

# XML Import And Export

## 1. Overview

Both the data in a process and a process itself can be imported and exported to XML - data as standard XML, processes as Playwright XML.

## 2. Data

Playwright entity types can be imported from, and role instance resources exported to, *standard XML*. Both Playwright entities and attributes equate to XML elements. For simplicity, there is no direct mapping either to XML element content where sub-elements exist, or to XML attributes. However, support for full XML syntax is provided via a [notational convention](#).

Entity types are imported via the task [Import Entity From XML File](#).

Role instance resources are exported via a [Role resources snapshot](#), which saves the current resources in an XML file on the server, shows the path name of the file, and displays its contents.

These features are useful particularly when defining XSL transformations of the Role space (for use in RADRunner screens, to generate data such as web service call parameters, or create text for use in an email message) and to provide compatibility with any [XML schemas](#).

### 2.1. XML Element Content and XML Attributes

To enable support for XML of arbitrary complexity, an attribute can have the special names:

**@**

This corresponds to the text content of the owner element, and can be used to generate an element which contains sub-elements (Playwright sub-entities or attributes) yet also has text content

**@xxx**

This generates an XML attribute named xxx of the owner element

The Playwright approach to data definition preserves full compatibility with standard XML, yet provides a simpler and more intuitive means of structuring information.

In particular, in standard XML the rationale for placing content in an *attribute* rather than an *element* is poorly understood, and can lead to confusion. XML attributes are data *qualifiers*, a difficult concept for many people to grasp. Playwright dispenses with the concept altogether, while retaining the ability to export and import standard XML as required.

## 2.2. XML Schema Support

A Playwright entity type can be *created directly from an XML schema*. An XML schema itself is only a framework, without sufficient detail to implement a business process, so it is necessary first to customise the data appropriately.

This can be done using any XML editing tool (XMLSpy, for instance):

1. Use a command such as *Generate sample XML file* to create a piece of XML text conforming to a specified schema
2. Set initial values and modify the contents of collections in this XML text as appropriate for your business process
3. [Import the XML text into RADRunner](#) to create an entity type - based directly on the original schema, but customized appropriately for your particular business process

The entity type can then be used as the basis for XSL transformations, web service calls, calculations, and so on within a business process.

## 3. Processes

Objects in a RADRunner process can be exported to and imported from the XML dialect Playwright.

It is important to understand that each Playwright object has its own *id*. Objects ids are used both to ensure uniqueness within the world, and to allow objects to reference each other. Hence, care should be taken to maintain these relationships when importing and exporting processes from XML.

For more information about the XML form of processes, see [Playwright Syntax](#).

### 3.1. Export

Once you have created a process, you may wish to export the definitions as *Playwright XML* for backup, editing by hand, analysis, etc. You can do this either:

- In 3 steps, via the tasks [Export Role Types As Playwright](#), [Export Entity Types As Playwright](#), and [Export Interaction Types As Playwright](#) - in each task, select multiple types to export them all at once

## XML Import And Export

*This approach will export types only*

or

- In 1 step, via the task [Export World To A Specified Directory As Playwright Files](#) - if you wish, you can then pick out the XML files which correspond to your new types.  
*This approach will export users and instances as well as types*

It is also possible to export Role and interaction instances:

- The parent of Role instances can export them via the task [Export Role Instances As Playwright](#).
- The user of a Role instance can export it via a [Role snapshot](#), which saves the current Playwright for the Role instance in an XML file on the server, shows the path name of the file, and displays its contents.
- The parent of interaction instances can export them via the task [Export Interaction Instances As Playwright](#).

### 3.2. Import

Exported Playwright files can be re-imported into any RADRunner system at any time. This is particularly useful for backup purposes: see [Backup](#) for a detailed explanation of the recommended backup procedure.

New files of Playwright XML can also be imported into RADRunner. Each file should contain one top-level object: user, Role type or instance, entity type or instance, interaction type or instance. Typically, one does not create such files from scratch, but sets up the objects initially via RADRunner, then exports them for editing by hand in an XML editor (or a standard text editor) before importing the revised versions into RADRunner.

*To make minor changes to running processes* in this way, carry out the import via the task:

- [Import All Playwright Files In A Specified Directory](#)

Both instances and types can be imported in this way. If there is a syntax error in an imported file, RADRunner will flag it and abort the entire import to maintain consistency.

*Use with care*, since this task will cause any object with the same id as an imported object to be overwritten. If only types are to be imported, the danger can be avoided by importing each object separately as below.

*To add new types to the world*, use the tasks:

- [Import Role Type From Playwright](#)
- [Import Entity Type From Playwright](#)
- [Import Interaction Type From Playwright](#)

In each task, select one Playwright file at a time to import it. If an object with the same id as

hat in the file already exists in the world, RADRunner will flag it and abort the import.