



1 Spatial

Concepts guide

March 2010

Version 2.1



©1Spatial Group Limited

All rights reserved. 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 Group Limited.

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

Telephone: +44 (0)1223 420414

Fax: +44 (0)1223 420044

<http://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 Group Limited reserves the right to change the specification of the software. 1Spatial Group Limited accepts no liability for any loss or damage arising from use of any information contained in this document.

Contents

Introduction	4
Why Radius Studio?	5
Radius Studio concepts	9
Radius Studio roles	13
Data stores and schema mapping	15
Rule discovery	17
Data quality	18
A brief tour of the user interface	21

Introduction

This guide provides the Radius Studio user with an introduction to the concepts of the software.

Documentation set

This guide is part of a documentation set which also contains the following documents:

- Radius Studio release notes
- Radius Studio installation guides (Oracle Application Server and JBoss Application Server)
- Radius Studio user guide (online help)
- Radius Studio programmer's reference guide

Why Radius Studio?

This chapter explains the business drivers behind the design of Radius Studio.

The value of spatial assets

Our society has gathered vast quantities of geospatial data over the years, often spending large sums of money in order to derive the information we need to make business critical decisions. Much of this data collection has been done in isolation, resulting in costly duplication and data that is inconsistent or poorly documented. Huge amounts of data are accumulated in disparate and proprietary formats resulting in single-purpose, high maintenance data silos that no one else within the organisation can access. All too often, organisations who rely on spatial data for decision making find that the relevant data is only partially available because they cannot find what they need, they have no way of accessing all of it, or that data belonging to a third party is not suitable for their needs. This leads to decision-making being a slow, error-prone and typically manual process based upon partial information.

Using spatial information beyond simply looking at it on a map has challenged our traditional assumptions about how the underlying data is stored and accessed. We are left bursting with jumbled-up, often irrelevant data with no way of replicating the storage or managing the knowledge contained from one creator to another.

Exploiting our knowledge

Spatial data is a key asset to be viewed as a corporate resource, not a departmental luxury. Organisations recognise they need to be much more agile and be able to add and adapt business processes quickly. Digital data is being collated in every process. Centralisation, integration and interoperability are the concepts of the moment and many organisations have recognised the need for them. However, many of the point solutions to implement these concepts fall short for two reasons. Firstly, there is no rigorous, measurable control over the quality of the data, which means that errors can recur even once they have been fixed. This leads to a time-consuming, ongoing and expensive clean up operation. Secondly, simply sharing the data does not necessarily mean sharing the knowledge, or how to use or re-use the data. Knowledge within an organisation is often incredibly difficult to access. It is inside people's heads, hidden away in the end application code, or tucked away in paper specifications. It is therefore important for the people who produce data and information to capture that knowledge, organise it and make it visible to others. Critically, this allows the knowledge to be validated to make sure it is indeed accurate.



Our investment in data and information is now supporting an emerging discipline of knowledge management.

The data pyramid illustrates the challenge common to modern life - you've got to refine large volumes of raw data to acquire the more valuable information, knowledge, and (most rarely) wisdom that rest upon that super-abundant but heterogeneous foundation.

The knowledge factory

Knowledge is a difficult thing to quantify. But, we can learn a lot from the approach of other industries. Today, many industries strive to gain competitive edge by improving efficiency (by reducing workflow time frames), improving cost-effectiveness (by reducing errors and data redundancy) and minimising risk (by incorporating comprehensive compliance and audit checking). The financial sector has embraced these principles and seen huge savings by adopting straight through processing (STP) - “the seamless integration of systems and processes to assist with the automated flow of data without the need for manual intervention”. By taking the risk of human error out of data processing workflows and allowing computerised services to gather, analyse and process data according to rigorous rules, we achieve our workloads with greater efficiency and more uniform quality.

By using tools to bring order and structure to our workflows we can establish an enterprise-wide knowledge management environment for spatial data. This can be used as and when it is required to enable us to make critical business decisions with all the relevant and correct information at our fingertips, safe in the knowledge that it is fit for our particular purpose: safe because it is tested and documented to be so. Radius Studio provides the framework for knowledge engineering for spatial technologies.

The knowledge engineering platform enables the people who really understand the data to define the rule base without any requirement for programming. It eliminates the systems engineering chasm, where the need to brief programmers results in misunderstandings and long lead times in attempts to develop the system.

Radius Studio incorporates these key features:

- the ability to window into scattered spatial data to be able to assess the data to establish its fitness for purpose and facilitate its re-use.
- data mining techniques to analyse data and identify statistically dominant patterns. Users can then determine whether the patterns that have been identified are indeed valid business rules and then have the ability to fine-tune them.
- rules stored as enterprise metadata in an open, portable knowledge base. Rules are created as conceptual models and detached from any physical data model (ontology creation). Users from different parts of an organisation collaborate in evolving and maintaining the rule base.
- reporting tools to identify non-conforming data. Minimisation of risk by ensuring data adheres to regulatory or in-house compliance rules (SLAs, ISO9000) or performance measurement (for example balanced scorecard) and therefore able to certify data.
- an interoperable service-oriented architecture (SOA) delivered through the implementation of standards (ISO19119).
- spatial rules engine to automatically fix up non-conforming objects.
- persistent quality metadata recording the results of conformance checks in standards-compliant form (ISO19115).
- extensibility: the ability to extend the system with additional built-in functions, operators and so on through Java programming using the Radius Studio API.

Radius Studio concepts

This chapter defines concepts used in Radius Studio.

Action

An action is a procedure or process to be applied to one or more objects, usually when they are found to violate a particular rule. For example an action could modify a geometry or attribute in order to fix it (a fix-up action).

Data store

A data store is an external source of data (expected to be geographic data). The data can be in either Oracle 10g or 11g databases, or in vector file formats which are accessed through feature data objects (FDO). Rules are applied to data drawn from one or more data stores.

Discovery

The process of analysing data from a data source, looking for patterns in it, and using statistical techniques to deduce a plausible set of rules that the data appears to satisfy. Discovery, or spatial data mining, is an important part of building a spatial knowledge base. Combined with information from published specifications and knowledge from specific individuals, it may be able to find rules that would otherwise have been omitted.

Folder

A folder is a logical grouping of rules or actions. The Radius Studio interface mimics the Windows Explorer interface, using a tree of folders and sub-folders to store rules and actions. Note that Radius Studio folders do not correspond directly to file-system directories.

Task

An abstraction referring to some process applied to data from a data store. Specific tasks performed by Radius Studio include:

- Open data
- Discover rules
- Check rules
- Apply action
- Apply action map
- Commit data
- Copy to...
- Build Topology

A task may take seconds, minutes, hours or days. Therefore it executes asynchronously, so that the user can monitor progress and abort sessions. Tasks continue processing even if the user who started them logs out or shuts down the client computer. All status values and reports associated with the task are retained.

Session

A session is a sequence of tasks that describe a workflow. For example, open data from one or more data sources, check conformance against selected rules, apply rule-based transformations to resolve problems and commit the changes to the original data source(s).

Grid processing

Grid processing allows Radius Studio to take advantage of the resources of multi-core machines and multiple machines. As a result, Radius Studio is a scalable system - if more processing power is needed it can be easily added by adding more hardware resources to the grid. A session can be automatically split into multiple partitions that can be run in parallel on different processors using a priority based system and will be automatically scheduled to take full advantage of the available hardware. Discovery and handling of new, failed and stopped session queues is handled appropriately, with fail-over to alternative processors if required.

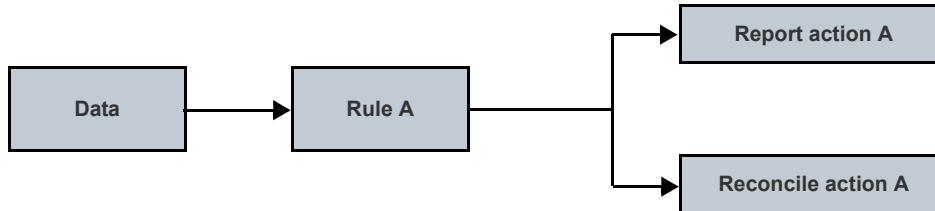
Rule

One or more conditions that objects from the data store should satisfy. A rule in Radius Studio is a structured tree of hierarchical conditions, against which objects can be tested. Rules are expressed in a form independent of the schema of any particular data store. This means they can easily be re-used with different data sources. In some rules-based systems, the term rule refers to a fact-pattern-action triplet:

- the data itself
- the conditions it should satisfy
- procedures to execute if it does not

Radius Studio uses this fact-pattern-action paradigm. However, the term rule refers solely to the pattern part of such a rule. That is, the conditions that the data should satisfy. The action to be performed is selected independently of the pattern and may be simply a reporting action (in the web browser, to an XML file or to a message queue) or a reconciliation action that applies algorithms to correct a problem.

The term action refers to the procedures or processes to be applied to the data when the condition is not satisfied. Both rules and actions are defined using the GUI provided in the web interface or through other programmes using the web services API.



Radius Studio roles

This chapter explains the roles that may be adopted by users of the system.

Each user account in the system may be allocated one or more of the following roles or privileges. This affects which parts of the system the user may access.

Administrator

A user who can set up system parameters and has all the privileges of the other roles. These include creating and modifying rules, data stores, sessions, actions and action maps. An administrator can also define sessions.

Data engineer

A user who may create and modify data sources, define and execute rule-based transformations or commit data to an enterprise data source.

Data loader

A user who may connect to vector formats such as MapInfo TAB, MapInfo Mid/Mif, ESRI Shapefile and Autodesk SDF using Feature Data Objects (FDO), for analysis in Radius Studio.

Data quality steward

A user who is responsible for overseeing the development of the rule base, testing data against it and providing feedback for on-going rule development.

Rule definer

A user who identifies potential rules in an enterprise data source and defines rules in the system. A rule definer may create and modify rules, actions and action maps.

User

A user who only has read access to Radius Studio. The user can view functions in the running application including conformance reports and sessions.

Data stores and schema mapping

This chapter explains how data stores and schema mappings are used to connect Radius Studio to an external data repository.

A data store is an external repository for data, usually including a geographic component. The data can be in either Oracle 10g or 11g databases, or in vector file formats which are accessed through Feature Data Objects (FDO). FDO is an open source component for manipulating, defining, and analysing geospatial information regardless of where it is stored. For supporting a variety of geospatial data sources, FDO uses a provider-based model where each provider typically supports a data format or data store. Initial formats supported in Radius Studio (read only) are:

- WFS
- ESRI Shapefile
- Autodesk SDF
- MapInfo TAB and MIF/MID

For more information on FDO see <http://fdo.osgeo.org/>

The user selects some or all of the data in a store. Radius Studio checks it for conformance to a defined set of rules and optionally applies automated corrections. The corrected data may be returned to the same data store or a different data store.

The schema mapping defines how data should be converted between the schema of a data store and the internal schema used by Radius Studio. The mapping translates between either relational database tables and columns or vector file layers and attributes to classes and properties in the Radius Studio workspace. The user selects the data to import and defines the names of the corresponding classes and properties.

It is possible to define several data stores that access the same data through different schema mappings. Data can be read from several data stores into Radius Studio for processing against a set of rules that analyse relationships between datasets as well as within a dataset.

Each data store has two schema mappings associated with it – one input and one output. Output mapping is not needed if you are only checking rules without changing data. It is also possible to input from one store and output to a different store.

Rule discovery

This chapter explains how Radius Studio can discover new rules through analysis of existing data.

Rules may come from a variety of sources including formal and informal specifications, application and data model source code and the knowledge of experts within an organisation. A further potential source of rules is the data itself. Rule discovery in Radius Studio is the application of data mining techniques to spatial data. Sophisticated analysis of dominant statistical patterns in the data may identify rules that have never been formally recorded; in some cases they may never have been recognised. By identifying these rules, Radius Studio can be used to automatically extend the rule base, increasing confidence in the quality of the data and therefore increasing its value.

The main difference between data mining in non-spatial and spatial environments is that attributes of features that interact spatially or are nearby an object of interest need to be considered. New techniques are needed for analysing the spatial relationships, such as topological, distance and direction relations. These relationships are considered, in addition to attribute relationships, to identify those relations sufficiently common to suggest they should always hold.

There are a number of different approaches and algorithms that can be used for spatial data mining. The rule discovery interface in Radius Studio is generic, allowing different algorithms to be used to identify as wide a range of rules as possible.

Data quality

This chapter discusses the quality of data and how it is managed and enhanced by Radius Studio.

Quality control

Data quality is about fitness for use and freedom from defects. Data should be accessible, accurate, up-to-date, manageable, consistent with other sources, complete and comprehensible.

Quality control is the process of monitoring and managing the quality of the data: how well it matches the specification for the data and therefore whether it is fit-for-purpose. In the absence of quality control, data becomes unreliable and unsuitable for re-use. Business decisions based on the information may be incorrect and the consequences may be very serious. Customers of the data may seek other suppliers who can provide consistently higher quality data.

Radius Studio provides tools that can be used to control the quality of a master database and the quality of data that is to be submitted to update it. Quality control procedures using Radius Studio can ensure it is updated only with high quality data. Users define the appropriate quality level for each dataset; Radius Studio can then verify fitness for purpose and provide appropriate certification of data quality.

Data quality improvement cycle

The data quality improvement cycle is a workflow for creating datasets that are fit-for-purpose. The process starts by agreeing the quality targets for the data and defining a set of spatial business rules that can be used to assess the data against these targets. A baseline assessment analyses the data to determine whether it achieves those targets. If it does not, a cycle is entered where:

- errors are corrected, automatically and/or manually
- rule discovery looks for new rules that can be identified in the higher quality data
- rule conformance is assessed and the results compared to the quality targets

When the data meets the defined targets, it can be certified and published. Consumers of the data can be confident it will meet their needs.



Knowledge management

Spatial knowledge management uses the same processes as management of more traditional business data:

- business rules are stored as metadata in an enterprise database.
- universal access is provided throughout the organisation, allowing any users with the appropriate privilege to define, refine and review the business rules.
- data is checked automatically against the business rules or a defined subset of those rules.
- rules driven data reporting and reconciliation resolves identified non-conformances.
- the data is published with certification that it has met the defined quality targets.

The only changes to this process when working with spatial data are in the nature of the rules themselves (“All gas pipes must be connected to a valve or a cap”, or “land parcels must never overlap”) and the corrective actions.

Radius Studio provides this environment by storing all metadata in an Oracle database and providing Internet-wide access through web browsers and web services.

A brief tour of the user interface

This chapter explains the basic principles behind the user interface and describes the main components.



The previous picture illustrates Radius Studio running in the Internet Explorer web browser, requiring no additional software to be installed on the client computer.

A row of tabs at the top of the user interface selects the component to work with. Some parts of this interface may be inaccessible or read-only depending on the privileges of the user logged in to the system. For example, a rule definer may create, edit and delete rules and actions but a data engineer is allowed only to browse them and use them to maintain and execute tasks such as checking rules as part of a session.

In this example, the rules component is selected, allowing the user to maintain the rule base. The basic principles described here apply to all of the components; an example of each is shown later. The rules are presented in a hierarchical folder structure, similar to that used by Windows Explorer. The path to the current folder appears below the main tabs, in this case:

```
Radius Studio >> Rules >> Philadelphia >> Fire_Stations 1.0
```

Philadelphia is the name of the folder and Fire_Stations 1.0 is the name of the selected rule. The main part of the window is divided into two areas. The form on the left is for creating, deleting and modifying rules and rule folders and for editing options that apply to all the rules. The form on the right is for browsing and editing the currently selected rule. Each of these forms provides tabs at the bottom that are used to view and edit different parts of this information.

The left hand form displays all rules in the current folder and a list of deleted items in the recycle bin. Items in the recycle bin may be restored or permanently deleted.

The **General** tab on the right hand side maintains the metadata about the rule:

- a name for the rule, chosen by the user.
- a free format text description of the rule.
- the date when the rule was created and the name of the user who created it, automatically maintained.
- the date when the rule was last modified and the name of the user who updated it, automatically maintained.
- other comments added by any user.

Data stores

The screenshot shows the 'Spatial' software interface. At the top, there is a navigation bar with 'Data Stores', 'Rule Discovery', 'Rules', 'Actions', 'Action Maps', and 'Sessions'. Below this, a breadcrumb trail reads 'Home > Rules > Data Stores > Philadelphia'. The main window is titled 'Map to ontology' and displays a mapping configuration for a data store named 'Philadelphia'.

On the left, a 'Data Stores' sidebar lists several options: Core Repository, Floorland, Philadelphia (selected), System Integrator, and Recycle Bin. The main area is divided into 'Source' and 'Target' columns. The 'Source' column lists various attributes from the 'Philadelphia' data store, and the 'Target' column lists corresponding attributes from an ontology. Each source attribute is mapped to a target attribute, with some mappings including data type dropdowns and checkboxes for 'Idc', 'Report', and 'Join'.

Source	Target
<input checked="" type="checkbox"/> BUILDING	→ BUILDING
<input checked="" type="checkbox"/> ADDRESS VARCHAR2	→ ADDRESS (String)
<input type="checkbox"/> BLDGS_NUMBER	→ BLDGS_NUMBER (Real)
<input type="checkbox"/> BLDGS_ID_NUMBER	→ BLDGS_ID (Real)
<input type="checkbox"/> BUILDING_TYPE VARCHAR2	→ BUILDING_TYPE (String)
<input type="checkbox"/> CEPT VARCHAR2	→ CEPT (String)
<input checked="" type="checkbox"/> FCODE_NUMBER	→ FCODE (Real)
<input type="checkbox"/> FSTATION_NUMBER	→ FSTATION_NUMBER (Real)
<input checked="" type="checkbox"/> GEOMETRY SDG_GEOMETRY	→ geometry (Geometry) <input checked="" type="checkbox"/> Join <input checked="" type="checkbox"/> Idc <input checked="" type="checkbox"/> Report
<input checked="" type="checkbox"/> ID_NUMBER	→ ID (Real) <input checked="" type="checkbox"/> Idc <input checked="" type="checkbox"/> Report
<input checked="" type="checkbox"/> SITE_NAME VARCHAR2	→ SITE_NAME (String) <input checked="" type="checkbox"/> Idc
<input checked="" type="checkbox"/> CURSINE	→ CURSINE
<input checked="" type="checkbox"/> FIRE_STATION	→ FIRE_STATION
<input type="checkbox"/> POLITICAL_WARD	→ POLITICAL_WARD
<input checked="" type="checkbox"/> STREETCENT	→ STREETCENT
<input type="checkbox"/> ZONING	→ ZONING

At the bottom of the window, there are tabs for 'General', 'Input Details', 'Input Mapping', 'Output Details', and 'Output Mapping', along with a 'Save' button.

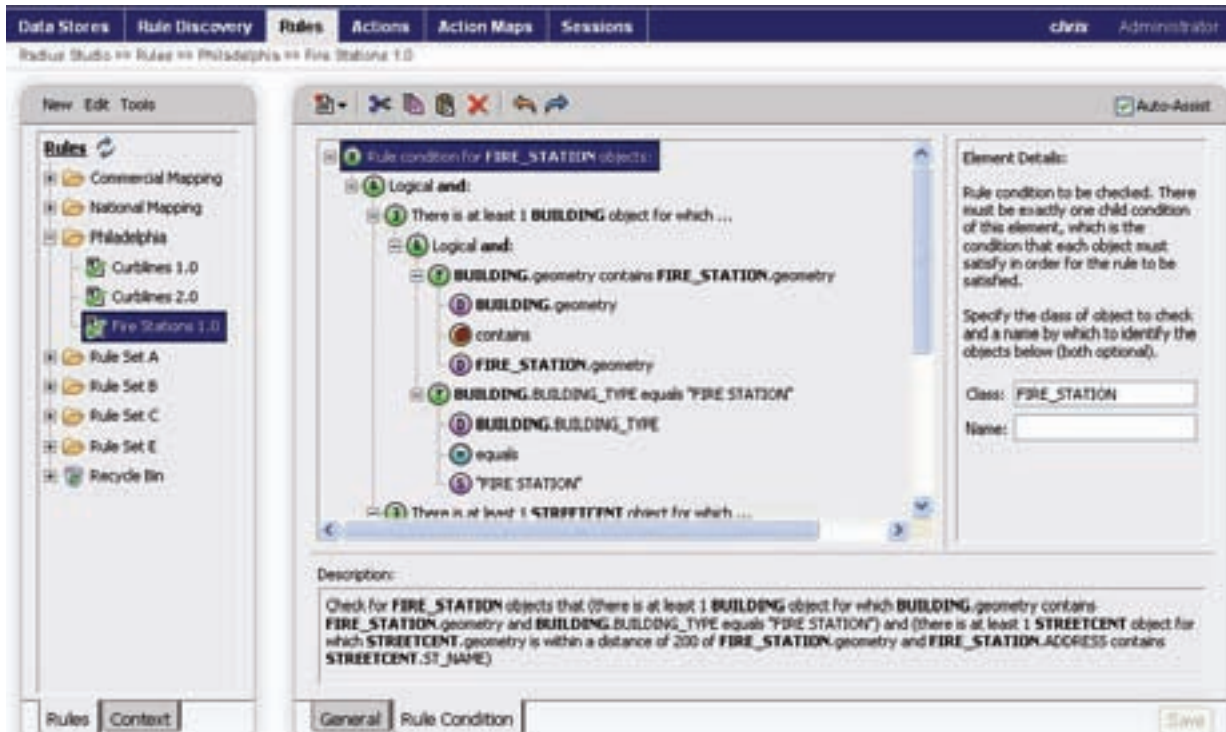
The data stores interface is used to define connections to Oracle 10g or 11g databases or vector file formats accessed through feature data objects (FDO), which contain data to be processed by Radius Studio. Each data store contains the following information in addition to the standard metadata described in the previous section:

- input details: the information required to connect to the source data for an open data task.
- input mapping: a mapping from the source data schema to the object-oriented schema in Radius Studio.
- output details: the information required to connect to the Oracle database and schema for a copy to... task.
- output mapping: a mapping from the object-oriented schema in Radius Studio to the relational schema in Oracle.

The example on the next page illustrates the input mapping interface. The names such as `BUILDING` and `CURBLINE` refer to tables accessed through the input connection. The `FIRE_STATION` table has been expanded to show the columns in the table. The ticks on the left hand side are used to select which tables and which columns from each selected table are to be analysed by Radius Studio. All other data is ignored.

The text boxes on the right hand side map table names onto classes and column names onto properties in the Radius Studio schema. The type of each property may also be overridden, for example to decide whether to represent a `NUMBER` as a real or an integer. By default, the interface is populated with an identity mapping.

Rules



The rule builder allows the definition of potentially complex rules with a sophisticated but easy-to-use interface. The rule is expressed as a series of clauses built up using pull-down menus from the bar immediately above the graphical illustration of the rule.

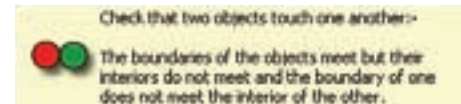
The description at the bottom provides English text representing the currently selected clause, in this case the complete rule. The element details are used to specify the parameters associated with the currently selected rule. This part of the form always includes a description of the information required. In this case, the top-level rule specifies the class to be checked. An optional name is used when the rule needs to distinguish between two different features of the same class.

There are four types of rule clauses:

- conditions, including comparisons, logical operators (AND, OR, NOT), IF...THEN...ELSE, existence and looping constructs.
- values, including constants, dynamic values (attributes), built-in functions, aggregates, and so on.
- relationships, including equals, less than, begins with, contains, and so on.
- spatial relationships: Radius Studio implements all of the Open Geospatial Consortium spatial operators: equal, disjoint, intersect, touch, overlaps, cross, within, contains and geometric relationships like covers or covered by and within distance or beyond. Tool tips are provided to clarify each of these operators.

While editing a rule, it may temporarily be incomplete until a new clause is added or parameters are defined. These problems are highlighted clearly in red and a description of what is required displayed.

Multi-level undo/redo ensures it is easy to recover from mistakes while editing. Drag and drop can be used to reorder clauses of a rule. Cut and paste can be used to transfer all or part of a rule into another rule.



Rule discovery

Algorithm	Spatial Boosting Algorithm ▾	
Within-distance tolerance	<input type="checkbox"/> 0.0	(default: 0.0)
Minimum support cutoff threshold	0.2	(default: 0.2)
Lower cutoff threshold	0.5	(default: 0.5)
Minimum rule probability	0.8	(default: 0.8)
Minimum probability improvement ratio	2.0	(default: 2.0)
Maximum number of interacting objects	20	(default: 20)

General Specification Save

The rule discovery interface is used to select a data mining algorithm and define parameters to control the discovery process. In Radius Studio one algorithm is provided, which is a variant of the boosting algorithm.

The `within distance` tolerance may be used to consider features that are nearby, but do not interact with it. Any features that approach within this distance will be analysed. The other parameters are described in the on-line help. They require more knowledge of the algorithm than is appropriate to cover in this document and the default values are usually appropriate.

The sessions interface is used to execute the discovery process based on a selected specification. This allows further parameters to be defined:

- a spatial region within which to analyse data
- a restricted set of classes to analyse
- the maximum number of objects to consider

The results from a rule discovery task are presented as a table of candidate rules and some statistics related to the confidence that the rule is correct. Any of these rules may be promoted to real rules that can be used in conformance checks and rule-based transformations. A single button click next to the rule will prompt for a name and then save the rule, after which it may be edited by the rule builder and/or selected for use in a session.

Actions

The screenshot shows the ArcGIS Desktop interface with the 'Actions' pane open. The top navigation bar includes 'Data Stores', 'Rule Discovery', 'Rules', 'Actions', 'Action Maps', and 'Sessions'. The user is logged in as 'chris Administrator'. The breadcrumb path is 'ArcGIS Desktop > Actions > Philadelphia > firestation.tbx'.

The 'Actions' pane on the left shows a tree view with 'Philadelphia' selected, containing 'firestation.tbx', 'moorland', and 'Recycle Bin'.

The main 'Action Definition' pane shows a rule for 'FIRE_STATION' objects. The rule is defined as follows:

- For all **BUILDING** objects for which...
 - Logical and:
 - FIRE_STATION**.geometry is within a distance of 200 of **BUILDING**.geometry
 - FIRE_STATION**.geometry
 - is within a distance of 200 of
 - BUILDING**.geometry
 - BUILDING**.BUILDING_TYPE equals 'FIRE STATION'
 - BUILDING**.BUILDING_TYPE
 - equals
 - 'FIRE STATION'
 - Let **FIRE_STATION**.geometry = get_point(**BUILDING**.geometry)

The 'Element Details' pane on the right provides instructions: 'Action to be performed. There must be exactly one child operation of this element, which is the main operation to be performed. Multiple operations may be performed by using a sequence.' It also includes a 'Class' field set to 'FIRE_STATION' and an empty 'Name' field.

The 'Description' field at the bottom contains the following text: 'For **FIRE_STATION** objects: for all **BUILDING** objects for which(**FIRE_STATION**.geometry is within a distance of 200 of **BUILDING**.geometry and **BUILDING**.BUILDING_TYPE equals 'FIRE STATION') do let **FIRE_STATION**.geometry = get_point(**BUILDING**.geometry)'. The 'Save' button is visible in the bottom right corner.

The action builder uses the same style of interface as the rule builder, but provides extra operations such as the ability to change the values of object properties.

The description of a sample action is shown below:

For FIRE_STATION objects:

```
for all BUILDING objects for which
```

```
(FIRE_STATION.geometry is within a distance of 200 of BUILDING.geometry and  
BUILDING.BUILDING_TYPE equals 'FIRE STATION') do
```

```
let FIRE_STATION.geometry =  
get_point(BUILDING.geometry)
```

In the above example, `get_point()` is a built-in function. This is very common in actions, which can choose from a wide range of built-in functions, including:

Conversion:

- `to_radians`
- `to_degrees`
- `to_integer`
- `to_real`
- `to_string`

Mathematical, for example

- `abs`
- `min`
- `max`
- `sin`
- `cos`

Bit manipulation:

- `bit_and`
- `bit_or`
- `bit_xor`
- `bit_not`
- `bit_shift`

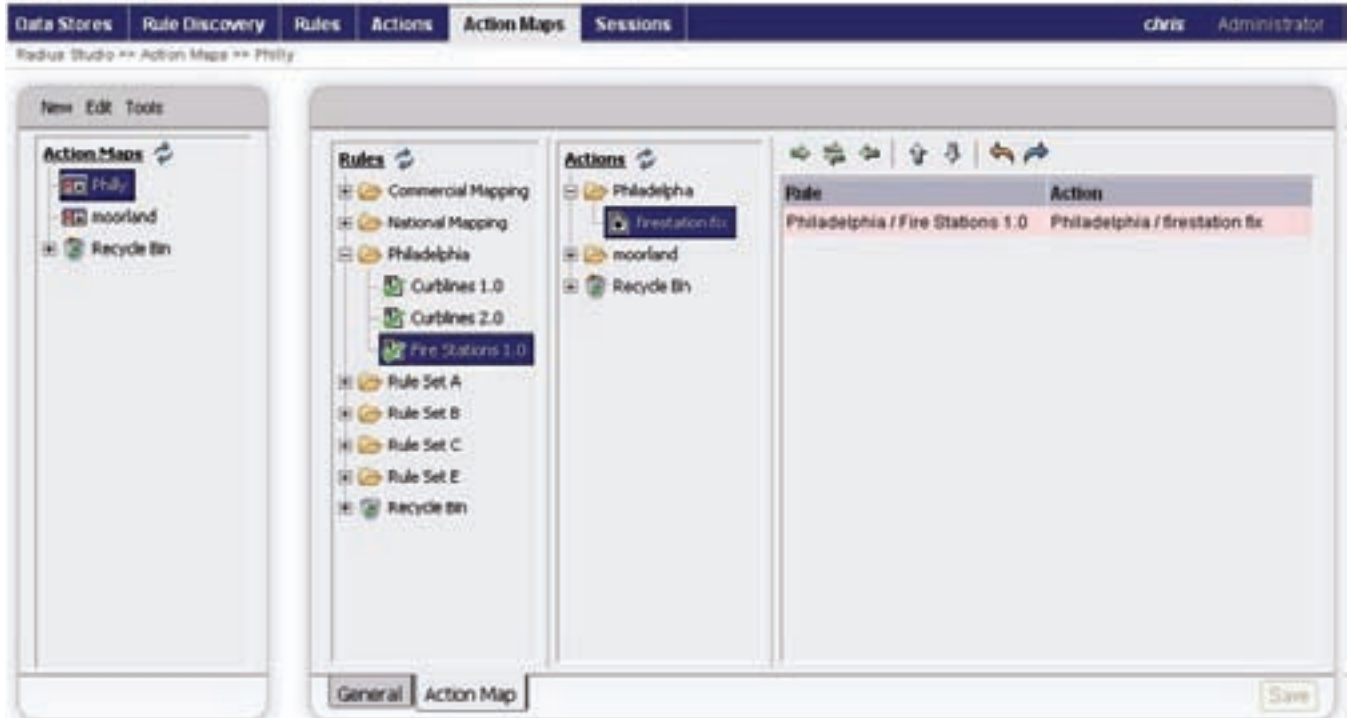
String functions, for example:

- `substring`
- `re_search`
- `re_subs_all`
- `to_uppercase`
- `to_lowercase`

Geometric, for example:

- `area`
- `buffer`
- `convex_hull`
- `difference`
- `distance`
- `douglas_peucker`
- `intersection`
- `outer_ring`
- `remove_small_loops`
- `remove_snapbacks`
- `remove_spikes`
- `union`

Action maps



Action maps are used to connect rules to actions. The lists in the centre of the screen can be used to browse all folders of rules and actions to select an action to be invoked as part of a rule-based transformation whenever a feature is found that does not conform to the rule.

If the left hand side of the pair is left blank – so that no rule is specified – the action will be applied to all objects in the relevant class.

The action map is a simple table of rules and the corresponding action. It is easy to add more rows to the table and to change the data in existing rows. Multi-level undo / redo may be used to correct mistakes. In the sessions interface, the apply action map task uses an action map to specify the rules to check and the actions to invoke.

Sessions

The screenshot shows the 'Sessions' window in the Spatial software. The window title is 'Sessions' and it is part of the 'Philadelphia session'. The interface includes a toolbar with various icons and a 'Run as multiple partitions' checkbox. The main area displays a list of tasks with their progress and status:

Task Name	Progress	Status
Open Data: (Philadelphia)	100%	✓
Check Rules: (Philadelphia / Curblines 1.0)	100%	✓
Pause	100%	✓
Check Rules: (Philadelphia / Curblines 2.0)	100%	✓
Pause	100%	✓
Apply Action Map: (Philly)	100%	✓
Commit (Philadelphia)	Running	🔄
Pause		

The 'Commit (Philadelphia)' task is highlighted in purple and has a 'Running' status with a circular arrow icon. Below the task list, there are 'General' and 'Tasks' tabs, and a 'Save' button in the bottom right corner.

The sessions interface allows the construction of an ordered sequence of tasks to process data. Task types are:

- **Open data:** enables access to data from a defined data source. A session may choose to open data from a number of data sources and then check rules based on relationships between features stored in different locations.
- **Discover rules:** analyses data based on a discovery specification to identify candidate rules.
- **Check rules:** checks a defined set of rules on the data and reports non-conformances. It is also possible to publish conformance checking results to a catalog server (for example INdicio from Galdos Systems Inc). Metadata consists of data quality items (qualitative and quantitative metrics) and is encoded in a standard form (TC211 ISO XML - 19139).
- **Apply action:** applies one or more actions to the data.
- **Apply action map:** checks a set of rules defined in an action map and applies the associated action to each non-conforming object.
- **Commit data:** incrementally commits any data changes back to the data store it came from. Typically this is after a correcting action has been applied by an action or action map.
- **Copy to...:** copies data to a different data store.
- **Pause:** requests Radius Studio to suspend processing to allow results to be examined before processing the next task.
- **Build Topology:** works out the relationships between the different types of objects in your Radius Studio session and stores this information as a set of topology.

The interface allows tasks to be added, deleted and reordered and for all required parameters to be browsed and updated. Multi-level undo and redo may be used to correct mistakes.

Media player style controls are used to run tasks.



The play button starts a task or resumes a paused task.



The pause button suspends execution of a task.



The rewind button rewinds the most recent task performed.



The stop button discards all data and results and rewinds to the start of the first task.

The session interface allows the results of tasks to be viewed. For check rules tasks, Radius Studio provides XML and HTML reports that give metrics on the conformance levels of data, against the rule(s) that have been run. These reports include a high-level summary and the IDs of the objects that fail the rule, which the user can use to verify the data is fit for purpose. The results can also be published to a geographic information catalog server.

Spatial

Rule Conformance Results

Started: 29-Aug-2009 11:53:10 Finished: 29-Aug-2009 11:53:37 Duration: 28 Seconds

Objects Checked: 21274 Number Passed: 21267 Number Failed: 17

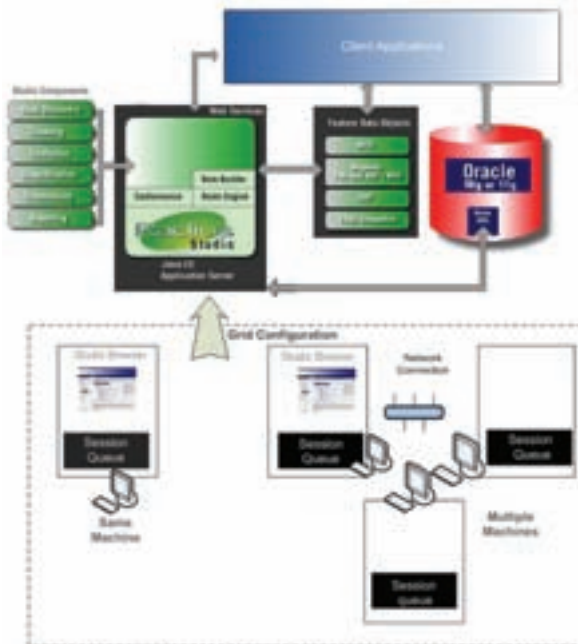
Conformance: 99.99% % Passed: 99.99 % Failed: 0.01

Conformance Metadata: [View XML](#) [Publish](#) Number of Errors: 0

Rule	id	geometry
RAE.MC.V04_Installed Server.nodes	129676 0	WDR min (450201.026442165, 5403641.81273577) max (450201.026442165, 5403641.81273577)
RAE.MC.V04_Installed Server.nodes	116716 0	WDR min (450204.894494495, 5402671.70220948) max (450204.894494495, 5402671.70220949)
RAE.MC.V04_Installed Server.nodes	302130 0	WDR min (4502161.826133834, 5440462.40483379) max (4502161.826133834, 5440462.40483379)
RAE.MC.V04_Installed Server.nodes	107887 0	WDR min (4502271.805458986, 5440182.86210751) max (4502271.805458986, 5440182.86210751)
RAE.MC.V04_Installed Server.nodes	501051 0	WDR min (4497987.486279776, 5472877.76752178) max (4497987.486279776, 5472877.76752178)
RAE.MC.V04_Installed Server.nodes	117962 0	WDR min (4492070.167134421, 5430017.87822842) max (4492070.167134421, 5430017.87822842)
RAE.MC.V04_Installed Server.nodes	337830 0	WDR min (4492958.324454887, 5454055.80624677) max (4492958.324454887, 5454055.80624677)
RAE.MC.V04_Installed Server.nodes	109496 0	WDR min (4491130.155807627, 5434729.86176795) max (4491130.155807627, 5434729.86176795)
RAE.MC.V04_Installed Server.nodes	310933 0	WDR min (4489862.158832304, 5469989.95152299) max (4489862.158832304, 5469989.95152299)
RAE.MC.V04_Installed Server.nodes	109063 0	WDR min (4487935.488110887, 5465783.42444874) max (4487935.488110887, 5465783.42444874)
RAE.MC.V04_Installed Server.nodes	371889 0	WDR min (4486988.483795846, 5473768.99534379) max (4486988.483795846, 5473768.99534379)
RAE.MC.V04_Installed Server.nodes	128812 0	WDR min (4487983.483088871, 5472483.19628881) max (4487983.483088871, 5472483.19628881)
RAE.MC.V04_Installed Server.nodes	308479 0	WDR min (4483583.702841814, 5488312.40181783) max (4483583.702841814, 5488312.40181783)
RAE.MC.V04_Installed Server.nodes	118279 0	WDR min (4484488.543542911, 5451967.85853847) max (4484488.543542911, 5451967.85853847)
RAE.MC.V04_Installed Server.nodes	302712 0	WDR min (4482891.878448382, 5441933.92244888) max (4482891.878448382, 5441933.92244888)
RAE.MC.V04_Installed Server.nodes	308899 0	WDR min (4482783.302233874, 5471720.12128781) max (4482783.302233874, 5471720.12128781)
RAE.MC.V04_Installed Server.nodes	333781 0	WDR min (4487688.52434378, 5473488.8782318) max (4487688.52434378, 5473488.8782318)

Radius Studio Architecture

This chapter provides an overview of the Radius Studio architecture.



Radius Studio is built using the Java EE application platform, which provides a scalable, location transparent platform to all applications that run within it. An organisation may deploy end-user applications from a range of vendors across the organisation to enable use of best-of-breed solutions.

An enterprise Oracle database is used to provide a robust, reliable, secure and scalable repository, enabling all the departments to share data. Radius Studio works with spatial data held in Oracle 10g or 11g databases or in a variety of different vector formats (using FDO) to provide spatial data quality management and enhancement services across the organisation. Radius Studio processes data using 1Spatial's extensive, robust and efficient object and geometric processing software, and is built on the same core technologies as that deployed in Radius Topology and other 1Spatial products and solutions.

Radius Studio allows a grid configuration consisting of the Session Queue application to control sessions, and the Radius Studio interface (Microsoft Internet Explorer web browser or ISO19119 web services) running on single or multiple machines. Leveraging multiple machines, and, in some cases multiple processors, provides efficient session performance.