reverse_lookup. </div>	<ul style="list-style-type: none"> ▶ The full path to the metadata store (without the DATASTORE:// prefix) e.g. TEST/CONSTANT_STORE. ▶ The name of the table in the metadata store. ▶ The key string. ▶ <i>(optional)</i> The default value to return if the query does not find any result.
metadata_store_reverse_lookup	<p>Look up the key of a value in a specific table within in a metadata store.</p> <p>The query will match all values in the table that <i>equals</i> the input value. If there are multiple matches, the first key found in the table is returned.</p> <div>  Note: This built-in function is the reverse of metadata_store_lookup. </div>	<ul style="list-style-type: none"> ▶ The full path to the metadata store (without the DATASTORE:// prefix) e.g. TEST/CONSTANT_STORE. ▶ The name of the table in the metadata store. ▶ The value string. ▶ <i>(optional)</i> The default key to return if the query does not find any result.

Shifting Functions


Function	Description	Parameter(s)
shift_geometry	<p>A shifted version of the geometry (see "Positional Data Shifting" on page 149).</p> <p>This will be a geometry of the same type as the input geometry with the same number of vertices.</p> <div>  Note: Ensure that the name used for the register_shift_vector built-in operation is used here. </div>	<ul style="list-style-type: none"> ▶ A name to identify a set of shift vectors. ▶ The geometry to be shifted, can be of any type


Sorting Functions

 **Note:** The **tsort_*** functions are used to implement iterating through objects in dependency order. Please contact 1Spatial Support for further guidance on their use.

Function	Parameter(s)
tsort_blocked_objects	<ul style="list-style-type: none"> ▶ The name of the topological sort. If not provided, or null, the default topological sort will be used.
tsort_blocked_predecessors	<ul style="list-style-type: none"> ▶ The successor. ▶ The name of the topological sort. <div>  Note: If not provided, or null, the default topological sort will be used. </div>
tsort_blocked_successors	<ul style="list-style-type: none"> ▶ The blocked predecessor. ▶ The name of the topological sort. <div>  Note: If not provided, or null, the default topological sort will be used. </div>
tsort_ordered_objects	<ul style="list-style-type: none"> ▶ The name of the topological sort. <div>  Note: If not provided, or null, the default topological sort will be used. </div>

Timestamp Functions

Function	Description	Parameter(s)
add_date	<p>Adds a date and time in the TIMESTAMP date format <i>yyyy-MM-dd HH:mm:ss.SSS</i>.</p> <div>  Note: Some data stores may not hold the same precision levels for time data as the TIMESTAMP. For example, if the TIMESTAMP value is exported to a data store with less accurate time, the TIMESTAMP value in will reflect that in the target data store. </div>	<ul style="list-style-type: none"> ▶ The Timestamp to increment. The Timestamp is obtained from the following values: <ul style="list-style-type: none"> ▶ static value ▶ dynamic value ▶ <code>get_current_date</code> function ▶ The number of milliseconds to add to the Timestamp.
current_datetime	Returns the current date and time as a string.	<ul style="list-style-type: none"> ▶ <i>(optional)</i> A string describing the date format.
get_current_date	Returns the current date and time in the TIMESTAMP date format <i>yyyy-MM-dd HH:mm:ss.SSS</i> .	None

Function	Description	Parameter(s)
to_timestamp	<p>Converts a date/time formatted string, or time in milliseconds, into a timestamp datatype.</p> <p>The default date/time format is "yyyy-MM-dd hh:mm:ss.SSS z". The default format allows just the date with no time to be specified. For example 2017-03-17 14:25:03 GMT or 2017-03-17.</p> <p>The time specified in milliseconds is from January 1, 1970, 00:00:00 GMT.</p> <div>  Note: For more information about the format rules, refer to the Oracle documentation. </div>	<ul style="list-style-type: none"> ▶ The date/time formatted (String), or time in milliseconds (Long). ▶ <i>(optional)</i> Format of the date/time string. If not specified the default format is used.



Topology Functions

Function	Description	Parameter(s)
is_structured	<p>Tests if an object is topologically structured.</p> <p>Returns true if the object is structured and false otherwise.</p>	<ul style="list-style-type: none"> ▶ The object to be tested

Built-in Operations


The following Built-in operations can be used within 1Integrate for ArcGIS.





Geometric Operations

Operation	Description	Parameters(s)
create_polygon	<p>Returns a geometry.</p> <p>Polygons are created in the specified feature class from given ring geometries. The function detects inner and outer rings and attaches them to the appropriate polygons.</p> <p>Any open lines passed to the operation are closed, where possible, before creating polygons.</p> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ The feature class ▶ A ring geometry ▶ <i>(optional)</i> A Boolean flag indicating whether to close any open lines (if possible) passed to the operation. <div>  Note: Defaults to true. If this parameter is false, it indicates that open lines should be ignored rather than closed. </div>

Network Graph Connectivity Operations





Network Graph Connectivity Operations are used to construct network graphs (see "Connecting Network Graphs" on page 146).



Operation	Description	Parameters(s)
add_position	<p>Adds an object, as a position, to a network graph.</p> <div>  Note: There will be no output results if the object is added to a network graph that already contains the object. </div>	<ul style="list-style-type: none"> ▶ The name of the network graph (can be null). ▶ The object to add.

Operation	Description	Parameters(s)
connect_positions	<p>Creates directed connections between two objects in a network graph.</p> <p>Both objects must have been previously added to the network graph.</p> <div>  Note: This connection is <i>one-way</i>. To specify a connection in both directions, call this again, reversing the order. </div> <div>  Note: There will be no output results if the connecting positions are already connected. </div>	<ul style="list-style-type: none"> ▶ The name of the network graph (can be null). ▶ The object at the start of the connection. ▶ The object at the end of the connection.
disconnect_positions	<p>Removes a direct connection from a network graph.</p> <div>  Note: There will be no output results if non-connected positions are disconnected. </div>	<ul style="list-style-type: none"> ▶ The name of the network graph (can be null). ▶ The object at the start of the connection. ▶ The object at the end of the connection.
remove_position	<p>Removes an object from a network graph. Disconnects it from any other objects in the network.</p> <div>  Note: Does nothing if the network graph did not contain the object. </div>	<ul style="list-style-type: none"> ▶ The name of the network graph (can be null). ▶ The object to remove.


Shifting Operations

Shift operations are used to perform Positional Data Shifting (see "Positional Data Shifting" on page 149).

Operation	Description	Parameters(s)
register_constraining_geometry	<p>Used to register constraining geometries (see "Positional Data Shifting" on page 149).</p> <div>  Note: Ensure that the name used for the register_shift_vector built-in operation is used here. </div> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ A name to identify a set of shift vectors. ▶ A line or area geometry with the original values for the point co-ordinates. The line segments or the edges of an area will be constrained.
register_shift_geometry	<p>Used to register shift geometries (see "Positional Data Shifting" on page 149).</p> <div>  Note: Ensure that the same name is passed to the shift_geometry built-in function. </div> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ A name to identify a set of shift vectors. ▶ A geometry with the original values for the point co-ordinates. ▶ A geometry, similar to the original geometry, but with the new values for the point co-ordinates.

Operation	Description	Parameters(s)
register_shift_vector	<p>Used to register shift vectors (see "Positional Data Shifting" on page 149).</p> <div>  Note: Ensure that the same name is passed to the shift_geometry built-in function in order to shift geometries using this set. Registering shift vectors with different shift vector set names will create independent shift vector sets. </div> <div>  Note: This function does not currently fully support 3D or measured geometries. Any 3D or measured geometries will be projected down to 2 dimensions. </div>	<ul style="list-style-type: none"> ▶ A name to identify a set of shift vectors. ▶ A two-vertex line or point geometry indicating the shift, where a point indicates no shifting



Sorting Operations

 **Note:** The **tsort_*** operations are used to implement iterating through objects in dependency order. Please contact 1Spatial Support for further guidance on their use.

Operation	Parameters(s)
tsort_add_dependency	<ul style="list-style-type: none"> ▶ The predecessor object that must appear before the successor. ▶ The successor object that must appear after the predecessor. ▶ The name of the topological sort. If not provided, or null, the default topological sort will be used.
tsort_add_object	<ul style="list-style-type: none"> ▶ The object to add. ▶ The name of the topological sort. If not provided, or null, the default topological sort will be used.

Operation	Parameters(s)
tsort_remove_dependency	<ul style="list-style-type: none"> ▶ The predecessor object that no longer must (but still may) appear before the successor. ▶ The successor object that no longer must (but still may) appear after the predecessor. ▶ The name of the topological sort. If not provided, or null, the default topological sort will be used.
tsort_remove_object	<ul style="list-style-type: none"> ▶ The object to remove ▶ The name of the topological sort. If not provided, or null, the default topological sort will be used.
tsort_reset	<ul style="list-style-type: none"> ▶ The name of the topological sort. If not provided, or null, the default topological sort will be used.

Topology Operations

Operation	Description	Parameters (s)
set_topology_edge_drag_mode	<p>Controls the reshaping of connected edges when a node is moved.</p> <p>Also applies when moving an end of an edge when changing an edge's geometry.</p> <div>  Note: This only takes effect when directly manipulating topology. </div> <p>When set to <i>true</i> (the default), edges connected to a moved node will be fixed at their other end. All other vertices will be scaled and rotated about the fixed point. This is equivalent to using the drag_vertex built-in function to move the end vertex.</p> <p>When set to <i>false</i>, only the end vertex of the edge is moved; all other vertices are fixed. This is equivalent to using the move_vertex built-in function to move the end vertex.</p> <div>  Note: This setting does not apply to edges that form a closed loop. For closed loops, only the vertex at the shared node will be moved. </div>	<ul style="list-style-type: none"> ▶ A boolean value (<i>True</i> or <i>false</i>), to set the mode on or off.

Auto-Actions

An auto-action is an action that can be run without having a rule attached to it.

Any action or quick action can be made an auto-action by use of the toggle in the Rule Author toolbar.



Figure 4-22: *Auto-Action toggle*

Auto-actions are typically used to apply processing actions in bulk on data. They are also used in conjunction with Issue Management, and for 1Integrate for ArcGIS Mobile users who want to correct data as it is being edited in the field.

Once an auto-action has been published, it will appear in the Enhancement tab of the 1Integrate for ArcGIS application Add-in or web app widget.

5 Validating and Enhancing Data

Validation and enhancement of data is performed using the 1Spatial Add-ins and widgets.

Each has a slightly different interface, but the basic functionality is the same.

- ▶ For **validation** rules, 1Integrate for ArcGIS will count the number of objects processed, and the number of features that failed your rule ("non-conformances"). These non-conformances are identified with pin icons within your dataset. A traffic light next to each rule will also indicate where large numbers of features failed (red) or where most features passed (green).
- ▶ For **enhancement** rules, 1Integrate for ArcGIS will count the number of objects processed, and any errors encountered. Any rules that encountered errors are indicated by a red traffic light. The corrections are automatically committed to your dataset.

Before proceeding, ensure that your Add-in or widget has been installed, your dataset is properly configured and your rulesets have been published. Then follow the steps according to the Add-in or widget you are using.

Using the Web AppBuilder Widget

The following steps outline how to use 1Integrate for ArcGIS to run validation and enhancement rules within the Web AppBuilder for ArcGIS.



Note: For general information on using the Web AppBuilder, please refer to the ArcGIS documentation.

Launch the Web AppBuilder widget:

1. Open a web browser.
2. Navigate to the URL of your dataset (e.g. [machine]:[port]/dataset)
3. Open the 1Spatial Widget by clicking on the widget logo.



Figure 5-1: *Widget logo*

Running Rules

Rules can be run across a specific area of your dataset, or the visible extent.



Note: When running rules in the Web AppBuilder Widget, the default extent to be processed is limited to the data visible within your data frame. If you want to process the entire dataset, make sure you are zoomed out so that the full extent is visible.

Run rules over the visible extent:

1. Select one or more rules by clicking on their checkboxes.

Packages (groups) of rules can be selected using the group checkbox.

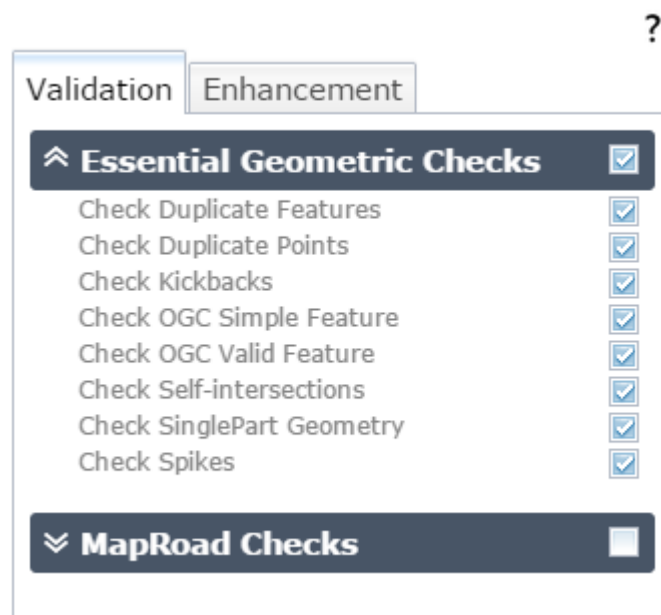


Figure 5-2: *Selecting rules to run*

2. Click **Process Extent** to run the selected rules on the visible extent.



Note: If at any time you wish to cancel the current process, press the **Stop** button.

A progress bar will display whilst the rules are being run.

Run rules over a selected region:

1. Select one or more rules by clicking on their checkboxes.

Packages (groups) of rules can be selected using the group checkbox.

2. Use the **Draw** tool to define an area of your data to be processed (an "extent").

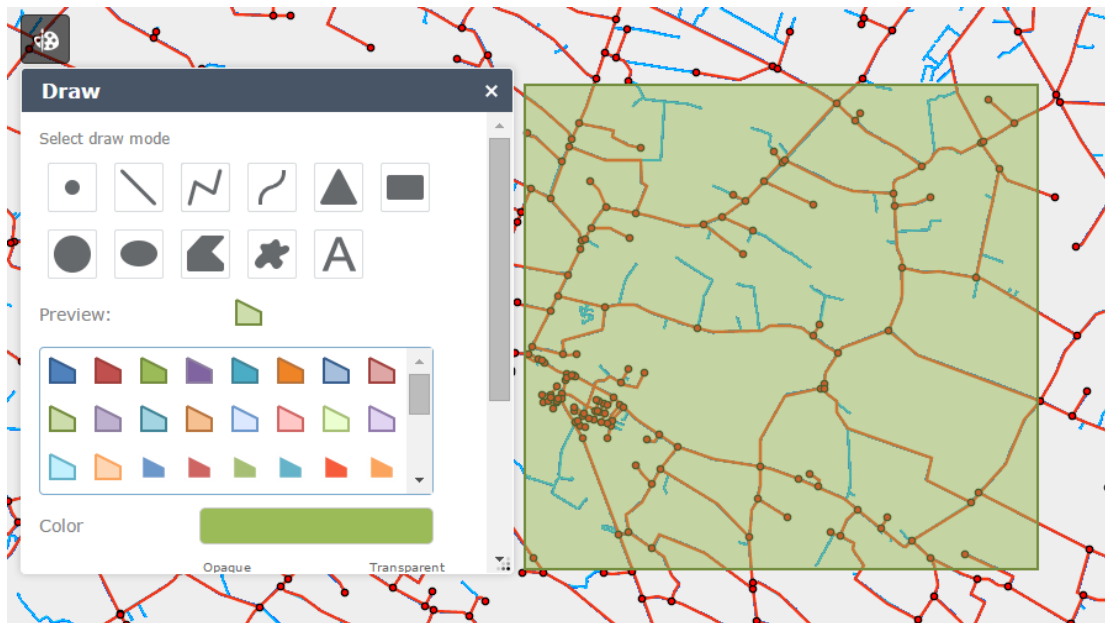


Figure 5-3: Using the Draw tool to define an extent



Note: If multiple areas are drawn, only the most *recently* drawn area will be processed.

3. Click **Process Extent** to run the selected rules across the defined area.



Note: If at any time you wish to cancel the current process, press the **Stop** button.

A progress bar will display whilst the rules are being run.

Viewing Validation Results

Once your rules have finished running, the progress bar will display "COMPLETED" and the results will be displayed.

Beneath this progress bar is a count of the number of objects processed, and the number of features that failed the rules ("non-conformances").

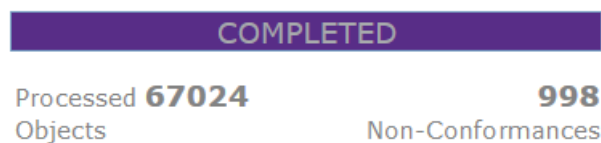
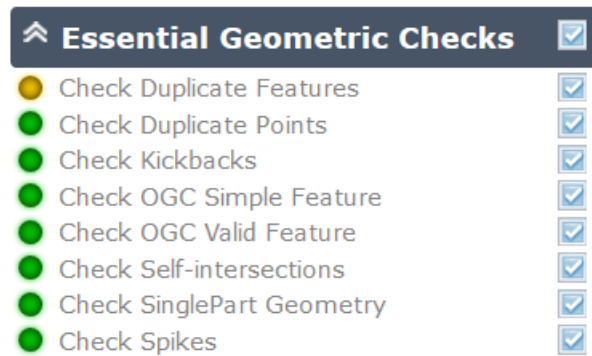
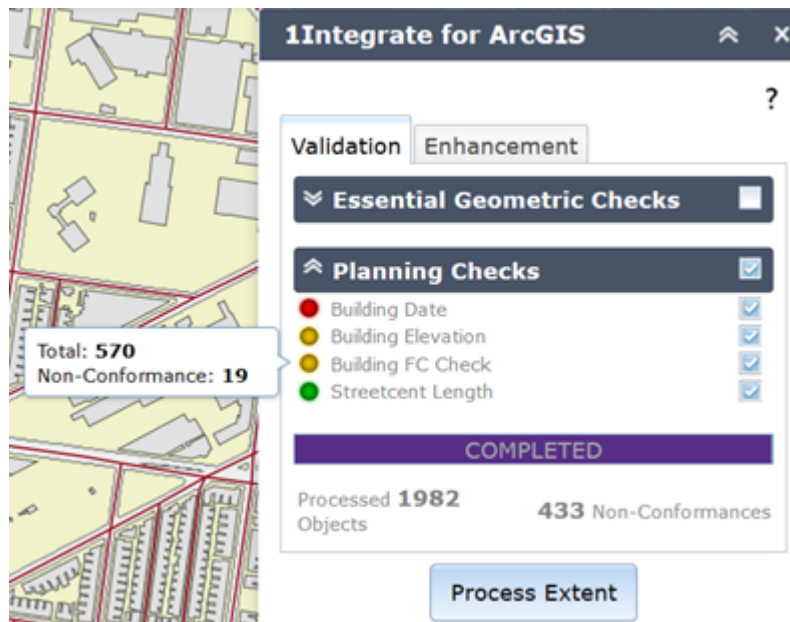


Figure 5-4: Validation results

A traffic light next to each rule will also indicate where large numbers of features failed (red) or where most features passed (green).

Figure 5-5: *Traffic light rating*

By hovering the mouse over each traffic light icon, a summary for than rule can be displayed.

Figure 5-6: *Traffic light summary for a rule*

Layers of Pins

All non-conformances are marked in your dataset with a pin.

Clicking on a pin will display details of the non-conformance, such as the associated feature IDs and the name of the rule.

As the pins are displayed as layers within the application, they can be turned on or off just like any other layer.

Within Web AppBuilder, open the layer list.

Figure 5-7: *Layer List icon*

The layer list displays which pin type represents each rule. These individual layers can then be turned on or off.

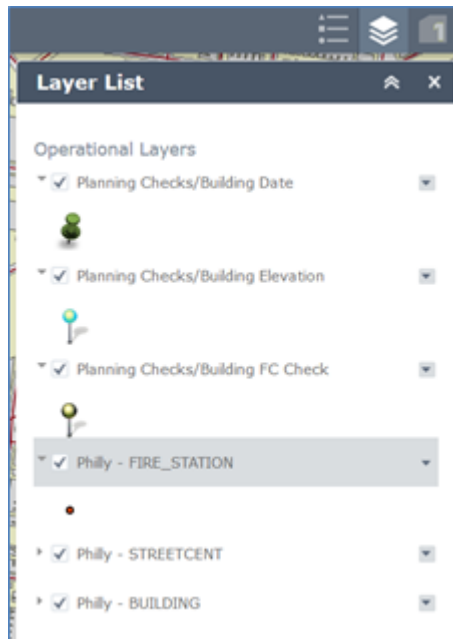


Figure 5-8: *Web AppBuilder Layer List*

Using the ArcMap Add-in

The following steps outline how to use 1Integrate for ArcGIS to run validation and enhancement rules within ArcMap.



Note: For general information on using ArcMap, please refer to the ArcGIS documentation.

Launch the ArcMap Add-in:

1. Open the ArcMap application.
2. Open your dataset.
3. Open the 1Spatial Add-in via the menu button.

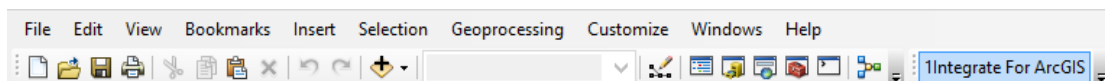


Figure 5-9: *ArcMap menu*

4. Click **Fetch Rules** to display the rulesets that have been published for your dataset.

Running Rules

Validation and enhancement rules are run in the same way within ArcMap.



Note: When running rules in ArcMap, the extent to be processed is limited to the data visible within your data frame. If you want to process the entire dataset, make sure you are zoomed out so that the full extent is visible.

Run rules over the visible extent:

1. Click **Fetch Rules** to display the rulesets that have been published for the current dataset.
2. Select one or more rules by clicking on their checkboxes.
Packages (groups) of rules can be selected using the group checkbox.
3. Click **Run** to run the selected rules on the visible extent.

A progress bar will display whilst the rules are being run.

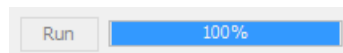


Figure 5-10: *ArcMap rules progress bar*

Viewing Validation Results

Once your rules have finished running, the progress bar will display 100% and the results will be displayed.

Displayed next to each rule is a count of the number of objects processed, and the number of features that failed the rule ("non-conformances").

A traffic light next to each rule will also indicate where large numbers of features failed (red) or where most features passed (green).

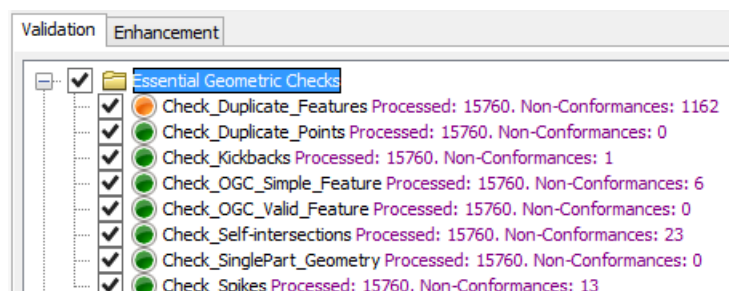


Figure 5-11: *Validation results*

All non-conformances are marked in your dataset with a pin.

The non-conformances for each rule are grouped with a similar pin style. These groups are displayed as **Layers** within the **Table of Contents** window, and can be hidden or displayed as required.

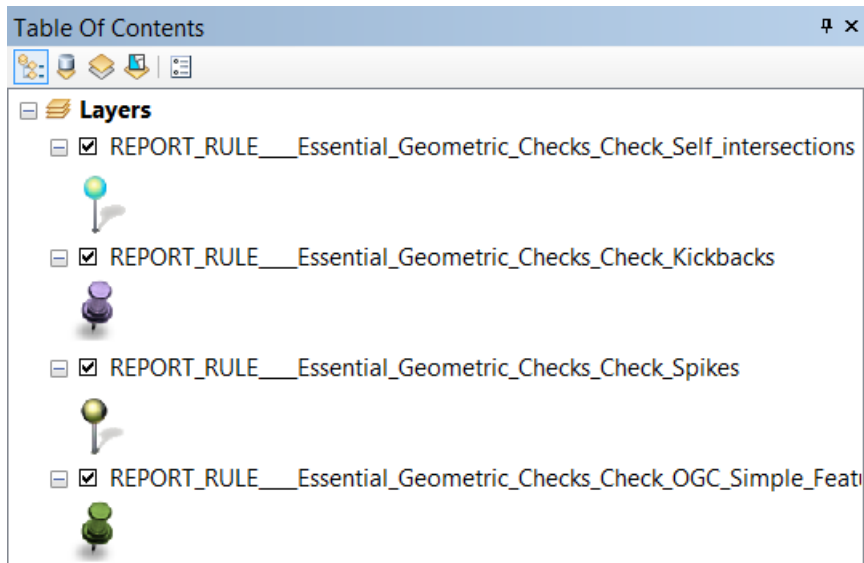


Figure 5-12: *Validation layers*

6 Extensions

Extensions, sometimes referred to as "built-ins" can be added to 1Integrate for ArcGIS to extend existing functionality.

They take the form of **.JSON** files, and can be uploaded in a **.zip** file via the Rule Author.



Note: Typically, these extensions will be provided directly by 1Spatial. Please contact us for more information.

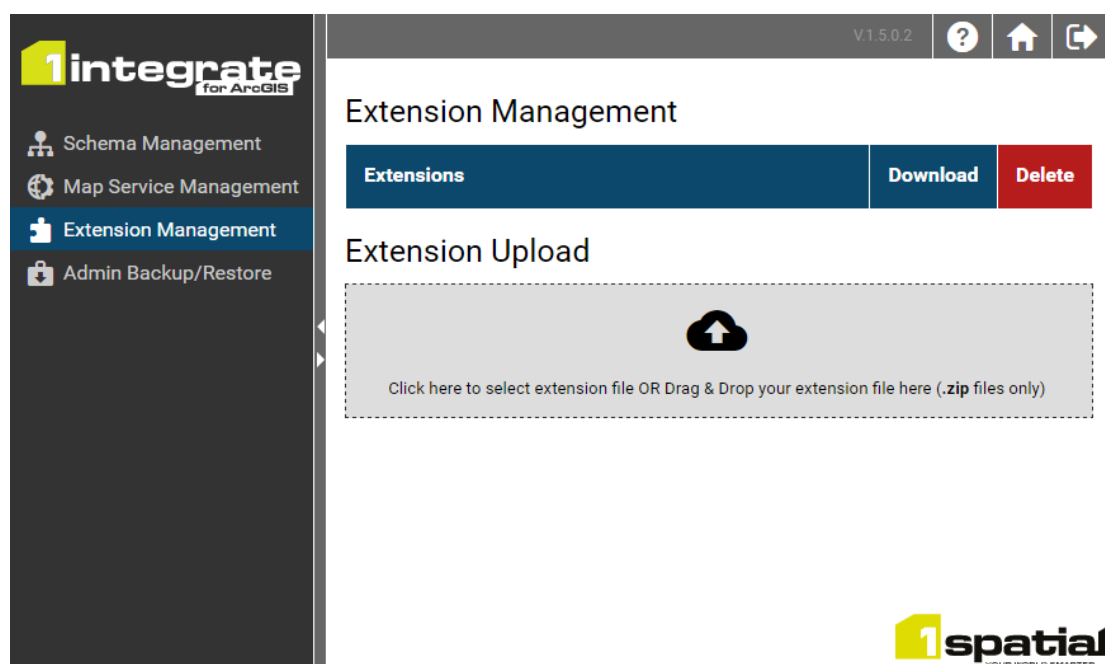


Figure 6-1: *Extension Management page*

Adding Extensions

Once you have your extension file, carry out the following steps to add your extension:

1. Log in to the Rule Author.
2. Navigate to **Settings > Extension Management**.
3. Either drag and drop a **.zip** file containing your extension onto the **Extension Upload** box, or click on the box and navigate to your file.

Your extension is now listed within the **Extensions** table at the top of the page.

Network Graphs

Using 1Integrate for ArcGIS, you can analyse the global graph connectivity between objects such as roads, railways, pipes, or cables to ensure you can navigate between any two objects within a network of these features, in order to confirm the whole network is properly connected.

1Integrate stores network graph definitions in a class defined in the system schema, which is separate from the import and export schemas.



Note: Once network graphs have been constructed via an action (see "Connecting Network Graphs" on the facing page) they can be tested using rules (see "Validating Network Graphs" on page 149).

Ideally, all the objects should be connected into one network graph; however, objects fail the validation check if they are contained in an area which is isolated from the rest of the network or has restricted access.



Figure 6-2: *Isolated island (no connection to network)*



Figure 6-3: *Isolated island (restricted access to network)*

Connecting Network Graphs

Network graph connectivity is defined by setting up an action using built-in operations (see "Network Graph Connectivity Operations" on page 131).

A network graph is defined in two parts, which must be performed in order using a sequence:

1. Define the objects within the network graph (using the **add_position** and **remove_position** built-in operations)
2. Set up the connections within the network graph (using the **connect_positions** and **disconnect_positions** built-in operations)



Note: You can add and connect features which already exist in the network graph, but an error occurs if you try to connect or disconnect features which have not yet been added. If you remove a feature from the network graph, you do not need to disconnect it first.

Spatial relationships are used to define how the points and lines interact (e.g. touch, overlap, or cross). In a network graph constructed from point and line features, the points become nodes and the lines are the connections between the nodes.

You can construct multiple network graphs in 1Integrate. Although rules are run against all network graphs, each is reported separately.

Actions can be created in many ways to achieve the same network graph connectivity. Labels and multiple actions can be used to connect a network graph in stages.

Line Networks

In the example below, the points of contact become nodes in the network graph, with road sections as the connections.

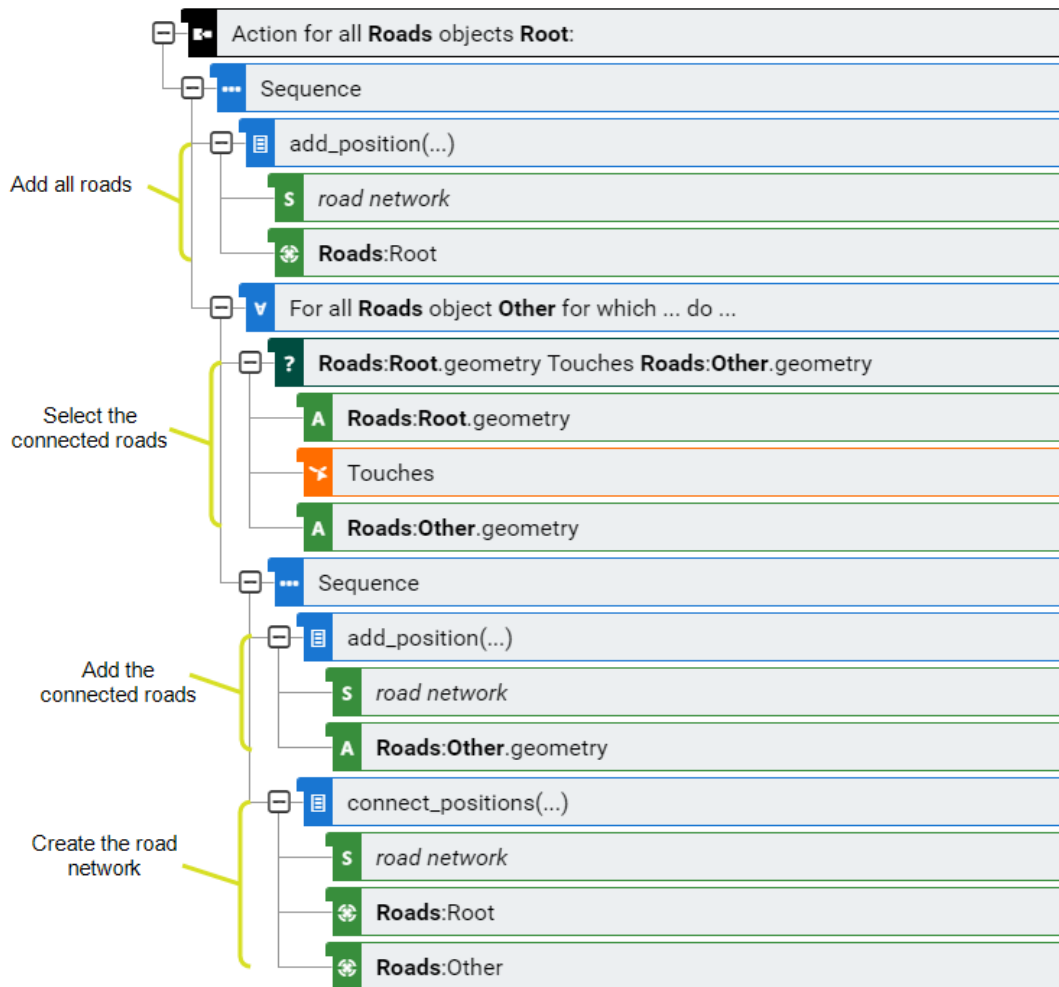


Figure 6-4: Example action defining a simple road network

Line and Point Networks

The following example shows how a network graph of valves (points) and pipes (lines) might be defined.

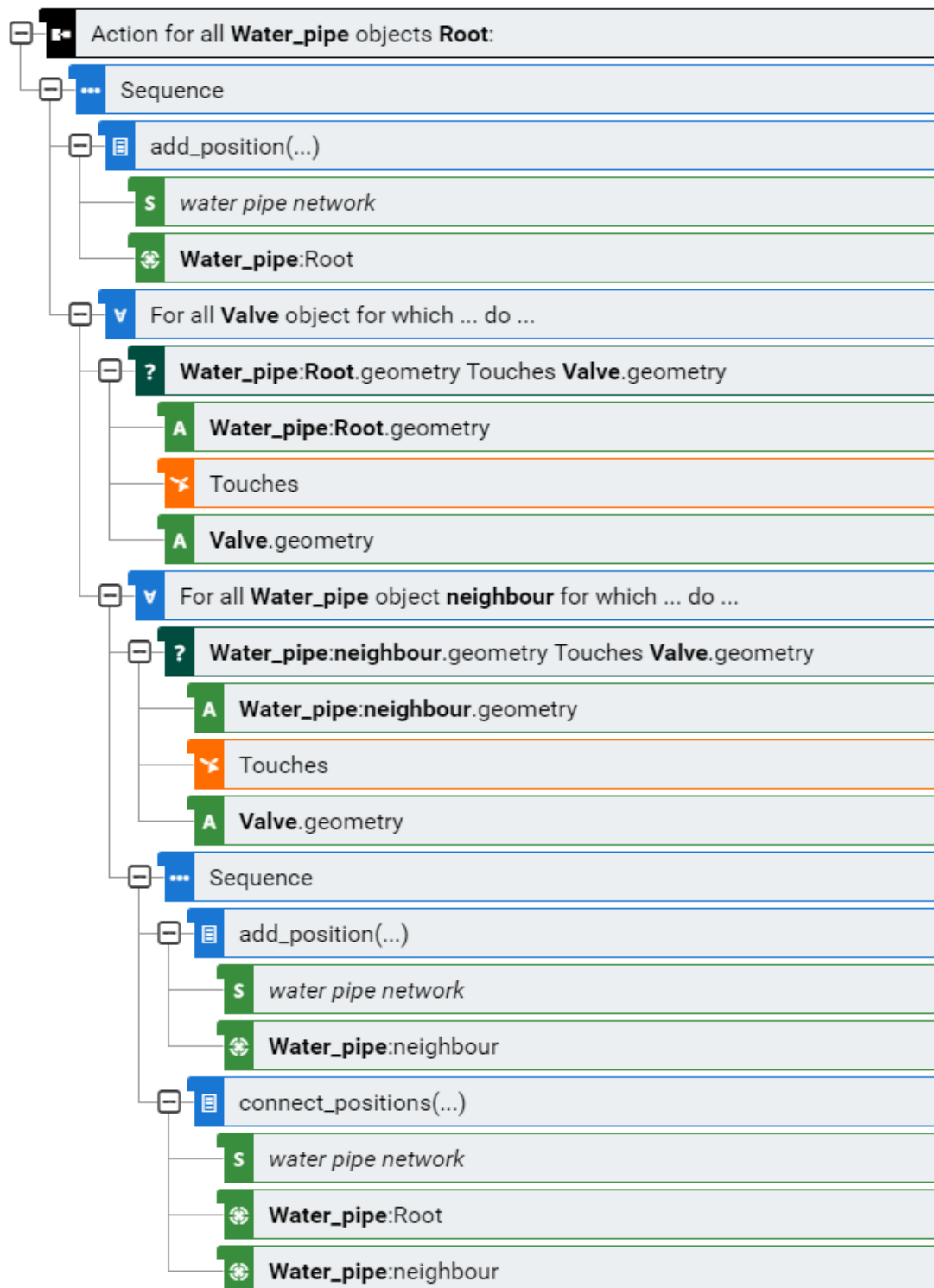


Figure 6-5: Example action defining a network of valves and pipes

Validating Network Graphs



Note: Before you can validate network graphs, a network graph must first be constructed (see "Connecting Network Graphs" on page 146).

Network graph connection can be validated using a rule to check Network Attributes.

Network Attributes:

Network Attribute	Description
max_reported_islands	Threshold for the number of isolated islands reported. If not set, default value of 1000 is used.
max_reported_positions	Threshold for the number of features reported in an island. If not set, default value of 1000 is used.
name	Name of the network graph (used when constructing the network graph using the name parameter of the add_position , remove_position , connect_positions and disconnect_positions built-in operations).
num_islands	Number of islands found in the network graph. A coherent network has one island which is equivalent to the whole network graph.
report_cutoff	Threshold for creating a report. If the number of nonconforming objects exceeds the report cutoff, no report is produced. If not set, default value of 1000 is used.



Note: In network graphs with more than one island, the main island is considered to be the one which contains 50% or more of the objects queried.

Positional Data Shifting

Shifting data is required when positional accuracy of some data is known to be poor, or at least not good enough for its intended use.

Techniques used to capture spatial data have improved over the years, especially in terms of the positional accuracy. This causes problems both for data suppliers who need to update their existing data in order to use more accurate capture techniques and also for data consumers if their own data becomes misaligned when the base data positioning is improved. For example, utility pipes that run within a road surface polygon within the base data may no longer run along inside the road after the data has been shifted.

Similar issues occur when performing data conflation (i.e. bringing together data from multiple sources). If the positional accuracy of the data is not the same, a straight integration can result in erroneous relative positions between features.

In order to resolve the inconsistencies caused by variations in positional accuracy, the best solution is to improve the low accuracy data to match or at least sufficiently approach the accuracy of the high accuracy data.

Shifting data to improve position is more complex than simply applying the same transformation to all features as the shifts required vary across the data. 1Integrate for ArcGIS uses **Shift Vectors** to optimise how data is shifted.

Using Shift Vectors

Shifting data requires a number of steps:

1. **Establish the required shifts.**

Consumers of the base data will be supplied with the shifts of the base data. Producers of the base data can calculate or generate these shift vectors for the base data.

2. **Shift the data** using the vectors.

1Integrate has built-in functions and operations to allow the shifting of data using these shift vectors (see "Shifting Functions" on page 128 and "Shifting Operations" on page 132).

3. **Validate the results** to ensure connectivity and relative positioning is maintained.

Establish the Required Shifts

The required shifts are typically collected as part of a ground truth exercise by surveyors and saved as shift vectors. Ortho photography can also be used to capture the vectors to re-position the base data.

Alternatively if there is access to more accurate geometries then custom 1Integrate actions can be written to calculate the shift between these data sets. This option is made easier when 2 corresponding features can be matched using an attribute (ID or name). Otherwise, a geometric matching can be performed, but it is more complex to put in place.

Once the shift vectors are available and *before* registering them in a shift field, they could be analysed for coverage and consistency using 1Integrate for ArcGIS. This could include:

- ▶ Applying a tolerance, to turn any shifts less than a specified tolerance into "no shift" points.
- ▶ Validating shift vectors to look for crossing or opposing vectors that would cause shearing.
- ▶ Analysing the coverage of shift vectors to find regions that are not near any vectors.

This will have the advantage of not moving any data in areas where all the observed shifts are very small, and can reduce operational impact.

Shift the Data

The process of shifting the data can be broken down into three sub-steps:

- a. Register the known shifts
- b. (*optional*) Record any geometric constraints
- c. Apply the shift vectors to the data

Register the Known Shifts

The aim of this stage is to build a shift field, based on known shifts in the data (shift vectors and points).

This field is built using multiple calls to the 1Integrate built-in operation **register_shift_vector**, which registers a single shift. This built-in operation requires two parameters:

- ▶ The **name** of the shift field.
This name must be the same for all shifts registered in the field. It will also be used later on to apply the shifts.
- ▶ A **geometry** to describe the shift.
This can be either a line with two points (a vector indicated by a start point and an end point), or a point (indicating a location where no shift has occurred)

The higher the density of shift vectors, the more accurate the shifting will be. However, the appropriate balance between data collection costs and the accuracy of the result needs to be worked out for each specific case.



Note: It is essential to record both shift and no-shift information. If no-shift locations are not recorded, then features around these locations will get moved based on the surrounding recorded shifts. The no-shift information will tend to keep nearby features close to their original position.

Record Geometric Constraints

When shifting data, any shared vertices between features will be shifted to the same location, keeping data connected.

Some data does not add vertices where features touch (e.g. branch pipes touch a mains pipe without adding a vertex to the mains pipe). In these situations there are two options:

- A. In a separate 1Integrate session: load the data, build topology and then commit the features to be shifted. This will add new vertices where the features intersect (this may not be acceptable for some data models).
- B. Identify 'master' features (e.g. the mains pipes) as constraining geometries, which attempts to keep features connected to the master features without adding new vertices.

If option B is required then this can be performed in an action run in the same session, after all the shift vectors have been registered, to register these master features as constraining geometries. This has the effect of adding shifting rules to make sure that any point on a constraining line should remain on (or very close) to that same line after shifting.



Note: Absolute connectivity cannot be guaranteed in this case, so we recommended adding an action afterwards to fix small errors.

Adding a constraining geometry to a shift field is performed using the built-in operation **register_constraining_geometry**. This built in operation requires two parameters:

- The **name** of the shift field.

This name must be the same as the one used to register shifts in the field.

- The constraining **geometry**.



Note: Adding constraining geometries to the process may take some time, so for performance reasons it is recommended to apply as few constraining geometries as possible.



Note: Constraints that cross or touch other constraints will raise an error and not be registered unless there is a common vertex at the cross/touch point.

Apply Shift Vectors

Once a shift field has been populated with shift vectors and any constraining geometries, the shift field can be used to shift other data.

This is performed using the built-in function **shift_geometry**. This built-in function requires two parameters:

- ▶ The **name** of the shift field.
This name must be the same as the one used to register shifts and any constraining geometries.
- ▶ The **geometry** to be shifted.

The result is the shifted geometry that can either be used to replace the old geometry, or to create a new feature as a copy of the original (e.g. using new target classes).

Validate the Results

Once the data has been shifted, then standard 1Integrate functionality can be used to ensure that the data has remained connected and valid.

Rules can be written to check that features are connected to the same features as before the shift, and that the relationships to the base features are the same as before.

Example: Shifting Pipes

In this example, pipes have been captured against a set of parcels from old cadastral data.

New (more accurate) cadastral data has been released, but the relative position of our pipes regarding these new cadastral parcels is often wrong.

The figure below shows the pipes as a light blue line, with point assets along these lines, and the old cadastral parcels in dark blue. The pipes were captured against these cadastral parcels, so their relative positions are considered to be accurate.



Figure 6-6: *Initial state: parcels, pipes and point assets*

The next figure below shows the same pipes against the new, more accurate cadastral data in red.



Figure 6-7: *Old pipes with new parcels*

We can see that the relative position of the pipes to these cadastral parcels cannot be correct; the pipes cross the cadastral parcels in places. We

therefore need to shift the pipes to adjust their position to the new cadastral parcels.

In this scenario the supplier of the base data has not provided shift vectors so these are first calculated from the old and new base data.

The figure below shows an action that compares the position of centre of two parcels **v1** (old parcels) **v2** (new parcels) which are matched using their **ID** attribute.

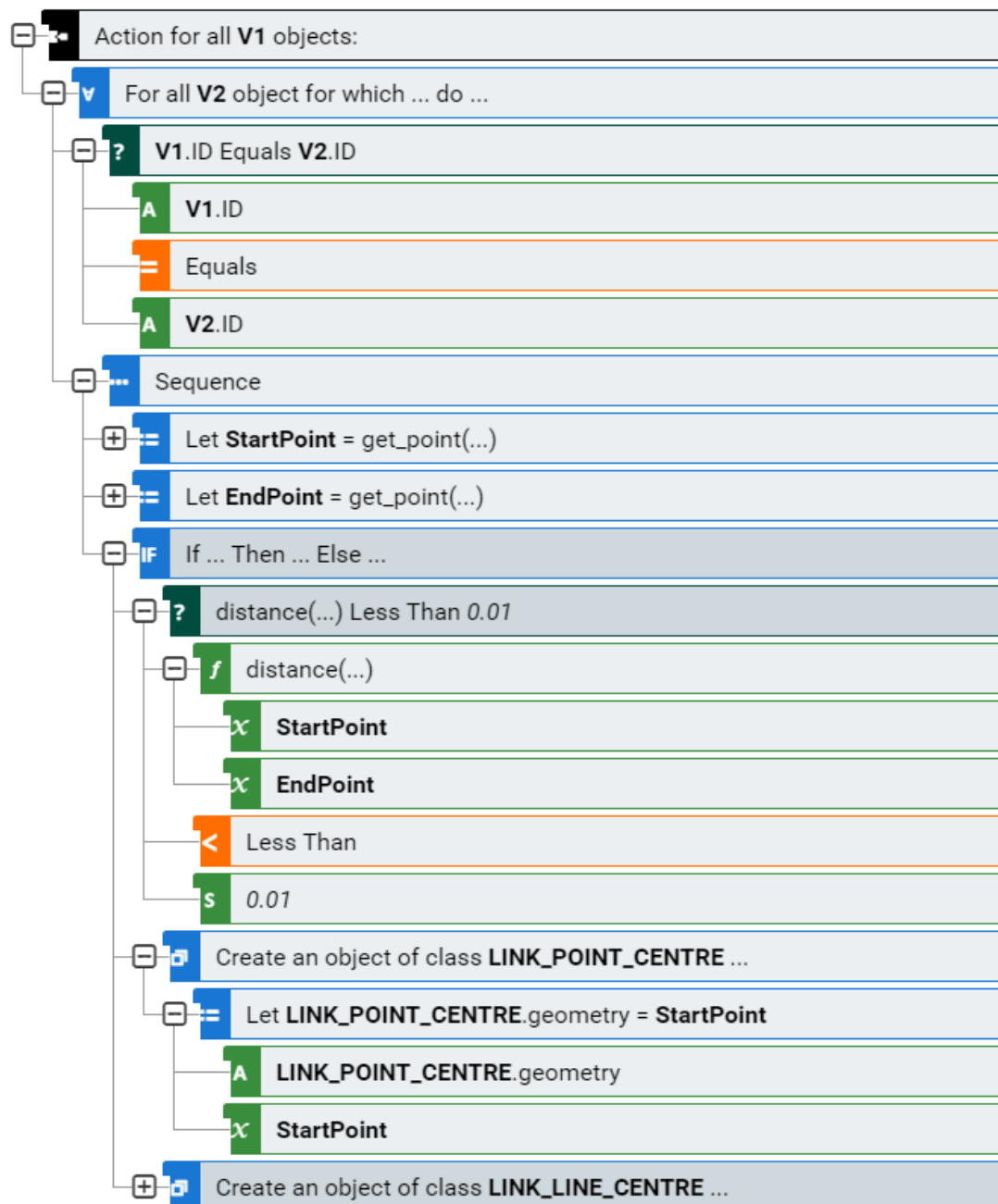


Figure 6-8: Action to generate shift vectors

If they are close enough, a **LINK_POINT_CENTER** object is created. This will be used to register a no-shift point.

Otherwise a **LINK_LINE_CENTER** object is created. This will be used to register a shift vector.

The figure below shows a set of shift vectors that have been computed between the centroids of the matching parcels using this action.

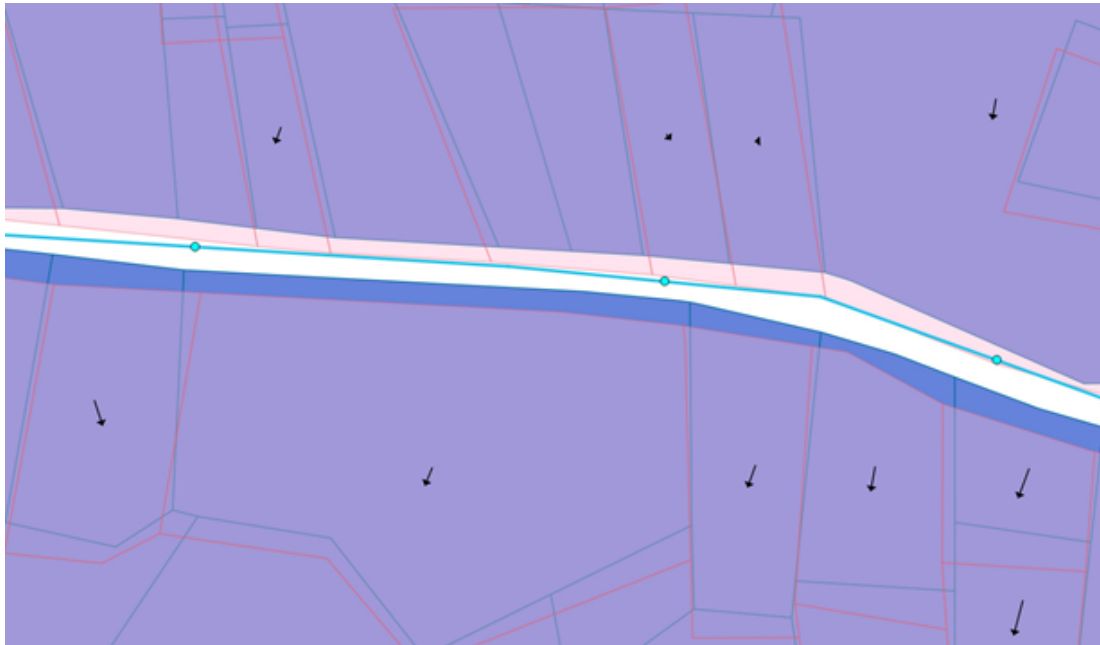


Figure 6-9: *New parcels and shift vectors*

The next step is to register our computed shifts to our shift field "Shift Network", using the built-in operation **register_shift_vector**.

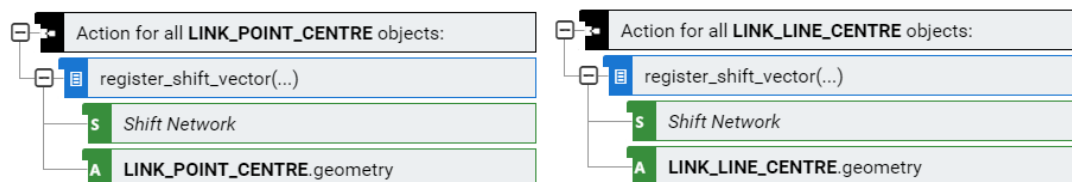


Figure 6-10: *Actions to register the no-shift points and shift vectors within the "Shift Network" shift field*

We can now apply the shift to the pipes and point assets.

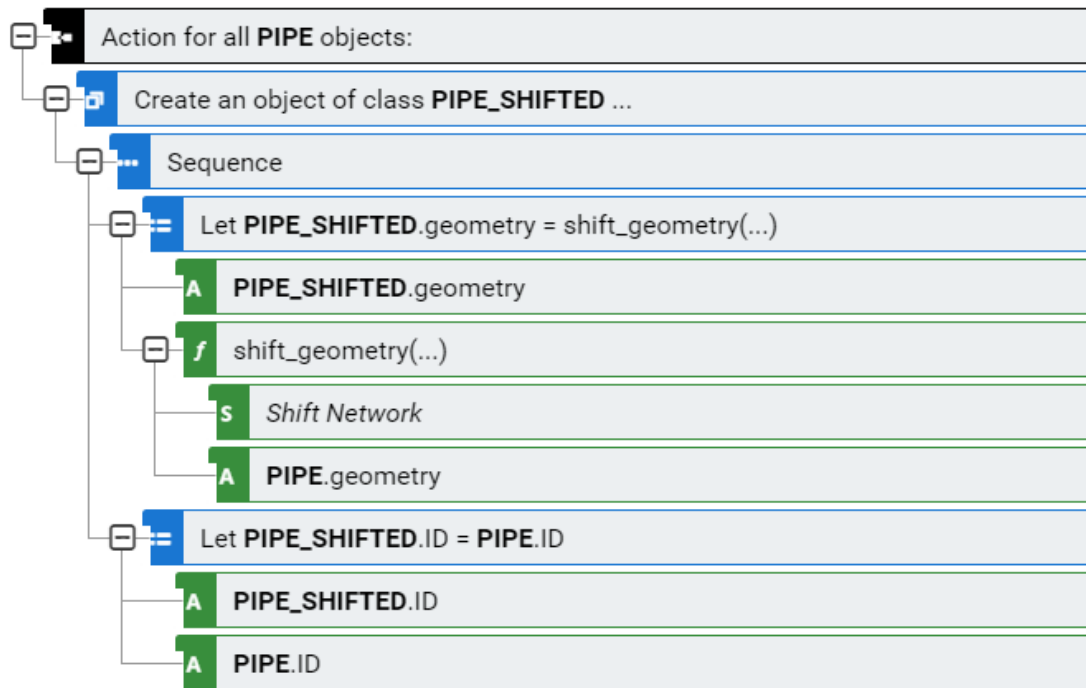


Figure 6-11: Action to shift the pipes

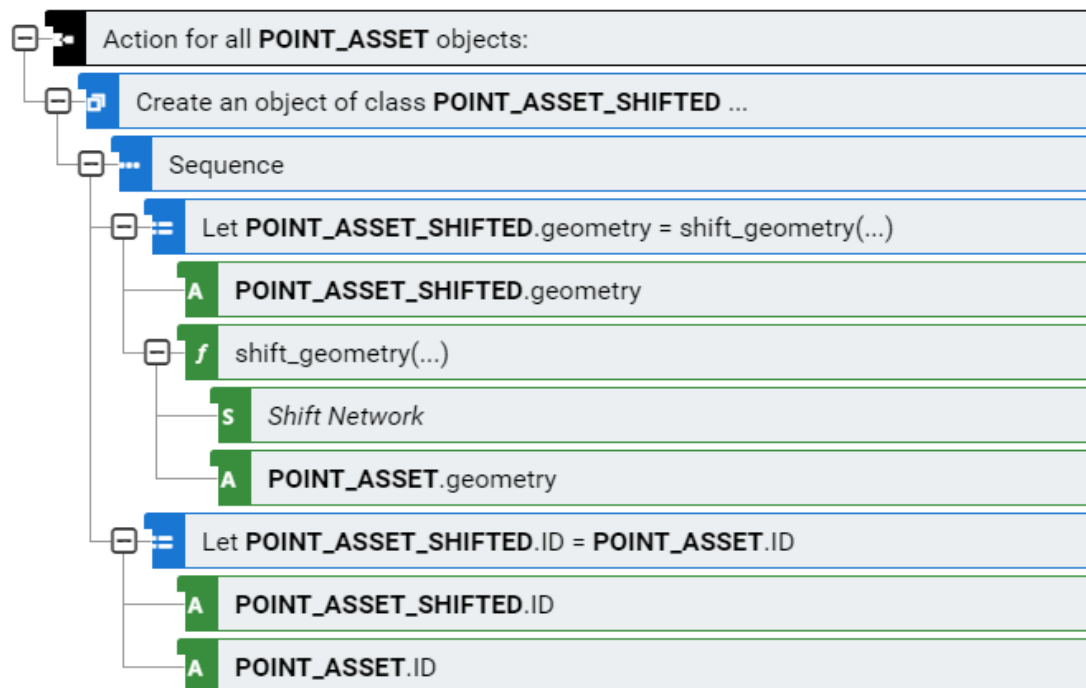


Figure 6-12: Action to shift the point assets

The figure below shows the results obtained by shifting the pipes and the point assets according to our shift field.

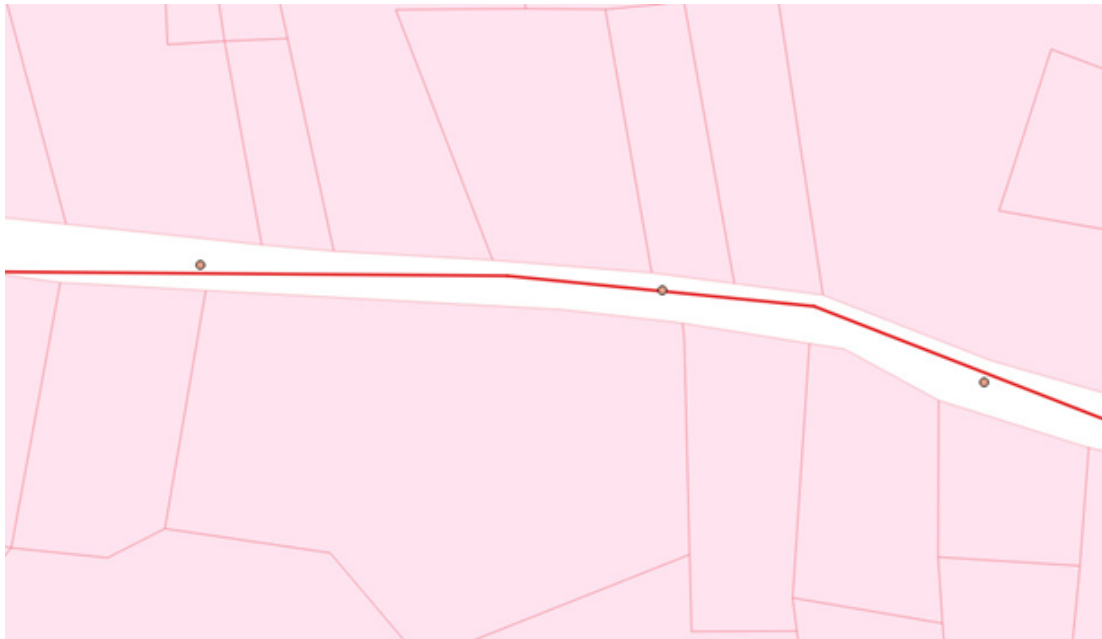


Figure 6-13: *Shifted pipes and point assets without constraint*

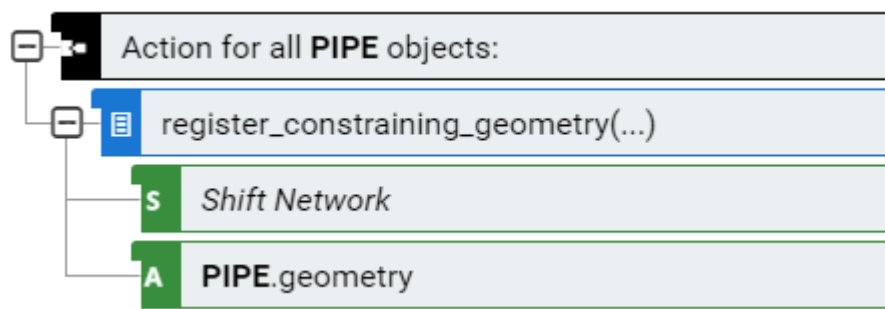
The pipe is now in an acceptable position regarding the new parcels, but we notice that the point assets which were initially positioned along the pipe are now disconnected from it.

This is because the line was moved by applying the shift vectors to its vertices. If the point assets were far away from vertices of the line, then their shifting could be dictated by other nearby shift vectors, which could be different to those affecting the two vertices on either side along the line.

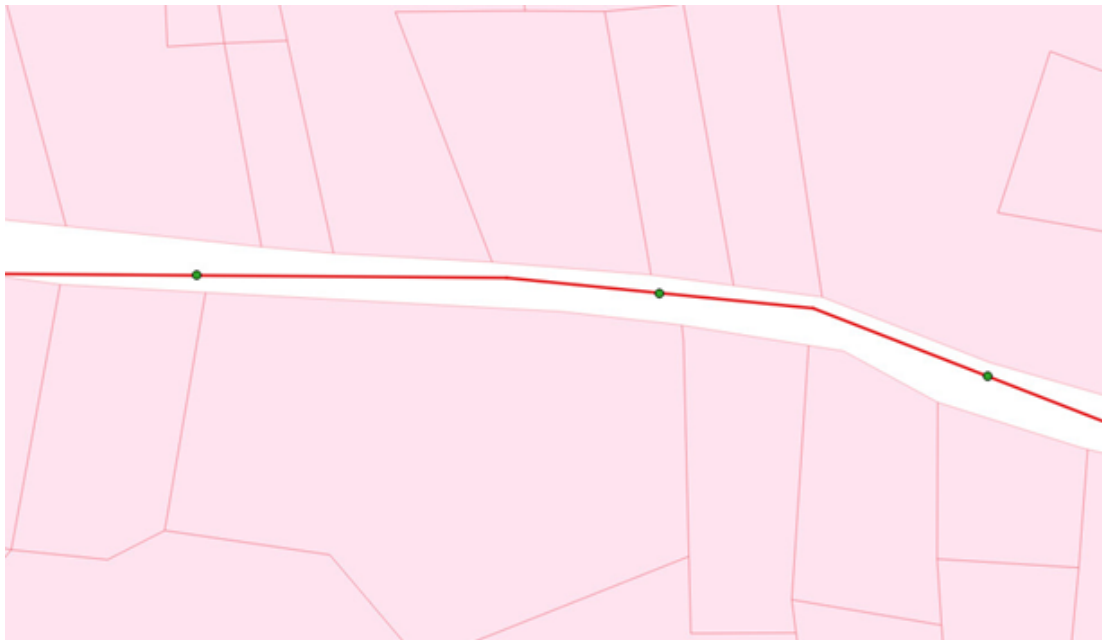
To avoid this problem, we register the pipe as a *constraining feature* (see "Record Geometric Constraints" on page 152), so that point features on the pipe remain on it after the shift.



Note: Adding constraining features must be done after having registered the shifts, and before shifting features.

Figure 6-14: *Registering constraining features*

The figure below shows the result obtained when applying the shift after having registered the pipe as a constrained feature. The shifted point assets now lay on the shifted pipe.

Figure 6-15: *Shifted pipe and point assets, using a constraining pipe*

The final figure below shows all the original data with the results superimposed, to help visualise the overall solution.

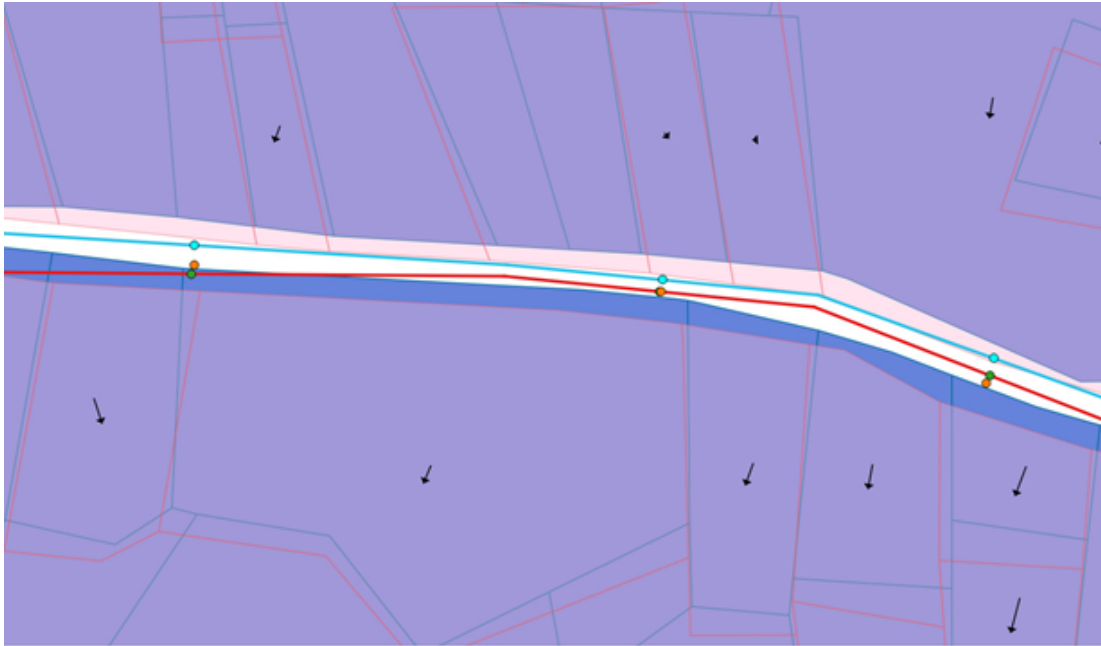


Figure 6-16: *Source data and results superimposed*

7 Issue Management

By default, when creating validation rules and reporting on validation failures, 1Integrate for ArcGIS creates pins within the user interface to pinpoint the location of a failure.

However, these pins are not permanent features. Sometimes there is a need for the location of the failures to be created as a feature layer that can be stored on a more permanent basis, or at least until the failure has been resolved.

Issue Management allows you to do just that. It manages the creation and deletion of "issue" features that are stored as geometry and attributes within a map service (in 1Integrate for ArcGIS Server Edition) or as a shape or file geodatabase layer (in 1Integrate for ArcGIS Desktop Edition).

The process of configuring Issue Management can be broken down into four steps:

1. [Configure an Issue Layer in your data](#)

A new layer is required to hold the issue features.

2. **Create a Rule**

A rule is required to identify non-conforming features.

3. [Create an Issue Management Quick Action](#)

An action is required to create the issue features and link them to the non-conforming features. It must be:

- ▶ a **Create or Delete Issues** Quick Action
- ▶ an **Auto-Action** (in order for it to appear in the Add-in or widget)

4. **Publish and Run the Action**

Run the Issue Management action to populate your layer with issue features.



Note: If the 1Integrate for ArcGIS Mobile extension is installed, then users do not run the actions through the 1Integrate for ArcGIS Add-in or widget. The Mobile extension can be configured to run rules and actions as soon as anything changes within the map service. When Issue Management is used in this way, then as soon as a user edits features, Issue Management actions are run to determine if issue features need to be created. Similarly when a user corrects a feature, Issue Management will remove issue features automatically without the user having to do anything.

Configure an Issue Layer

In order to create issue features, you must first create a new layer within your map service, shapefile or geodatabase layer using the following schema:

Column Name	ESRI Type	Description
OBJECT_ID or FID	Object ID	auto generated ID column by ESRI
SHAPE	Geometry	geometry
OBJECT_REF	Long/Text	store the ID of the real world object which failed the rule
DETAIL	Text	store the name of the rule that the real world object failed

Create an Issue Management Quick Action

Issue Management is a category of Quick Action containing the **Create or Delete Issues** template.

Quick Action

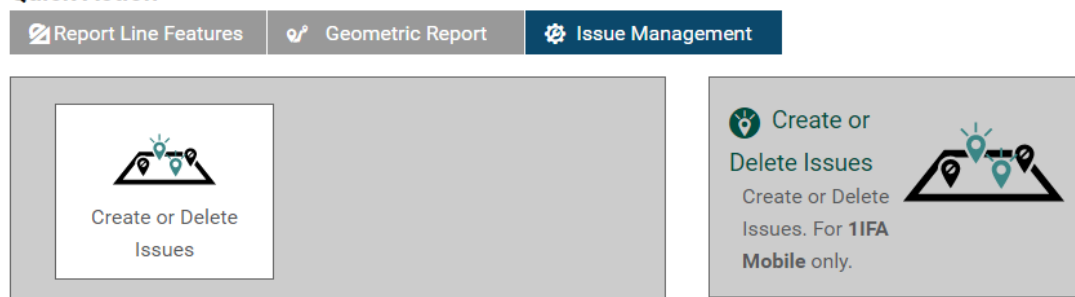


Figure 7-1: *Issue Management Quick Action*

This Quick Rule requires the following parameters:

- ▶ **ruleReference** - the rule that is to be run
- ▶ **objClass** - the feature class the rule is targeting
- ▶ **objectIdAttribute** - the attribute that will uniquely identify the features that fail the rule (typically the GlobalID, FID or ObjectID of the feature)
- ▶ **issueClass** - the name of the issue feature class you would like issue features to be created in

- ▶ **issueReferenceAttribute** - where you would like the unique identifier of the failed feature (set in **objectIdAttribute**) to be stored on the Issue feature (if using the recommend schema for the Issue layer, then this will be OBJECT_REF)
- ▶ **detailAttribute** - specifies the attribute where you would like the name of the rule to be added to the Issue feature (if using the recommended schema for the Issue layer, then this will be DETAIL)

Figure 7-2: Parameters required for the Issue Management Quick Action

In order for the Issue Management action to appear within the 1Integrate for ArcGIS Add-in or widget, it is necessary to make it an **Auto Action** (see "Auto-Actions" on page 136).



Figure 7-3: Auto-Action toggle

Worked Example: Manhole covers with an undefined cover shape

In this example we are going to define an Issue Management Quick Action that will create issue features for any manhole cover features that have an undefined cover shape.

We will make this an auto-action, so that issue features can be automatically created and deleted as their linked features are updated. This means that if a manhole cover is updated to include a defined cover shape then the corresponding issue feature will be deleted, and if a new manhole cover is

created with an undefined cover then shape a corresponding issue feature will be created.

1. In order to create issue features, you will need a new layer within your map service or a shapefile or file geodatabase layer to be created that will persist the issue features.

Create an **ISSUE** layer using the following schema:

Column Name	ESRI Type	Description
OBJECT_ID or FID	Object ID	auto generated ID column by ESRI
SHAPE	Geometry	geometry
OBJECT_REF	Long/Text	store the ID of the real world object which failed the rule
DETAIL	Text	store the name of the rule that the real world object failed



Figure 7-4: *ISSUE layer created in ArcGIS Pro*

2. Open the Rule Author.
3. First we must create a simple validation rule to identify the **MANHOLE** point features.

Create a new rule to check that an attribute called **COVER_SHAP** is not equal to **U** (U stands for "unspecified").

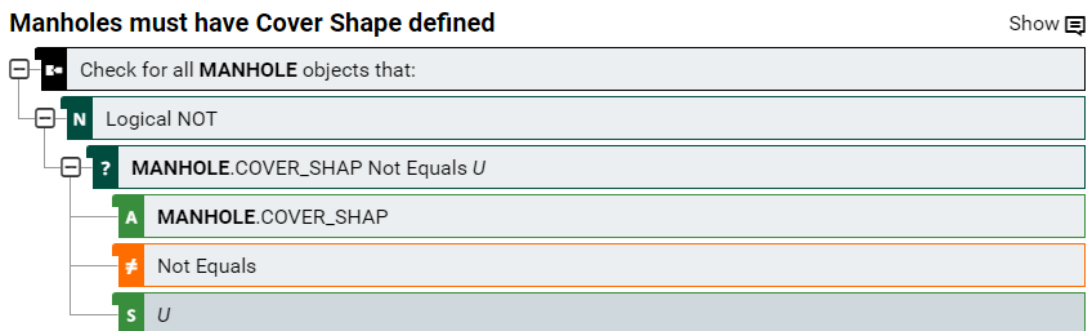


Figure 7-5: Rule to identify Manhole features with an unspecified cover shape

4. Now we must create our Issue Management Quick Action.

Create a new Quick Action and select type **Issue Management > Create or Delete Issues**.

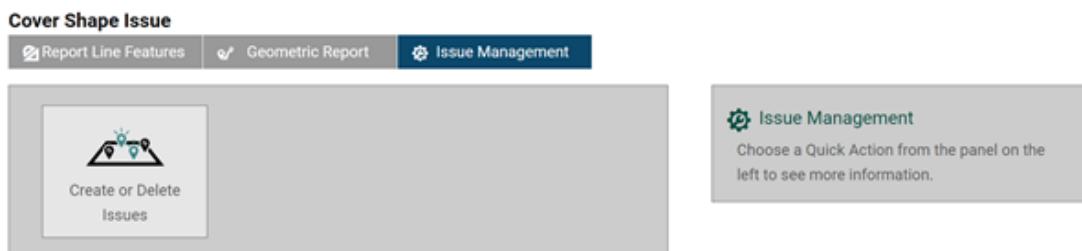


Figure 7-6: **Create or Delete Issues** Quick Action

5. The quick action now requires parameters in order to ensure there is a reference between the features that fail validation and the issue features that are created as a result.

Note: In our example dataset, **GLOBALID** is the attribute that uniquely identifies the features that fail validation. **ISSUE** is the name of the layer we created earlier to contain the issue features. **OBJECT_REF** is the attribute in this layer that uniquely identifies the issue features, and **DETAIL** is the attribute that lists the rule that the linked feature failed.

Enter the following parameters for the Quick Action:

- ▶ **ruleReference** - (select the rule created earlier)
- ▶ **objClass** - MANHOLE
- ▶ **objectIdAttribute** - GLOBALID
- ▶ **issueClass** - ISSUE

- ▶ **issueReferenceAttribute** - OBJECT_REF
- ▶ **detailAttribute** - DETAIL

Figure 7-7: Defining the MANHOLE quick action

6. In order for the Issue Management action to appear within the 1Integrate for ArcGIS Add-in or widget, it is necessary to make it an **Auto Action**. Select the Auto Action toggle from the toolbar at the top of the screen.



Figure 7-8: Auto-Action toggle

7. Save the Quick Action and publish the associated Ruleset.
8. Open the 1Integrate for ArcGIS Add-in or widget (web application, ArcMap or ArcGIS Pro).
9. Within your Esri application, give your Issue feature class a representation so that when issue features are created, they will be in the symbology / colour you will identify.

10. Open the 1Integrate for ArcGIS Add-in or widget. The Issue Management action will be found in the Enhancement tab.

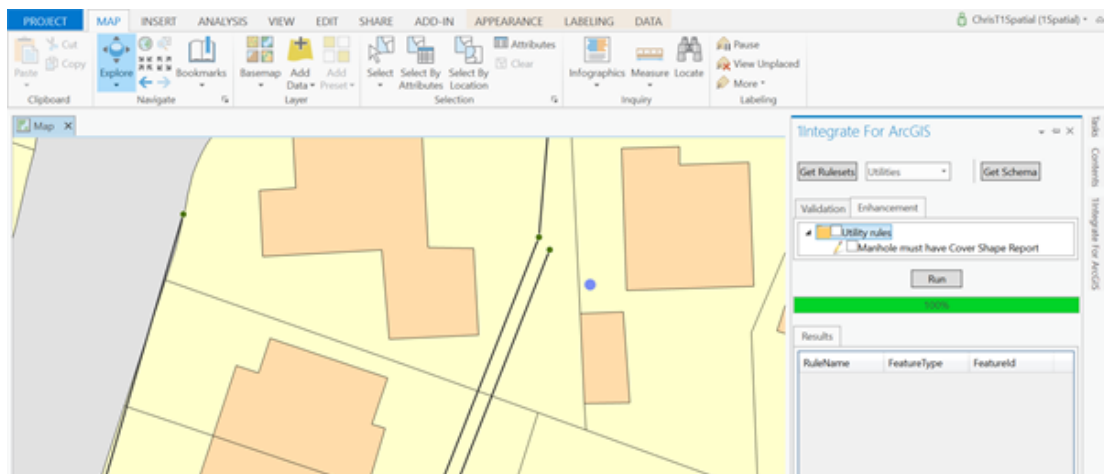


Figure 7-9: Manhole action displayed in the Enhancement tab

11. Run the enhancement action. If any features fail a rule, then new Issue features will be created.

In our example image below, new issue features have been created on Manhole features where the **COVER_SHAP** attribute is set to **U**. The Issue class representation has been set up to be a question mark

symbol .

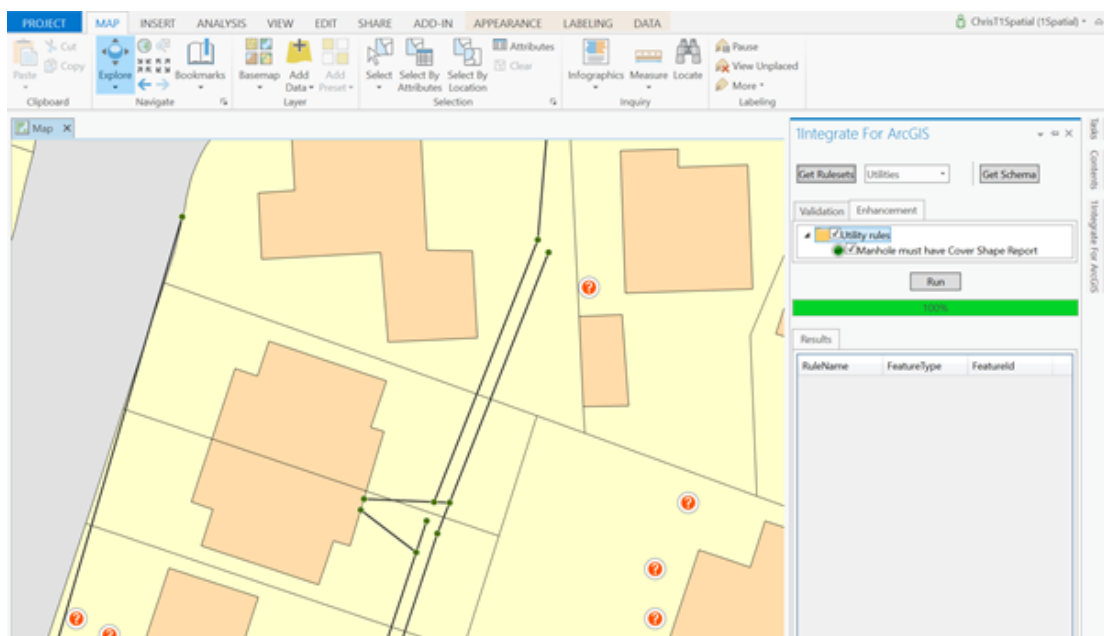


Figure 7-10: New issue features are created

12. On inspection we can see that the Issue feature contains the information set up in the Issue Management configuration.

It contains the GlobalID of the underlying feature that has failed (**OBJECT_REF**), and also the name of the rule that has failed (**DETAIL**).

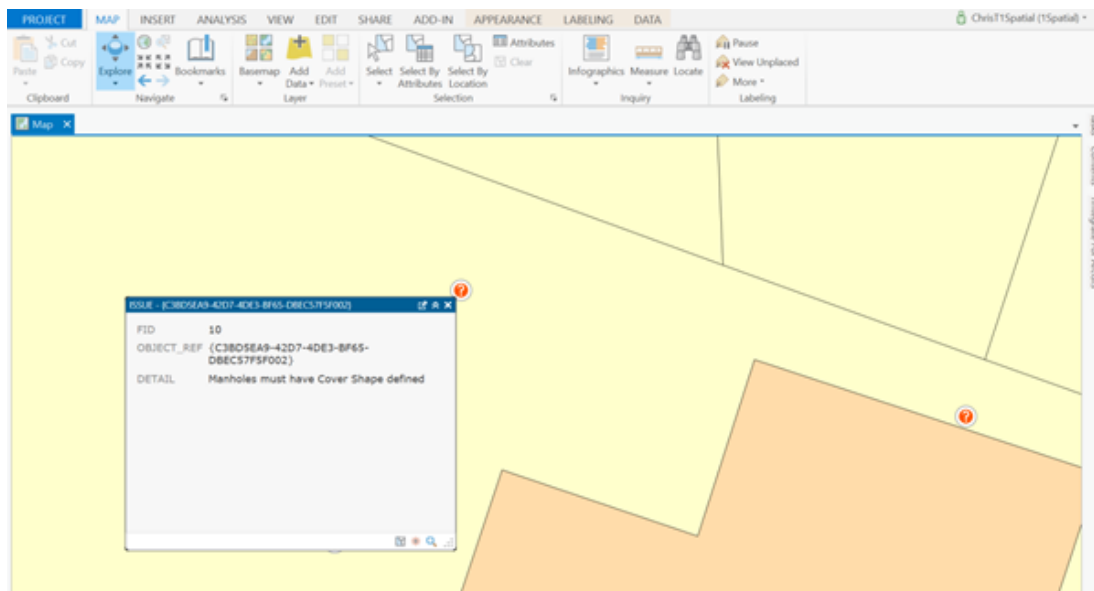


Figure 7-11: *Issue feature attributes*

13. If the user then corrects a feature that failed validation (in this case giving the Manhole feature a cover shape) then re-runs the action, Issue Management will automatically delete the Issue feature if it now passes validation.

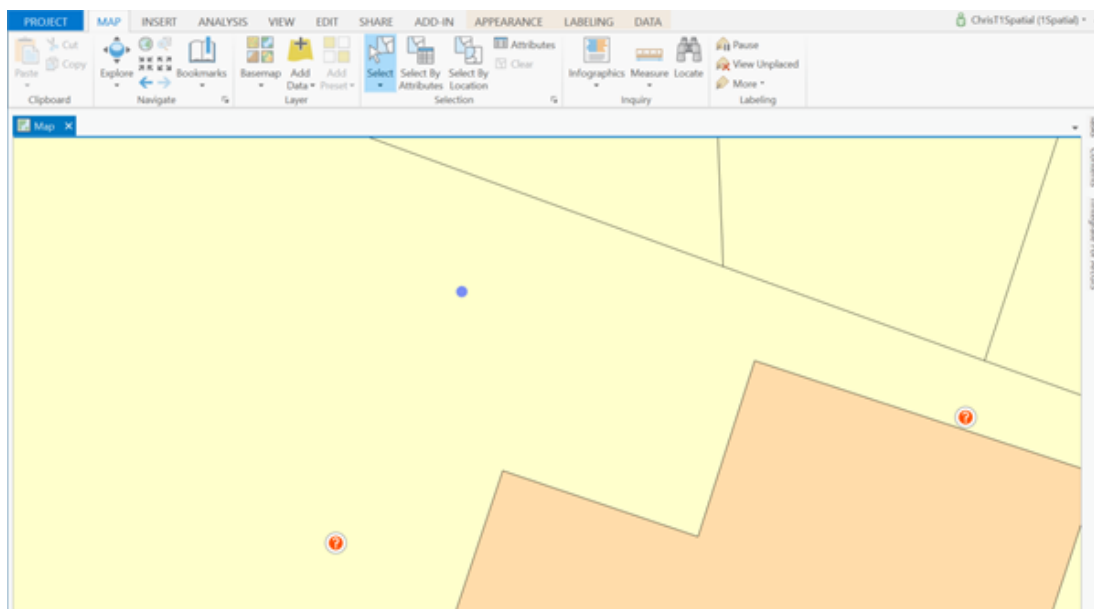


Figure 7-12: *Issue feature removed*



Note: If the 1Integrate for ArcGIS Mobile extension is installed, then users do not run the actions through the 1Integrate for ArcGIS Add-in or widget. The Mobile extension can be configured to run rules and actions as soon as anything changes within the map service. When Issue Management is used in this way, then as soon as a user edits features, Issue Management actions are run to determine if issue features need to be created. Similarly when a user corrects a feature, Issue Management will remove issue features automatically without the user having to do anything.

8 Backup and Restore

Backup and Restore can be performed in the **Admin Backup/Restore** page, accessed via the **Settings Menu**.



Note: Individual rulesets (such as the [free ruleset](#)) can be uploaded and downloaded as **.rules** files via the Rule Author menu (see "Rule Sets" on page 20).

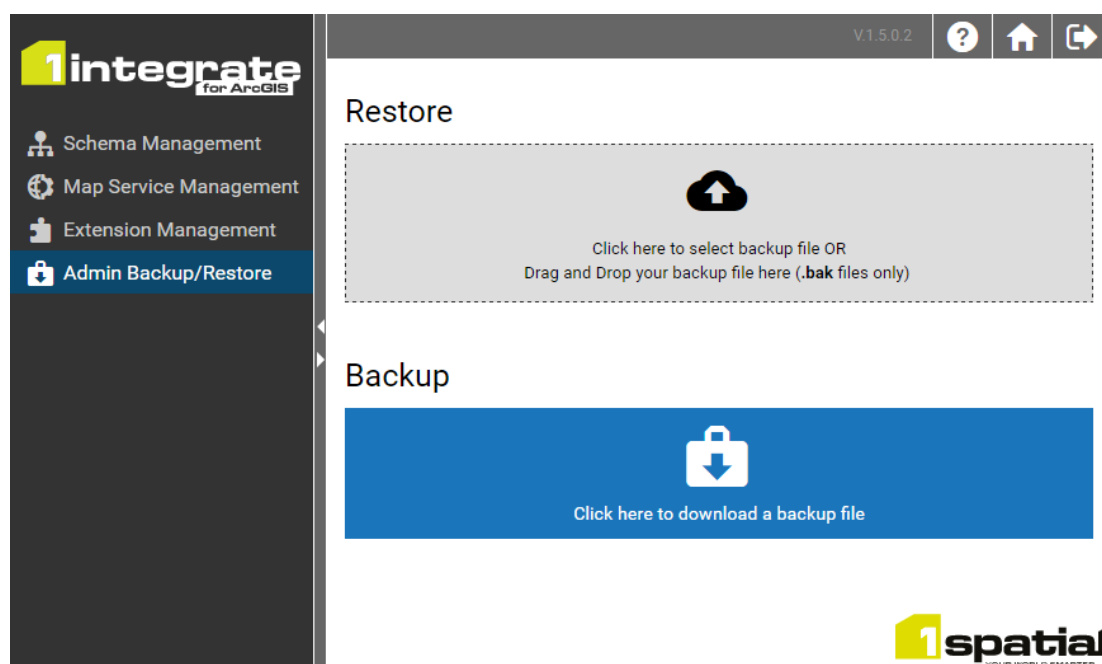


Figure 8-1: Admin Backup/Restore page

Backup

Upon clicking the download button in Backup part of the page, a file named **ruleauthor.bak** will be downloaded via your web browser.

Restore

A **.bak** file can be uploaded in the Restore part of the page, restoring the contents of the Rule Author to a previous version, or to duplicate another Rule Author.



Note: Uploading a **.bak** file will override all current Rulesets and any Packages, Rules and Actions that they contain.



Note: It is also possible to upload XML data. Please contact 1Spatial for further information.