

Process Definition

1. Overview

Process definition is what you do to set up a RADRunner system.

A RADRunner system typically contains many different *process models* - groups of related Role types with their supporting entity and interaction types, as well as instances and associated users. Notionally, each process model corresponds to a particular business process. However, the boundary between process models is often hard to establish, since (for example) some Roles may participate, or some entities may be of use, in several business processes.

RADRunner caters for this overlap by removing the need to have boundaries in the first place. A *Playwright world* (RADRunner data store) simply contains a number of objects - types, instances and users. These can be imported, exported and used as required, singly or in any combination.

This document explains how to use RADRunner to set up support for your own business processes.

2. Step-By-Step Guide

Here is a step-by-step guide to process definition:

1. [Optional] The first step is generally to create *Role Activity Diagrams* (RADs) of your processes, to outline the Roles, activities and interactions you wish to include. We supply a graphical tool for making RADs - alternatively, any standard drawing tool can be used *Once you have designed your processes in outline, you need to enter them into RADRunner: the entity types used to store data, plus the Role and interaction types shown in your RADs.*
2. Log in to RADRunner as the [Boss user](#) - if you have more than one Role (initially, you will not), choose [42](#).
Alternatively, you can use [other Roles capable of process definition](#) - new instances of the 42 type, or instances of other Role types which contain suitable tasks (sub-types of 42, for example).
3. Add the entity types needed to store data - some may be used in multiple Roles, and passed from one to another via interactions.

4. Add the Role types in your process, making [public](#) those that you wish anyone to be able to use.
5. Add entity types to each Role as appropriate.
6. Add the interaction types needed for communication between Roles.
7. Add tasks to each Role, grouped into activities for transaction control.
RADRunner is supplied with many pre-built task types. Some are available in 42, and are used for [process definition](#) - adding Roles, entities, interactions, and so on. Other Role types can include these tasks if desired. However, the tasks usually assigned to other Role types are those for [process enactment](#) .
8. Create initial instances of your Role and interaction types.
9. Create an initial set of users.
10. Assign Roles to users as appropriate.

That's it. Now you need to start a process.

Often a process is started by creating a single Role instance and assigning it to a user, which can be done in a single step via the *Maintain Users* activity in 42 . Either assign the new Role instance to *boss* or (preferably) create a new user to act it.

It may not even be necessary to do this much. If suitable *public* Role types are imported into the initial 42 Role instance, anyone can register as a new user of the system via the *Register New User* page, and start a process by instantiating one of these Role types via the *Start New Role* page.

The initial Role instance then creates new users and starts other Roles as required, and these new Roles will do the same. Once users start logging in and using their Roles, the system will grow organically.

Once you have created a process, you may wish to [export the definitions as Playwright XML for backup](#) .

3. Example Process Models

To help with getting started, we provide some example processes:

- Report Ambulance Service Activity
- Stockbroker Call Centre

The processes are provided in archive files of zip format. To use the archives, extract their files to a directory, and [import the contents of the directory into RADRunner](#). For example, login as *Boss/42* and use *Maintain Frameworks/Import All Playwright Files In A Specified Directory*.

Some points to watch out for:

Process Definition

- Make sure that the directory used for import is either at a different location to the zip files, or has a different name than both zip files - otherwise, some operating systems with built-in zip support (e.g., Windows XP) may try to read from the correspondingly named zip file instead of the directory and no files will be imported.
- The *Report Ambulance Service Activity* process requires Internet access in order to access via HTTP the stylesheets used to generate custom entry forms (for submission and review of Ambulance Service Activity data). If Internet access is not available, you cannot run the process.
- The *Stockbroker Call Centre* process requires Internet access in order to call public web services (for example, for real-time stock quote generation). If Internet access is not available, you can still run the process, but each Quoter Role instance will suspend after the first quote is requested, since no further activities are then available until the quote has been generated.

Guidance on how to use the example processes is given in our Presentation On RADRunner.

4. The 42 Role Type

When RADRunner is first installed, there are only 3 objects in the world:

1. The Boss user
2. The 42 Role type
3. The root Role, an instance of the 42 Role type assigned to the Boss user - this is generally known simply as 42

The 42 Role type includes all the tasks necessary for process definition. Hence it is possible to extend a new system as follows:

1. [Log in to the Boss user](#) and 42
2. Create further users and Role types
3. Instantiate the Roles and assign them to the users

The new Role types can also include process definition tasks. The easiest way to set this up is to make them sub-types of the 42 Role type, removing any tasks from the sub-types that for security or other reasons you do not wish other users to possess (such as [Export World To A Specified Directory As Playwright Files](#)). Using this means, multiple users can be empowered to carry out process definition.

In particular, users engaged in a day-to-day business process can [evolve the system from within](#).

The 42 Role type includes the following activities and tasks:

- Maintain Role Types
 - [Add Role Type](#)

- [Add Entity To Role Type](#)
- [Delete Entities From Role Types](#)
- [Export Role Type Resources](#)
- [Delete Role Types](#)
- [Export Role Types As Playwright](#)
- [Import Role Type From Playwright](#)
- Maintain Interaction Types
 - [Add Interaction Type](#)
 - [Delete Interaction Types](#)
 - [Add Give To Interaction Type](#)
 - [Add Get To Interaction Type](#)
 - [Export Interaction Types As Playwright](#)
 - [Import Interaction Type From Playwright](#)
- Maintain Entity Types
 - [Add Entity Type Component](#)
 - [Delete Entity Type Components](#)
 - [Export Entity Types As Playwright](#)
 - [Import Entity Type From Playwright](#)
 - [Import Entity From XML File](#)
- Maintain Activity Types
 - [Delete Activities And Tasks From Role Types](#)
 - [Add Access Relational Database To Role Type](#)
 - [Add User Definition Tasks To Role Type](#)
 - [Add Computation To Role Type](#)
 - [Add Convert User To Entity To Role Type](#)
 - [Add Create Reference To User To Role Type](#)
 - [Add Download File To Role Type](#)
 - [Add External Component To Role Type](#)
 - [Add Get Entity From URL To Role Type](#)
 - [Add Get Entity From XSL To Role Type](#)
 - [Add Get Resource From Collection To Role Type](#)
 - [Add Introduce This Role Instance To Another Role Instance To Role Type](#)
 - [Add Manual Action To Role Type](#)
 - [Add Part Interaction Get To Role Type](#)
 - [Add Part Interaction Give To Role Type](#)
 - [Add Presentation To Role Type](#)
 - [Add Put Resource Into Collection To Role Type](#)
 - [Add Resource Attribute Maintenance Form To Role Type](#)
 - [Add Send Email To Role Type](#)
 - [Add Start A Role Instance And Introduce It To This Role Instance To Role Type](#)

Process Definition

- [Add Upload File To Role Type](#)
- [Add Web Service Call To Role Type](#)
- Maintain Conditions
 - [Add Simple Condition To Role Type](#)
 - [Add Compound Condition To Role Type](#)
 - [Assign Precondition To Activity In Role Type](#)
 - [Assign Postcondition To Activity In Role Type](#)
 - [Assign Always Condition To Role Type](#)
 - [Assign Terminating Condition To Role Type](#)
- Maintain Users
 - [Add User](#)
 - [Edit User](#)
 - [Delete Users](#)
 - [Start Role Instance And Assign User](#)
 - [Change User Of A Role Instance](#)
 - [Do Automated Role Instances](#)
 - [Update Role Instances To Current Type Definition](#)
 - [Tell Role Instance About User](#)
 - [Tell Role Instance About Role Instance](#)
 - [Tell Role Instance About Interaction Instance](#)
 - [Stop Role Instances And Deassign Users](#)
 - [Stop Terminated Role And Interaction Instances](#)
 - [Export Role Instances As Playwright](#)
- Maintain Interactions
 - [Start Interaction Instance](#)
 - [Introduce A Giver To An Interaction Instance](#)
 - [Introduce A Getter To An Interaction Instance](#)
 - [Add Give To Interaction Instance](#)
 - [Add Get To Interaction Instance](#)
 - [Stop Interaction Instances](#)
 - [Export Interaction Instances As Playwright](#)
- Maintain Frameworks
 - [Export World To A Specified Directory As Playwright Files](#)
 - [Import All Playwright Files In A Specified Directory](#)

5. Process Evolution

A RADRunner system can evolve naturally, to meet changing organisational needs.

There are the following methods of enabling this:

1. If your Roles include tasks for process definition as well as process enactment, users

- themselves will be able to change the process definitions as required and as permitted
2. Process definition is itself done via a Role, which at any time can modify the types and instances created by it or (where visible) other Roles
 3. A Role instance which has the necessary tasks can directly edit the XML text which defines types and instances within the system.

These approaches to process evolution can be used singly or in any combination to create a truly dynamic system, in which flexibility, coarse-grained control, and fine-grained control are simultaneously available.

Changes to process *types* are straightforward, but it is more complex to change the behaviour of running Role *instances*. In some cases this is possible by direct change to the instance objects from within RADRunner, but in others it is necessary *either* to export the Role instances to XML form for modification and re-import *or* to implement a "behaviour change pattern", by which new instances of an updated Role type are made and the contents of current Role instances transferred into them.

In the latter case (where the Role type in general is changed), this procedure is simplified by the task type, [Update Role Instances To Current Type Definition](#). This task allows the user to select running Role instances, and will then automatically update their behaviour to conform to the latest version of their Role type. Hence the option is available, after changing the type of a process, to bring as many process instances as required into line with the updated type - entirely without user intervention.

This feature makes RADRunner a uniquely powerful system for process evolution. However, it should be used with care, as the effects of such dynamic change can be confusing in the case where data previously in the Role body is removed from the new type. The automatic update will leave this data available in the updated Role instance(s), but the user may find (for instance) that this data is now inaccessible via maintenance screens, or no longer available as parameters for web service calls, database access, etc. Caution is advised!

The process definition task types required to implement process evolution in RADRunner are described [below](#).

6. Process Definition Task Types

6.1. Add Access Relational Database To Role Type

6.1.1. Description

Add an [Access Relational Database](#) task to a Role type.

6.1.2. Parameters

The RADRunner page allows for a maximum of 10 bind variables. More can be added if required by editing the Playwright XML.

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

URL of database

An entity attribute containing the URL of the database to be accessed. This is the JDBC URL without the standard *jdbc:* prefix. For example - *db2://woody/sales* or *oracle:thin:@144.212.123.24:1822* or *mysql://guthrie:3306/metrics*. If you are not sure what to put here, consult your database vendor.

Database username

An entity attribute containing the database username to be used

Database password

An entity attribute containing the database password to be used

SQL statement (command or query)

An entity attribute containing the SQL statement to be invoked (insert, update, delete, or select). The statement can contain *bind variables* - placeholders for parameters of the form *?*. These are indexed from left to right, starting at 1.

%1

An entity attribute containing bind variable 1 [optional]

%2

An entity attribute containing bind variable 2 [optional]

%3

An entity attribute containing bind variable 3 [optional]

%4

An entity attribute containing bind variable 4 [optional]

%5

An entity attribute containing bind variable 5 [optional]

%6

An entity attribute containing bind variable 6 [optional]

%7

An entity attribute containing bind variable 7 [optional]

%8

An entity attribute containing bind variable 8 [optional]

%9

An entity attribute containing bind variable 9 [optional]

%10

An entity attribute containing bind variable 10 [optional]

Entity to get from response

An entity into which the results will be placed [query only]

6.2. Add Automatable Start Role (And Introduce) To Role Type

6.2.1. Description

Add a [Start A Role Instance And Introduce It To This Role Instance](#) task to a Role type.

6.2.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Role type

Role type to be instantiated

Automated

Whether or not the new Role instance should be automated

User property name

Entity attribute containing the property name of the new Role's user [optional]

Interaction type for give

Interaction type for give [optional]

Activity to add give part interaction task to

Activity to add a new give part interaction task to - in current Role [required if and only if *Interaction type for give* specified]

Get part interaction

Get part interaction - in new Role [required if and only if *Interaction type for give* specified]

Interaction type for get

Interaction type for get [optional]

Give part interaction

Give part interaction - in new Role [required if and only if *Interaction type for get* specified]

Activity to add get part interaction task to

Activity to add a new get part interaction task to - in current Role [required if and only if *Interaction type for get* specified]

6.3. Add Compound Condition To Role Type

6.3.1. Description

Add a [compound condition](#) to a Role type - a logical statement formed by combining 1 or 2 existing conditions with a logical operator.

6.3.2. Parameters

Role type

The Role type to which the new object will be added

Property name

The "nickname" by which the new object will be known to its owner

Left-hand side

The simple condition to use as the left-hand side of the condition

Operator

A logical operator:

And

Nand

Not

Or

Xor

Right-hand side

If the operator is binary, the simple condition to use as the right-hand side of the condition

6.4. Add Computation To Role Type

6.4.1. Description

Add a [Computation](#) task to a Role type.

6.4.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Left-hand side attribute

The attribute whose value is to be used as the left-hand side of the operation, and which will be updated by the computation

Operation

The operation to be performed. This will be validated to check that it is appropriate for the left-hand side and right-hand side value types. The full operation list is:

And

Append

Decrement

DivideBy

FractionalPart

Increment

Lower

Minus

MinusDays

MinusHours

MinusMilliseconds

MinusMinutes

MinusSeconds

Process Definition

MinusWeeks
Modulo
MultiplyBy
Nand
Negate
Not
Now
Or
Plus
PlusDays
PlusHours
PlusMilliseconds
PlusMinutes
PlusSeconds
PlusWeeks
Prepend
Round
SetTo
SetToNull
Truncate
Upper
Xor

Right-hand side attribute

If the operation is binary, the attribute whose value is to be used as the right-hand side of the operation. The values shown here include 2 *pseudo-attributes* for each entity X: X.# (the total number of attributes and sub-entities belonging to the entity) and X.+ (the sum of all numeric attributes belonging to the entity and its sub-entities)

Right-hand side constant value

If the operation is binary, a constant value to be used as the right-hand side [as an alternative to an attribute value]

Automated

Whether or not the computation is to be carried out in the background, i.e., without manual instigation

6.5. Add Convert User To Entity To Role Type

6.5.1. Description

Add a [Convert User To Entity](#) task to a Role type.

6.5.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Entity to get from User(s)

The entity into which the user details will be placed

User

The user to convert

Automated

Whether or not the computation is to be carried out in the background, i.e., without manual instigation

6.6. Add Create Reference To User To Role Type

6.6.1. Description

Add a [Create Reference To User](#) task to a Role type.

6.6.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

User

An entity attribute containing the nickname of the user once a reference to it has been added to the Role

Id

An entity attribute containing the id of the user

6.7. Add Download File To Role Type

6.7.1. Description

Add a [Download File](#) task to a Role type.

6.7.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Prompt

An entity attribute containing a message on screen, explaining to the user what they are downloading and why

Attribute from which to take the downloaded filename

An entity attribute containing a URL (http:// or file://) for the file to be downloaded - if no protocol is specified, http:// is assumed

6.8. Add Entity To Role Type

6.8.1. Description

Add an instance of an entity type to a Role type as a top-level resource.

6.8.2. Parameters

Role type

The Role type to which the new object will be added

Property name

The "nickname" by which the new object will be known to its owner

Entity type

The entity type of which an instance will be added to the Role type

6.9. Add Entity Type Component

6.9.1. Description

Add a new:

- Entity type to the current Role
- Sub-entity to an existing entity type, or sub-entity of an entity type, in the current Role
- Entity attribute to an existing entity type, or sub-entity of an entity type, in the current Role.

6.9.2. Parameters

Object owner

The current Role, an entity type known to it, or a sub-entity of an entity type known to it

Property name

The "nickname" by which the new object will be known to its owner

Entity component type

Choose from:

Entity Type

Boolean Attribute

String Attribute

Float Attribute

Integer Attribute

Date Time Attribute

Now Attribute

Object supertype

The existing type to use as a template for the new type (*Entity Type* only)

Initial value

The initial value for the new component (*Attribute* only) [optional]

6.10. Add External Component To Role Type

6.10.1. Description

Process Definition

Add an [External Component](#) task to a Role type.

6.10.2. Parameters

The RADRunner page allows for a maximum of 10 parameters to the URL. More can be added if required by editing the Playwright XML.

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

URL of external component

An entity attribute containing the URL (http:// or file://) to be invoked - if no protocol is specified, http:// is assumed

Parameter name [1-10]

An entity attribute containing a parameter name

Parameter value [1-10]

An entity attribute containing a parameter value

6.11. Add Get Entity From URL To Role Type

6.11.1. Description

Add a [Get Entity From URL](#) task to a Role type.

6.11.2. Parameters

The RADRunner page allows for a maximum of 10 entry fields. More can be added if required by editing the Playwright XML.

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Entity to get from response

An entity into which to parse the XML elements and attributes returned from the URL

URL from which to get Entity

An entity attribute containing the URL (http:// or file://) to be invoked - if no protocol is specified, http:// is assumed

Parameter name [1-10]

An entity attribute containing a parameter name

Parameter value [1-10]

An entity attribute containing a parameter value

6.12. Add Get Entity From XSL To Role Type

6.12.1. Description

Add a [Get Entity From XSL](#) task to a Role type.

6.12.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Entity to use as destination

An entity into which to parse the XML elements and attributes returned by the specified XSL transformation of the Role resources

Resource attribute containing stylesheet URL

The XSL stylesheet defining the transformation required

6.13. Add Get Resource From Collection To Role Type

6.13.1. Description

Add a [Get Resource From Collection](#) task to a Role type.

6.13.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Source entity for collection

The entity whose sub-entities or attributes are used as the collection

Attribute to use as index into collection

An attribute containing an index into the collection - the attribute does not have to be of type Integer, but on execution of the task it must contain a value which can be used as an integer

Entity or attribute to use as destination

A Role resource into which the indexed collection element will be placed. If the destination resource is an entity, the sub-entities of the source entity are used as the collection. If the destination resource is an attribute, the attributes of the source entity are used as the collection.

Retain original element in collection?

If checked, the element will be left in the collection. Otherwise it will be removed.

Automated

Whether or not the selection is to be carried out in the background, i.e., based on the value of the index attribute, and without user involvement. If this flag is left unchecked, the task page allows the element to be chosen from the collection, whose items are displayed on screen in a selection box (if no selection is made,

the index attribute value will be used as in the automated case). *If the destination resource is an attribute*, the items in the source collection will also be attributes, in which case their values are displayed in the list. *If the destination resource is an entity*, the items in the source collection will also be entities, in which case the values shown are the values of a particular attribute in each item - the one named @, which corresponds to the *top-level text content of the item* when converted to [standard XML form](#). In either case, if no value exists, the index of the item will be shown enclosed in square brackets - for example, item 5 will be shown as [5].

6.14. Add Get To Interaction Instance

6.14.1. Description

Add a *get* (an outbound message send) to an interaction instance - the get is actually added to a *give* within the interaction instance, to provide an additional delivery for the message received via the give.

For a detailed explanation of interactions, see [Playwright semantics](#).

See also [Add Get To Interaction Type](#), an equivalent *define time* mechanism for interaction extension.

6.14.2. Parameters

Give property in interaction

The give to which the get will be added

Get property in interaction

The name of the new get

Resource to use as message

The name of the resource in the receiving Role

Retain original of message in getter?

Whether or not to leave any existing resource of the same name in the receiving Role - if so, a copy with a generated name will be added rather than overwriting the original

Messaging protocol when user is offline

How to send the message if the user is [offline](#) (its Roles are not operated by this system):

Email

Fax

JMS

MQ
Post
SOAP
TIA
Telephone

6.15. Add Get To Interaction Type

6.15.1. Description

Add a *get* (an outbound message send) to an interaction type - the get is actually added to a *give* within the interaction type, to provide an additional delivery for the message received via the give.

For a detailed explanation of interactions, see [Playwright semantics](#).

See also [Add Get To Interaction Instance](#), an equivalent *runtime* mechanism for interaction extension.

6.15.2. Parameters

Give property in interaction

The give to which the get will be added

Get property in interaction

The name of the new get

Resource to use as message

The name of the resource in the receiving Role

Retain original of message in getter?

Whether or not to leave any existing resource of the same name in the receiving Role - if so, a copy with a generated name will be added rather than overwriting the original

Messaging protocol when user is offline

How to send the message if the user is [offline](#) (its Roles are not operated by this system):

Email
Fax
JMS
MQ
Post
SOAP
TIA

Telephone

6.16. Add Give To Interaction Instance

6.16.1. Description

Add a *give* (an inbound message receipt) to an interaction instance, to provide an additional message delivered via the interaction.

For a detailed explanation of interactions, see [Playwright semantics](#).

See also [Add Give To Interaction Type](#), an equivalent *define time* mechanism for interaction extension.

6.16.2. Parameters

Interaction instance

The interaction instance to which the give will be added

Give property in interaction

The name of the new give

Resource to use as message

The name of the resource in the sending Role

Retain original of message in giver?

Whether or not to leave the resource in the sending Role - if not, it will be deleted from the sender

Messaging protocol when user is offline

How to receive the message if the user is [offline](#) (its Roles are not operated by this system):

Email

Fax

JMS

MQ

Post

SOAP

TIA

Telephone

6.17. Add Give To Interaction Type

6.17.1. Description

Process Definition

Add a *give* (an inbound message receipt) to an interaction type, to provide an additional message delivered via the interaction.

For a detailed explanation of interactions, see [Playwright semantics](#).

See also [Add Give To Interaction Instance](#), an equivalent *runtime* mechanism for interaction extension.

6.17.2. Parameters

Interaction type

The interaction type to which the give will be added

Give property in interaction

The name of the new give

Resource to use as message

The name of the resource in the sending Role

Retain original of message in giver?

Whether or not to leave the resource in the sending Role - if not, it will be deleted from the sender

Messaging protocol when user is offline

How to receive the message if the user is [offline](#) (its Roles are not operated by this system):

Email

Fax

JMS

MQ

Post

SOAP

TIA

Telephone

6.18. Add Interaction Type

6.18.1. Description

Add an interaction type to the current Role.

6.18.2. Parameters

Property name

The "nickname" by which the new object will be known to its owner

Object name

The internal name of the new object [optional - if left blank, the property name will be used]

Object supertype

The existing type to use as a template for the new type

Terminate after single execution?

If not, the interaction is enabled to repeat once all its gives and gets have completed - see the explanation of interactions under [Playwright semantics](#)

6.19. Add Introduce This Role Instance To Another Role Instance To Role Type

6.19.1. Description

Add an [Introduce This Role Instance To Another Role Instance](#) task to a Role type.

6.19.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Role instance

Role instance to which the introduction(s) will be made

Interaction type for give

Interaction type for give [optional]

Give part interaction

Give part interaction - in current Role [required if and only if *Interaction type for give* specified]

Activity to add get part interaction task to

Activity to add a new get part interaction task to - in specified Role instance [required if and only if *Interaction type for give* specified]

Interaction type for get

Interaction type for get [optional]

Activity to add give part interaction task to

Activity to add a new give part interaction task to - in specified Role instance
[required if and only if *Interaction type for get* specified]

Get part interaction

Get part interaction - in current Role [required if and only if *Interaction type for get* specified]

6.20. Add Manual Action To Role Type

6.20.1. Description

Add a [Manual Action](#) task to a Role type.

6.20.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

6.21. Add Part Interaction Get To Role Type

6.21.1. Description

Add a [Part Interaction Get](#) task to a Role type.

6.21.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Resource to use as message

The entity to use as a message, which can be nested at any level within the Role resources

6.22. Add Part Interaction Give To Role Type

6.22.1. Description

Add a [Part Interaction Give](#) task to a Role type.

6.22.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Resource to use as message

The entity to use as a message, which can be nested at any level within the Role resources

6.23. Add Presentation To Role Type

6.23.1. Description

Add a [Presentation](#) task to a Role type.

6.23.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Resource attribute containing stylesheet URL

An entity attribute containing the URL (<http://> or <file://>) of an XSL stylesheet - this will usually be a file of form *.xsl

6.24. Add Put Resource Into Collection To Role Type

6.24.1. Description

Add a [Put Resource Into Collection](#) task to a Role type.

6.24.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Destination entity for collection

The entity whose sub-entities or attributes are used as the collection

Attribute to use as index into collection

An attribute containing an index into the collection - the attribute does not have to be of type Integer, but on execution of the task it must contain a value which can be used as an integer. If the value is 0, the element is added at the start of the

collection and all existing elements have their index incremented by 1. If the value is between 1 and the current collection size, the existing element at that index is replaced. If the value is larger than the current collection size, the element is added at the end of the collection.

Entity or attribute to use as source

A Role resource from which the indexed collection element will be taken. If the source resource is an entity, the sub-entities of the source entity are used as the collection. If the source resource is an attribute, the attributes of the source entity are used as the collection.

6.25. Add Resource Attribute Maintenance Form To Role Type

6.25.1. Description

Add a [Resource Attribute Maintenance Form](#) task to a Role type.

6.25.2. Parameters

The RADRunner page allows for a maximum of 10 entry fields. More can be added if required by editing the Playwright XML.

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Prompt [1-10]

The label for an entry field [optional]

Resource Attribute [1-10]

An entity attribute to be updated [optional]

6.26. Add Role Type

6.26.1. Description

Add a new Role type to the current Role.

6.26.2. Parameters

Property name

The "nickname" by which the new object will be known to its owner

Object name

The internal name of the new object [optional - if left blank, the property name will be used]

Object supertype

The existing type to use as a template for the new type

Make Role type available to public?

Whether or not the new Role type should be [public](#)

6.27. Add Send Email To Role Type

6.27.1. Description

Add a [Send Email](#) task to a Role type.

6.27.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Email Recipient Address

Where to send the email to - entity attribute

Email Subject

The subject line of the email - entity attribute

Email Body Text

The body text of the email - entity attribute or entity (if entity, the [standard XML format](#) will be used)

6.28. Add Simple Condition To Role Type

6.28.1. Description

Add a [simple condition](#) to a Role type - a logical statement formed by combining 1 or 2 entity attributes and/or constants with an expression operator.

6.28.2. Parameters

Role type

The Role type to which the new object will be added

Property name

The "nickname" by which the new object will be known to its owner

Left-hand side

The attribute whose value is to be used as the left-hand side of the condition.
The values shown here include 2 pseudo-attributes for each entity X: X.# (the total number of attributes and sub-entities belonging to the entity) and X.+ (the sum of all numeric attributes belonging to the entity and its sub-entities)

Operator

An expression operator:

Equal To

Less Than

Less Than Or Equals

Greater Than

Greater Than Or Equals

Starts With

Contains

Ends With

Matches Regular Expression

In

Isa

Is Null

Not Null

And

Or

Xor

Nand

Right-hand side

If the operator is binary, the attribute whose value is to be used as the right-hand

side of the condition. The values shown here include 2 pseudo-attributes for each entity X: X.# (the total number of attributes and sub-entities belonging to the entity) and X.+ (the sum of all numeric attributes belonging to the entity and its sub-entities)

Right-hand side constant value

If the operation is binary, a constant value to use as the right-hand side of the condition [as an alternative to an attribute value]

6.29. Add Upload File To Role Type

6.29.1. Description

Add an [Upload File](#) task to a Role type.

6.29.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Prompt

Entity attribute containing the prompt

Uploaded filename

Entity attribute containing the name of the uploaded file on the server

Place URL to uploaded file in attribute

Entity attribute in which a URL to the uploaded file will be placed

6.30. Add User Definition Tasks To Role Type

6.30.1. Description

Add User Definition tasks to a Role type:

- [Add User](#)

- [Change User Of A Role Instance](#)
- [Delete Users](#).

This gives instances of the Role type the ability to extend and maintain the user base.

See also [Edit User](#); this task is included by default in all new Role types, along with the automated tasks [Do Automated Role Instances](#) and [Stop Terminated Role And Interaction Instances](#).

To transfer knowledge of a user to another Role (so as to delegate the ability to change their details, start and stop their Role instances, and so on), it is necessary to use [Convert User To Entity](#) to store the id of the user in a Role resource, transfer the resource to another Role via an interaction, then use [Create Reference To User](#) to recreate a reference to the user in the receiving Role.

6.30.2. Parameters

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

6.31. Add Web Service Call To Role Type

6.31.1. Description

Add a [Web Service Call](#) task to a Role type.

6.31.2. Parameters

The RADRunner page allows for a maximum of 5 parameters and 5 return values. More can be added if required by editing the Playwright XML.

Object owner

The Role type or activity to which the task will be added

Property name

The name to be assigned to the task

New activity name for task (if owner is not itself an activity)

If the owner is a Role type, a new activity will be created within it to contain the task - specify its name here

Property index

The numeric index (starting at 1) of the task within the owner Role type or activity

Description of the Task

A description of the task

Location of WSDL file for web service [network path/URL]

A URL or network path to the WSDL file for the web service

Web service operation name

The operation to be invoked from the web service

Parameter name [1-5]

A parameter to the web service

Resource to use as parameter [1-5]

An entity attribute or entity containing a parameter value - if an entity is specified, the [standard XML form](#) will be used

Constant value to use as parameter [1-5]

A constant parameter value [used if no attribute containing the parameter value is specified]

Return value name [1-5]

A return value from the web service

Resource for return value [1-5]

An entity attribute or entity into which the return value will be placed - if an entity is specified, the return value is assumed to be in [standard XML form](#) and parsed accordingly into attributes and sub-entities

6.32. Assign Always Condition To Role Type

6.32.1. Description

Assign an [always condition](#) to a Role type.

6.32.2. Parameters

Condition

The condition to assign - the Role type to use must be its owner, so there is no need to specify it separately

6.33. Assign Postcondition To Activity In Role Type

6.33.1. Description

Assign a [postcondition](#) to an activity in a Role type.

6.33.2. Parameters

Role type node

The activity, or group of activities, to which the condition will be assigned

Condition

The condition to assign

6.34. Assign Precondition To Activity In Role Type

6.34.1. Description

Assign a [precondition](#) to an activity in a Role type.

6.34.2. Parameters

Role type node

The activity, or group of activities, to which the condition will be assigned

Condition

The condition to assign

6.35. Assign Terminating Condition To Role Type

6.35.1. Description

Assign a [terminating condition](#) to a Role type.

6.35.2. Parameters

Condition

The condition to assign - the Role type to use must be its owner, so there is no need to specify it separately

6.36. Delete Activities And Tasks From Role Types

6.36.1. Description

Delete a number of activities and/or tasks from Role types known to the current Role.

6.36.2. Parameters

Activities and/or tasks

The activities and/or tasks to be deleted

6.37. Delete Entities From Role Types

6.37.1. Description

Delete top-level resources from Role types known to the current Role. These will be [entities rather than entity attributes](#).

6.37.2. Parameters

Entities

The top-level entity resources to delete

6.38. Delete Entity Type Components

6.38.1. Description

Delete entity types and/or entity type sub-entities and/or entity type attributes known to the current Role.

If entity types are specified which have been added to a Role type, the task will fail, since entities within Role types are considered to be instances of their entity type.

6.38.2. Parameters

Entity type components

The entity types and/or attributes to delete

6.39. Delete Interaction Types

6.39.1. Description

Delete interaction types known to the current Role.

If the specified types have active instances, the task will fail.

6.39.2. Parameters

Interaction types

The interaction types to delete

6.40. Delete Role Types

6.40.1. Description

Delete Role types known to the current Role.

If the specified types have active instances, the task will fail.

6.40.2. Parameters

Role types

The Role types to delete.

6.41. Do Automated Role Instances

6.41.1. Description

Perform background automation on any [automated Role instances](#) known to the current Role.

This task is placed by default in every new Role type. It can be deleted if not required.

6.41.2. Parameters

None.

6.42. Export Entity Types As Playwright

6.42.1. Description

Export specified entity types known to the current Role as [Playwright XML](#), with each object placed in a separate file. The files are placed in the [system out directory](#).

6.42.2. Parameters

Entity types

The objects to be exported as Playwright XML

6.43. Export Interaction Instances As Playwright

6.43.1. Description

Process Definition

Export specified interaction instances known to the current Role as [Playwright XML](#), with each object placed in a separate file. The files are placed in the [system out directory](#).

6.43.2. Parameters

Interaction instances

The objects to be exported as Playwright XML

6.44. Export Interaction Types As Playwright

6.44.1. Description

Export specified interaction types known to the current Role as [Playwright XML](#), with each object placed in a separate file. The files are placed in the [system out directory](#).

6.44.2. Parameters

Interaction types

The objects to be exported as Playwright XML

6.45. Export Role Instances As Playwright

6.45.1. Description

Export specified Role instances known to the current Role as [Playwright XML](#), with each object placed in a separate file. The files are placed in the [system out directory](#).

6.45.2. Parameters

Role instances

The objects to be exported as Playwright XML

6.46. Export Role Type Resources

6.46.1. Description

Export the resources of specified Role types known to the current Role as [standard XML](#), with each Role's resources placed in a separate file. The files are placed in the [system out directory](#).

This task is particularly of use when defining XSL transformations based upon Role resources, for use in tasks such as [Presentation](#) and [Get Entity From XSL](#).

6.46.2. Parameters

Role types

The Role types whose resources are to be exported

6.47. Export Role Types As Playwright

6.47.1. Description

Export specified Role types known to the current Role as [Playwright XML](#), with each object placed in a separate file. The files are placed in the [system out directory](#).

6.47.2. Parameters

Role types

The objects to be exported as Playwright XML

6.48. Export World To A Specified Directory As Playwright Files

6.48.1. Description

Export the entire Playwright world to [Playwright XML](#) files in a specified directory on the server LAN.

For details on how to export individual processes, see [exporting processes as XML](#).

6.48.2. Parameters

Directory name

A directory on the server LAN to which Playwright files representing the entire Playwright world will be exported

6.49. Import All Playwright Files In A Specified Directory

6.49.1. Description

Import to the Playwright world all [Playwright XML](#) files found in a specified directory on the server LAN.

Use with care, since this will cause any object with the same id as an imported object to be overwritten. For details on how to avoid this, see [importing processes from XML](#).

6.49.2. Parameters

Directory name

A directory on the server LAN containing Playwright files to upload and import

6.50. Import Entity From XML File

6.50.1. Description

Import an entity type or sub-entity from a file containing [standard XML](#).

See also [Import Entity Type From Playwright](#).

6.50.2. Parameters

Object owner

The current Role or an entity type known to it

Property name

The "nickname" by which the new object will be known to its owner

File name

The client-side file containing [standard XML text for an entity](#) - this file will be uploaded, and its content used to define the new object

6.51. Import Entity Type From Playwright

6.51.1. Description

Import an entity type from a file containing [Playwright XML](#).

See also [Import Entity Type From XML File](#).

6.51.2. Parameters

Property name

The "nickname" by which the new object will be known to its owner

Content to upload to file

If desired, Playwright text can be pasted into this field - if non-empty, this will be placed in the server-side file and used to define the new object [optional]

File name

The client-side file containing Playwright text - this will be uploaded, and (if no content is specified separately) its content used to define the new object

6.52. Import Interaction Type From Playwright

6.52.1. Description

Import an interaction type from a file containing [Playwright XML](#).

6.52.2. Parameters

Property name

The "nickname" by which the new object will be known to its owner

Content to upload to file

If desired, Playwright text can be pasted into this field - if non-empty, this will be placed in the server-side file and used to define the new object [optional]

File name

The client-side file containing Playwright text - this will be uploaded, and (if no content is specified separately) its content used to define the new object

6.53. Import Role Type From Playwright

6.53.1. Description

Import a Role type from a file containing [Playwright XML](#).

6.53.2. Parameters

Property name

The "nickname" by which the new object will be known to its owner

Content to upload to file

If desired, Playwright text can be pasted into this field - if non-empty, this will be placed in the server-side file and used to define the new object [optional]

File name

The client-side file containing Playwright text - this will be uploaded, and (if no content is specified separately) its content used to define the new object

6.54. Introduce A Getter To An Interaction Instance

6.54.1. Description

Bind a [Part Interaction Get](#) in a Role instance to a *get* in an interaction instance.

For a detailed explanation of interactions, see [Playwright semantics](#).

6.54.2. Parameters

Get property in interaction

The get to be bound

Get part interaction

The part interaction to be bound

6.55. Introduce A Giver To An Interaction Instance

6.55.1. Description

Bind a [Part Interaction Give](#) in a Role instance to a *give* in an interaction instance.

For a detailed explanation of interactions, see [Playwright semantics](#).

6.55.2. Parameters

Give property in interaction

The give to be bound

Give part interaction

The part interaction to be bound

6.56. Start Role Instance And Assign User

6.56.1. Description

Start a new Role instance and assign a user to it.

6.56.2. Parameters

Property name

The "nickname" by which the new object will be known to its owner

Object name

The internal name of the new object [optional - if left blank, the property name will be used]

Role type

The Role type to instantiate

Automated

Whether or not the new Role instance is enabled for [automation](#) by the current Role

User

The user to assign to the new Role instance

6.57. Start Interaction Instance

6.57.1. Description

Create a new interaction instance.

6.57.2. Parameters

Property name

The "nickname" by which the new object will be known to its owner

Object name

The internal name of the new object [optional - if left blank, the property name will be used]

Interaction type

The interaction type to instantiate

Terminate after single execution?

This setting over-rides [the equivalent setting in the interaction type](#), for use when a new interaction instance is created *manually* (rather than *automatically* via one of the tasks [Start A Role Instance And Introduce It To This Role Instance](#) or [Introduce This Role Instance To Another Role Instance](#))

6.58. Stop Interaction Instances

6.58.1. Description

Terminate specified interaction instances.

6.58.2. Parameters

Interaction instances

The instances to terminate

6.59. Stop Role Instances And Deassign Users

6.59.1. Description

Terminate specified Role instances, and deassign their users.

This task is placed by default in every new Role type. It can be deleted if not required.

6.59.2. Parameters

Role instances

The instances to terminate

6.60. Stop Terminated Role And Interaction Instances

6.60.1. Description

Stop any [terminated Role instances](#) known to the current Role.

This task is placed by default in every new Role type. It can be deleted if not required.

6.60.2. Parameters

None.

6.61. Update Role Instances To Current Type Definition

6.61.1. Description

Update specified Role instances to the current definition of their Role type.

For each instance, a corresponding new instance is made of its Role type, and the resources of the old instance transferred into it. The old instance is then terminated, and its id given to the new instance - which preserves any connections previously inherent in running processes (knowledge of the Role instance, bindings to interactions, etc).

If the new instance contains data not available in the old instance, this will be left at its default values. Conversely, if the old instance contains data not available in the new instance, this will be left intact, but it is unlikely that the user will now be able to make much use of this data (e.g., in maintenance screens, as parameters to web service calls, for database access, etc) - however, it is available via resource snapshots and exports if required.

Note that if a particular resource in the updated Role type also exists in the old instances, but with some elements added, these new elements will *not* appear in the updated instances. It is therefore necessary to cater for this eventuality, either by using tasks which create the elements as and when required, or by creating new resources rather than re-using old ones and allowing for transfer of any existing data into them (which is safer, since an audit trail is then kept of data from the previous incarnation of the Role instance).

This task is placed by default in every new Role type. It can be deleted if not required.

6.61.2. Parameters

Role instances

The instances to update