Configuring UAX - UAXDIRECTORY - project files

On this page:

Target audience:

CAST Administrators

Summary: This page describes how to configure the **.UAX files**, **.UAXDIRECTORY file** and the **xxx_project.UAX file** for use with the Universal Importer in **Use Case 2: Import an additional technology not supported by CAST**.

.UAX files

The .UAX that contain the results of your external analysis must conform to a pre-defined format that can be understood by the Universal Importer. Objects and links that you would like to import are described as **instances** using **categories** inherited by **type**: the types and categories are all defined in the **XXXMetaModel.xml file** (a file that is part of the **Universal Analyzer language package** you created - see Creating or modifying a custom extension using the CAST Universal Analyzer framework).

The instances do not have to respect any **order,** i.e. links can be created before objects.

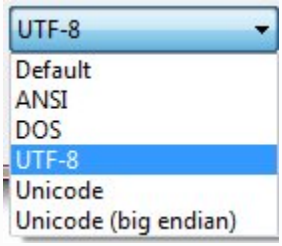
Note that:

- it is compulsory to define values in your Language Pack for the "**identification.name**" and "**identification.fullName**" properties for any object to be imported - otherwise the objects will not be available to view (i.e. in CAST Enlighten).
- you **should not** define parent relationships (via "parentLink" type) when using "externalXXX" properties. This will lead to inconsistent results. It is also not possible to define Foreign Keys, Constraints, or Common Keys in this way.

.UAX files must be encoded in UTF-8 format as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Therefore please ensure that you save the files in your text editor using the correct format:



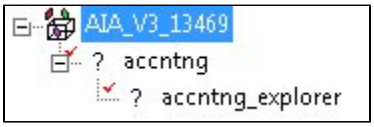
Example .UAX file for Graphtalk

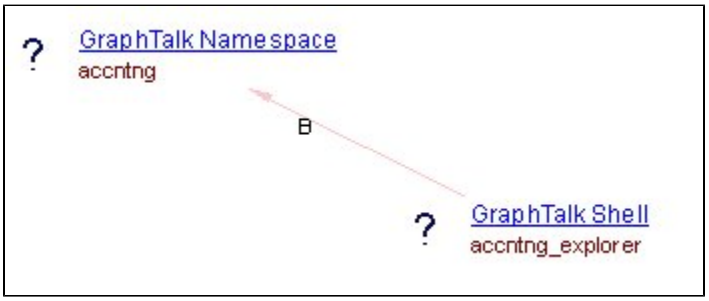
Let's create a simple .UAX file (called **accntng.uax**). It contains instances that define the following:

- a "GTpackage" namespace object belonging to the "AIA_V3" project
- a link between the "GTpackage" namespace object and the "AIA_V3" project
- a "GTshell" object belonging belonging to the "AIA_V3" project
- a link between the "GTpackage" namespace object and the "GTshell" object

```
<?xml version="1.0" encoding="UTF-8"?>
<instances>

<!-- First we declare the GTpackage object instance and define its "id", "name", "fullName" and a "persistent
guid":
- id: must be unique in the source code delivery (i.e. the set of .uax files analyzed in the same Universal
Importer Analysis Unit
- name: can be non-unique and no string constraints exist
- fullName: try to limit the length because a long name can impact analysis performance and the "size" of the
results
- guid: must be identical throughout all source code deliveries (V1, V2, V3 etc.)

Note that the 'guid' MUST be unique throughout all projects.
If no 'guid' is specified, then the analyzer will use the value of the 'id' attribute
of the 'instance' element, to construct the 'guid'.
-->

        <instance id="WPDi60000KY0M" instanceOf="GTpackage">
                <identification name="accntng" fullName="Package - $accntng"/>
                <persistent guid="package.accntng"/>
        </instance>

<!-- We then declare the parent project as well as its external/internal status:
- we ensure that the GTpackage namespace object is part of the "AIA_V3" project
- projectRelationKind=0 ensures the object is considered by CAST AIP as an object
internal to the application (snapshots are run on internal objects only)
- projectRelationKind=1 ensures the object is considered by CAST AIP as an object
external to the application, i.e. created by a third party (snapshots are run on
internal objects only, so external objects are ignored)
-->

        <instance id="" instanceOf="isInProjectLink">
                <isInProjectLink projectRelationKind="0"/>
                <link callee="AIA_V3" caller="WPDi60000KY0M"/>
        </instance>

<!-- we then ensure that there is  a belongs to (parentLink) from the GTpackage namespace object to the root
project:
- root objects always belong ot the root project %ProjectRoot%
-->

        <instance id="" instanceOf="parentLink">
                <link callee="%ProjectRoot%" caller="WPDi60000KY0M"/>
        </instance>

<!-- we now describe the GTshell object and define its "ID", "name", "fullName" and an "object guid" -->

        <instance id="sh3SQ00023Y0O" instanceOf="GTshell">
                <identification name="accntng_explorer" fullName="Shell - $accntng:accntng_explorer"/>
                <persistent guid="package.accntng.shell.accntng_explorer"/>
        </instance>

<!-- we then ensure that the GTshell object is part of the "AIA_V3" project:
- projectRelationKind=1 ensures the object is considered by CAST AIP as an object
external to the application, i.e. created by a third party (snapshots are run on internal objects only, so
external objects are ignored)
-->

        <instance id="" instanceOf="isInProjectLink">
                <isInProjectLink projectRelationKind="1"/>
                <link callee="AIA_V3" caller="sh3SQ00023Y0O"/>
        </instance>

<!-- we then ensure that there is a belongs to (parentLink) from the GTshell object to the GTpackage namespace
object -->

        <instance id="" instanceOf="parentLink">
                <link callee="WPDi60000KY0M" caller="sh3SQ00023Y0O"/>
        </instance>
</instances>
```

When importing this .UAX file into the CAST Analysis Service using the **Universal Importer**, the following results will appear in CAST Enlighten:





If you want to create links between other objects which have been created in the CAST Analysis Service via a standard CAST analyzer (i.e. not created/imported with Universal Importer), you must use the properties "externalCaller" and/or "externalCallee" instead of "caller/callee" in the "link" element. The properties "externalXXX" use the object ID (as found in the "Keys.IdKey" column in the Analysis Service). Please see Advanced - Links with external objects stored in the Analysis Service for more information about this process.

CodeAndCommentsChecksum and CodeOnlyChecksum

If you define checksum values for an instance using the **CodeAndCommentsChecksum** and **CodeOnlyChecksum** attributes in the **<checksum>** tag as shown below, then you must ensure that the values are decimal ONLY. If you use a hexidecimal value (for example), the value will not be interpreted correctly by the Universal Importer.

```
<instance id="WPDi60000KY0M" instanceOf="GTpackage">
        <identification name="accntng" fullName="Package - $accntng"/>
        <persistent guid="package.accntng"/>
        <checksum CodeAndCommentsChecksum="456987" CodeOnlyChecksum="687456"/>
</instance>
```

projectRelationKind

It is possible to declare an object instance as either "internal" to the application or "external" to the application via the "**projectRelationKind**" attribute. This determines whether an object is included in a snapshot or not:

Attribute	Description
projectRelationKind =0	Ensures the object is considered by CAST AIP as an object **internal** to the application (snapshots are run on internal objects only)
projectRelationKind =1	Ensures the object is considered by CAST AIP as an object **external** to the application, i.e. created by a third party (snapshots are run on internal objects only, so external objects are ignored)

Splitting instance definitions

It is possible to split the definition of an instance into small pieces if necessary by defining more than one <instance> tag for a unique instance. For example, the following is valid:

```
<instance id="C:\test.php" instanceOf="sourceFile">
   <identification name="test.php"/>
   <identification fullName="C:\eiffel\test\php\test.php"/>
</instance>
<!-- ... -->
<instance id="C:\test.php" instanceOf="sourceFile">
   <bookmark source="C:\test.php"/>
</instance>
```

However the following is invalid (because the properties of the category "identification" are defined in two separate code pieces):

```
<instance id="C:\test.php" instanceOf="sourceFile">
   <identification name="test.php"/>
</instance>
<!-- ... -->
<instance id="C:\test.php" instanceOf="sourceFile">
   <identification fullName="C:\eiffel\test\php\test.php"/>
</instance>
```

.UAXDIRECTORY and xxx_project.UAX files

These files are **not required** when you have installed a dedicated CAST Delivery Manager Tool plugin which will generate the .UAXDIRECTORY/xxx_project.UAX files for you.

If you do NOT have a **CAST Delivery Manager Tool plugin** (i.e. you are using the **Generic Extraction** option in the CAST Delivery Manager Tool), then you will need to additionally supply a **.UAXDIRECTORY** file and a **xxx_project.UAX** file together with your standard .UAX files (all in the same folder). Without the files, the import will not function. You can configure your external anayzer to generate these two files for you, or you can generate them manually.

.UAXDIRECTORY file

This file is similar to a "manifest" file and simply defines the Universal Importer project and the location of the various .UAX files (containing the object/link definition information). You are free to choose whatever name for the file you like, but you must:

- use the .UAXDIRECTORY extension
- save the file in UTF-8 format
- include a UTF-8 encoding declaration in the file
- save the file the same folder as all other .UAX files.

An example .UAXDIRECTORY file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<UAXFiles>
 <UAXOption name="technicalVersion" value="1.1"/>
 <UAXFile path="CAST_project.uax" name="AIA_V3" type="GraphtalkProject">
          <UAXFile path="accntng.uax" name="" type="GTpackage">
          </UAXFile>
 </UAXFile>
</UAXFiles>
```

Explanation of required entries:

Entry	Description
<UAXOption name=" technicalVersion" value="1.1"/>	This entry is not required, but can be useful to document the version of the external analyzer that has generated the .UAX files.
<UAXFile path="CAST_project. uax" name="AIA_V3" type=" GraphtalkProject"> </UAXFile>	This entry defines the name of the **xxx_project.UAX** file (explained in a section below): • path="CAST_project.uax" - defines the path to the **xxx_project.UAX** file • name="AIA_V3" - defines the name of the project in the **xxx_project.UAX** file • type="GraphtalkProject" - defines the type of the project inside the **xxx_project.UAX** file - the type must also be defined in the Universal Analyzer framework xxxMetaModel.xml file (only used for discovery in the CAST Delivery Manager Tool).
<UAXFile path="accntng.uax" name="" type="GTpackage"> </UAXFile>	You must define one entry like this for EACH .UAX file you have created (or has been automatically created by the external analyzer): • path="accntng" - defines the path to the **.UAX** file • name="" - leave blank • type="" - describes the type of object configured in the .UAX file

xxx_project.UAX

This file defines the project type that is discovered by the CAST Delivery Manager Tool. You are free to choose whatever name for the file you like (CAST recommends using the **xxx_project.UAX** naming convention), but you must:

- use the .UAX extension
- save the file in UTF-8 format
- include a UTF-8 encoding declaration in the file
- save the file the same folder as all other .UAX/.UAXDIRECTORY files.

An example **xxx_project.UAX** file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<instances>
<instance id="AIA_V3" instanceOf="GraphtalkProject">
    <identification name="AIA_V3" fullName="AIA_V3"/>
</instance>
<instance id="" instanceOf="projectDependencyLink">
    <link callee="%ProjectRoot%" caller="AIA_V3"/>
    <projectDependencyLink dependencyKind="0"/>
</instance>
<instance id="" instanceOf="isInProjectLink">
    <isInProjectLink projectRelationKind="0"/>
    <link callee="AIA_V3" caller="AIA_V3"/>
</instance>
</instances>
```

Explanation of required entries:

Entry	Description
`<instance id="AIA_V3" instanceOf="GraphtalkProject"><identification name="AIA_V3" fullName="AIA_V3" /></instance>`	Defines the **ID** of the project (AIA_V3), which must be unique and an **instanceOf** parameter referring to the parent element as defined in the Universal Analyzer framework xxxMetaModel.xml file. The **instanceOf** parameter must be entered into the CAST Delivery Manager Tool - see Appendix - Importing an additional technology not supported by CAST AIP. A name and fullname must also be defined. CAST recommends using the same as was used for the ID parameter.
`<instance id="" instanceOf="projectDependencyLink"><link callee="%ProjectRoot%" caller="AIA_V3"/><projectDependencyLink dependencyKind="0"/></instance>`	Defines root of the import. The project (AIA_V3) is a child of this item.
`<instance id="" instanceOf="isInProjectLink"><isInProjectLink projectRelationKind="0"/><link callee="AIA_V3" caller="AIA_V3"/></instance>`	Ensure that callee and caller are defined with the project ID.

Object GUIDs

CAST uses an internal object ID naming system that is progressively being introduced. This ID naming system effectively means that each object that is saved in the CAST Analysis Service is given a unique OBJECT_GUID. This value is then used for a wide variety of different features and processes.

When creating your .UAX files, you can give each instance (i.e. object you want to import and save in the CAST Analysis Service) a specific OBJECT_GUID. For example, the following code example would create an OBJECT_GUID ("thefileguid") for the object "test.php" (this example is for import from file):

```
<instance id="C:\test.php" instanceOf="sourceFile">
   <bookmark source="C:\test.php"/>
   <persistent guid="thefileguid"/>
   <identification name="test.php" fullName="C:\eiffel\test\php\test.php"/>
</instance>
```

To summarize:

- For import from file: use **<persistent guid="<value>"/>** when you want to create a specific OBJECT_GUID.
- If you do not want to generate a specific OBJECT_GUID, then omitting the above values will force the Universal Importer to **automatically generate** an OBJECT_GUID. Currently, the OBJECT_GUID is generated from the object's **instance ID**. For example, the following code would automatically create an OBJECT_GUID ("**C:\test.php**") for the object "test.php":

```
<instance id="C:\test.php" instanceOf="sourceFile">
   <bookmark source="C:\test.php"/>
   <identification name="test.php" fullName="C:\eiffel\test\php\test.php"/>
</instance>
```

- If you are generating a specific OBJECT_GUID, you MUST make sure that the values are **unique** within the CAST Analysis Service.

Please note, however, that in future versions, this automatic OBJECT_GUID naming system will be modified and the **instance id** will NO LONGER be used to generate the OBJECT_GUID where specific naming values have been omitted.

If you have been relying on the automatic OBJECT_GUID generation process (i.e. not specifying any specific values) for legacy objects and you want to retain the same OBJECT_GUID in future CAST versions, you should **NOW** start to modify your importation processes to ensure that a specific OBJECT_GUID is created that matches the object's instance ID (use the instructions above).

The impact of not doing so on legacy objects in future CAST versions would be that a new OBJECT_GUID would be automatically created that is different to the OBJECT_GUID automatically created in previous versions of CAST. As such this would be recorded as an object change, falsely skewing the results displayed in the CAST Dashboard.