Creating a new custom component

Summary: This section describes how to create a new custom component. Please note that it is not possible to create custom components to interact with the CAST Management Studio installation packs: PMC, PMC_LOCAL, CORE_PMC and PMC_CENTRAL.

Define the .pck file for the component

CAST recommends naming the **.pck** file with the same name as defined in the "**PackName**" attribute in the .pck file and with a CUST_ prefix. So for example:

- CUST_DIAG_PROC.pck
- PackName = "CUST_DIAG_PROC".

Use the following template to define your .pck file:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<Package PackName = "CUST_xxxx" Type = "SPECIFIC" SupportedServer=" " Version = "x.x.x.x" Display=" "
Description=" " DatabaseKind=" ">
   <Include>
      <PackName>SYSTEM</PackName>
      <Version>1.0.0</Version>
   </Include>
   <Exclude>
      <!-Do not use for Custom component->
   </Exclude>
   <Install>
      <!-Add step for installation->
   </Install>
   <Update>
      <!-Add step for update->
   </Update>
   <Refresh>
      <!-Add step for refresh->
   </Refresh>
   <Remove>
      <!-Do not use for Custom component->
   </Remove>
</Package>
```

The following table explains the attributes that need to be defined in the root "package" tag:

Name	Description	Information	Status
PackName	Component Name	Must be unique (used as a reference key). CAST recommends prefixing your PackName with CUST_	Mandatory
Version	Component version	<VerMaj>.<VerMin>.<Rev>.<Build>	Mandatory
Display	Display string	<used only in log file>	Optional

Description	Description	<used only in log file>	Optional
DatabaseKind	Describes the target CAST schema	Can be any of the following: • KB_LOCAL (Analysis Service) • KB_CENTRAL (Dashboard Service) • KB_AAD (Measurement Service) • KB_PMC (Management Service)	Mandatory

Note that "**PackName**" must not use a name that is already used by the CAST AIP schema installation process. This includes (but is not an exhaustive list):

- ADG_CENTRAL
- ADG_FULL_CENTRAL
- ADG_LOCAL
- ADG_LOCAL_APPW
- ADG_LOCAL_AngularJS
- ADG_LOCAL_HTML5
- ADG_LOCAL_JQuery
- ADG_LOCAL_NodeJS
- ADG_LOCAL_WbsLinker
- ADG_METRIC_TREE
- ADG_METRIC_TREE_ANGULARJS
- ADG_METRIC_TREE_HTML5
- ADG_METRIC_TREE_JQuery
- ADG_METRIC_TREE_NODEJS
- ADG_METRIC_TREE_PLUGIN_JEE
- ADG_METRIC_TREE_PLUGIN_SQLSCRIPT
- ADG_METRIC_TREE_WbsLinker
- AED_CENTRAL
- APM_CENTRAL
- APM_LOCAL
- APPMARQ_CENTRAL
- APPMARQ_DEMOGRAPHICS
- APPW
- APPWCOMPAT
- AUTOVIEW
- BASE_LOCAL
- CAL
- CALIBRATION_RULES
- CAL_SHARED
- CAT
- CORE_APPW
- CORE_PMC
- CORE_VIEWER
- COST
- COSTCENTRAL
- CSV
- CSV_CENTRAL
- DBGREPLAYAMT
- DBGSAVELOCAL
- DELTATOOL_ANALYSIS
- DELTATOOL_SNAPSHOT
- DIAGCORE
- DSS
- DSSAPP_LOCAL
- DSS_CENTRAL
- DSS_LOCAL
- EFP
- EFP_CENTRAL
- KMS2_LOCAL
- KMS_LOCAL
- MANAGE_PLUGINS_ASSESSMENTMODEL
- MANAGE_PLUGINS_CENTRAL
- MANAGE_PLUGINS_LOCAL
- MANAGE_PLUGINS_MNGT
- METRICS
- OBJSET
- PLUGIN_AngularJS_LOCAL
- PLUGIN_HTML5_LOCAL
- PLUGIN_JEE_LOCAL
- PLUGIN_JEE_SET
- PLUGIN_JQuery_LOCAL
- PLUGIN_NodeJS_LOCAL
- PLUGIN_SQLSCRIPT_APPW
- PLUGIN_SQLSCRIPT_DSS
- PLUGIN_SQLSCRIPT_LOCAL
- PLUGIN_SQLSCRIPT_PMC
- PLUGIN_WbsLinker_LOCAL

- PMC
- PMC_CENTRAL
- PMC_LOCAL
- PMC_MAIN
- RDBMSTOOLS
- STAT
- STD_ADG_VIEWER
- SYSTEM
- TCC

Define installation and update operations in the .pck file

INSTALL tag

The commands defined within the INSTALL tags are executed once during the initial installation of a new component. Use this zone only to install critical objects that must not be dropped (e.g.: tables for persistent data storage). For working tables, reference tables, procedures and functions use the REFRESH tag instead.

REFRESH tag

Commands defined within the REFRESH tags are executed each time the component is installed, upgraded or reinstalled. CAST Server Manager processes the script and drops existing objects before creation if they exist on the server (data in tables will be lost). Use this zone for:

- all procedural objects and temporary or working tables
- data that can be restored (reference tables)

IMPORTANT: always try to use the REFRESH tag where possible

UPDATE tag

Commands defined within the UPDATE tags are executed during a migration. Use the migration process only when necessary:

- Addition of new tables (only tables that will also be added in the INSTALL zone)
- Structural change for a table defined in the INSTALL zone
- Data update (for data not defined in the REFRESH zone)
- drop for obsolete objects

Define an operation step in the .pck file

Operation steps are added to the **.pck** in the zones specified above (INSTALL, REFRESH and UPDATE). A step can be defined between the <Step xxxx> and <\Step> tags. "xxxx" defines all the information required for the execution of this step. Steps are executed in the order of their definition in the **.pck** file.

Name	Description	Information	Zone INSERT (I) /UPDATE (U) /REFRESH (R)	Status

Type	Operation type	Authorized values:
		• **PROC** and **TABLE** - used to process **SQL script files** for creation of either PROCEDURES or TABLES in the target AIP schema. For example:
		`<Step Type="PROC" Option="[value]" File="[path\to\some_file.sql]">`
		`<Step Type="TABLE" Option="[value]" File="[path\to\some_file.sql]">`
		• **DATA** - performs a **scoped import** into **existing tables** of the content of XMLTODB data .xml file, with the scope defined in .xml model file (i.e a DML placeholder). For example:
			I / U / R	Mandatory

```
<Step Type="DATA" File="[path\to\some_data_file.xml]" Model="
[path\to\some_model_file.xml]" Scope="[scope]">

//Format of some_data_file.xml

<?xml version="1.0" encoding="UTF-8" ?>
<DATA>
        <TEST_TABLE>
                <OBJECT_ID>1</OBJECT_ID>
                <OBJECT_DESCRIPTION>Description 1</OBJECT_DESCRIPTION>
        </TEST_TABLE>
        <TEST_TABLE>
                <OBJECT_ID>2</OBJECT_ID>
                <OBJECT_DESCRIPTION>Description 2 </OBJECT_DESCRIPTION>
        </TEST_TABLE>

</DATA>

//Format of some_model_file.xml

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
noNamespaceSchemaLocation="..\CAST_XMLDB_Format.xsd" >

        <table name="TEST_TABLE"  description="Give description of
table" >
                <column name="OBJECT_ID" type="int" nullable="false"
description="Column description" />
                <column name="OBJECT_DESCRIPTION" type="string"
nullable="true" length="250" description="Column description"  />
                <index name="IDX_TEST_TABLE" type="index" >
                        <column name="OBJECT_ID" sort="asc" />
                </index>
        </table>

  <scope name="TESTSCOPE">
                <scopetable name="TEST_TABLE"/>
  </scope>

</config>
```

- **XML_MODEL** - (mostly used internally by CAST) performs a **scoped import** into **new tables** of the content of XMLTODB data .xml file with a scope (i.e. a DDL placeholder). For example:

```
<Step Type="XML_MODEL" File="[path\to\some_data_file.xml]" Scope="
[scope]">

//Format of some_data_file.xml

<?xml version="1.0" encoding="UTF-8" ?>
<DATA>
        <TEST_TABLE>
                <OBJECT_ID>1</OBJECT_ID>
                <OBJECT_DESCRIPTION>Description 1</OBJECT_DESCRIPTION>
        </TEST_TABLE>
        <TEST_TABLE>
                <OBJECT_ID>2</OBJECT_ID>
                <OBJECT_DESCRIPTION>Description 2 </OBJECT_DESCRIPTION>
        </TEST_TABLE>

</DATA>
```

Option	Parameter handling	Numeric value, one of the following: • Option = "0" (Normal flow) • Option = "4" (OVERWRITE - force drop for existing objects in UPDATE scripts) • Option = "8" (ADD_GRANT) • Option = "16" (ADD_VOLATILE) • Option = "32" (NOT_LOGGED) • Option = "64" (SKIP_EXEC) • Option = "128" (NO_RETRY) • Option = "256" (PLAY_INDEX_ONLY - install index only) • Option = "512" (PLAY_TABLE_ONLY - install table only)	U	Optional
File	File name	Path for file: • Relative path file with .pck directory as reference. • SQL script file or XML file (depending on Type)	I / U / R	Mandatory
ToVersion	Destination version	New version for the component	U	Mandatory
FromVersion	Initial version	Initial version for the component	U	Mandatory / Optional
ServerVersion	Supported version	Provides a means to configure different scripts for Microsoft SQLServer. Warning: not to be used to specify scripts for different RDBMS (Oracle / Microsoft SQLServer / CSS)	I / U / R	Optional
Model	XML Model File	Path file for XML model file to use in data transfer (relative path)	I / U / R	Mandatory for type DATA
DataWay	Way of transfer ToDb / ToXML	Use this to perform an export or import of data into/out of the database (with datatransfer)		Optional. Default = ToDB
Scope	import xml parameter	Scope defined in Model XML		Optional

Example with comments

Component name: CUST_COMPONENT

First release of component

- One table for user parameter storage: **fileTable1.sql**
- Default data for user parameter table: **fileXMLData1.xml / ModelXML.xml**
- One working table: **fileTable2.sql**
- One procedure: **fileProc2.sql**
- One reference table: **fileTable3.sql**
- Data for reference table: **fileXMLData2.xml / ModelXML.xml**

.pck file

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<PackagePackName = "CUST_COMPONENT" Type = "SPECIFIC" Version ="1.0.0.1" SupportedServer = "ALL" Display=""
Description="" DatabaseKind=" ">
   <Include>
      <PackName>SYSTEM</PackName>
      <Version>1.0.0</Version>
   </Include>
   <Exclude>
   </Exclude>
   <Install>
      <Step Type="TABLE" File ="fileTable1.sql></Step>
      <Step Type="DATA" File="fileXMLData1.xml" Model = "ModelXML.xml"></step>
   </Install>
   <Update>
   </Update>
   <Refresh>
      <Step Type="TABLE" File=" fileTable2.sql"></step>
      <Step Type="TABLE" File=" fileTable3.sql"></step>
      <Step Type="PROC" File=" fileProc2.sql"></step>
      <Step Type="DATA" File="fileXMLData2.xml" Model = "ModelXML.xml"></step>
   </Refresh>
   <Remove>
   </Remove>
</Package>
```

Initial execution will run:

- fileTable1.sql
- Transfer data fileXMLData1.xml with ModelXML.xml
- fileTable2.sql
- fileTable3.sql

- Transfer data fileXMLData2.xml using ModelXML.xml

Second execution using reinstall will run

- fileTable2.sql (drop existing objects defined in the file)
- fileTable3.sql (drop existing objects defined in the file)
- fileProc2.sql (drop existing objects defined in the file)
- Transfer data fileXMLData2.xml using ModelXML.xml

Second release of component

- You add a new table for storage: fileNewTable1.sql
- You update data in table (fileTable1.sql): (fileXMLData1.xml modified)
- You update data in a reference table (change in fileXMLData2.xml)
- You add new working table and table: FileNewTableProc.sql

.pck file

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<Package PackName = "CUST_COMPONENT "Type = "SPECIFIC" Version ="1.1.0.1" SupportedServer = "ALL" Display=""
Description="" DatabaseKind=" ">
   <Include>
      <PackName>SYSTEM</PackName>
     <Version>1.0.0</Version>
   </Include>
   <Exclude>
   </Exclude>
   <Install>
      <Step Type="TABLE" File ="fileTable1.sql></Step>
      <Step Type="TABLE" File ="fileNewTable1.sql"></Step>
      <Step Type="DATA" File="fileXMLData1.xml" Model = "ModelXML.xml"></step>
   </Install>
   <Update>
      <Step Type="TABLE" File ="fileNewTable1.sql" FromVersion="1.0.0.1" ToVersion=1.1.0.1></Step>
      <Step Type='DATA" File="fileXMLData1.xml" Model = "ModelXML.xml" FromVersion="1.0.0.1" ToVersion=1.1.0.1><
/step>
   </Update>
   <Refresh>
      <Step Type="TABLE" File=" fileTable2.sql"></step>
      <Step Type="TABLE" File=" fileTable3.sql"></step>
      <Step Type="PROC" File=" fileProc2.sql"></step>
      <Step Type="PROC" File=" FileNewTableProc.sql"></step>
      <Step Type="DATA" File="fileXMLData2.xml" Model = "ModelXML.xml"></step>
   </Refresh>
   <Remove>
  </Remove>
</Package>
```

First execution on previously installed database (version 1.0.0.1):

- fileNewTable1.sql (existing objects in this script will not be dropped)
- Transfer data fileXMLData1.xml with ModelXML.xml
- fileTable2.sql (drop existing objects defined in the file)
- fileTable3.sql (drop existing objects defined in the file)
- fileProc2.sql (drop existing objets defined in the file)
- FileNewTableProc.sql
- Transfer data fileXMLData2.xml using ModelXML.xml

Specific information for UPDATE

The migration process for user components may be complex. For example, suppose you have the following case:

- **CAST AIP Version**: V1 > V2 > V3 > V4 > V5
- **Component Version**: V1 > V2 > V3

Your component might be impacted by a modification made between each version of CAST. Upgrades should be written from one version to the next without using cumulative updates. A migration from V1 to V3 will not be V1 to V2 + V2 to V3 but a direct migration from V1 to V3. For this example you will need to write two migration processes: V1 to V3 and V2 to V3. In the **.pck** file this update is added to the UPDATE zone as follows:

```
<Step Type="TABLE" File ="updateV1V3.sql" FromVersion="V1" ToVersion=V3></Step>
<Step Type="TABLE" File ="updateV2V3.sql" FromVersion="V2" ToVersion=V3></Step>
```

In this case:

- Update V1 > V3 will be executed only if the current version is V1
- Update V2 > V3 will be executed only if the current version is V2

Further information

CAST Server Manager automatically performs the following tasks:

- grants on tables & procedures to public
- indexes & tables are installed in the tablespace/device specified in CAST Server Manager. **Warning: do not set the tablespace in your scripts**
- Install objects in the selected schema. **Do not specify a schema in your scripts**
- Drop existing objects for all objects defined in a REFRESH script. For an UPDATE script, an existing object will not be dropped except if Mode=" 4" is set for the script. An object is dropped if it already exists with same name and type (this is true for Tables, Procedures, Functions, Views and Indexes).
- A drop object in your scripts will be skipped if the object is not present in the database.

Writing SQL script for CAST Server Manager

Separators need to be added between each statement:

- **/** (PostgreSQL/CSS)

Finish the script file with a separator on a separate line.