Semarchy

xDI DEV

H Deployment

- Web Service

Semarchy

xDI DEV

H Deployment

H1 - Concepts & Use in xDI

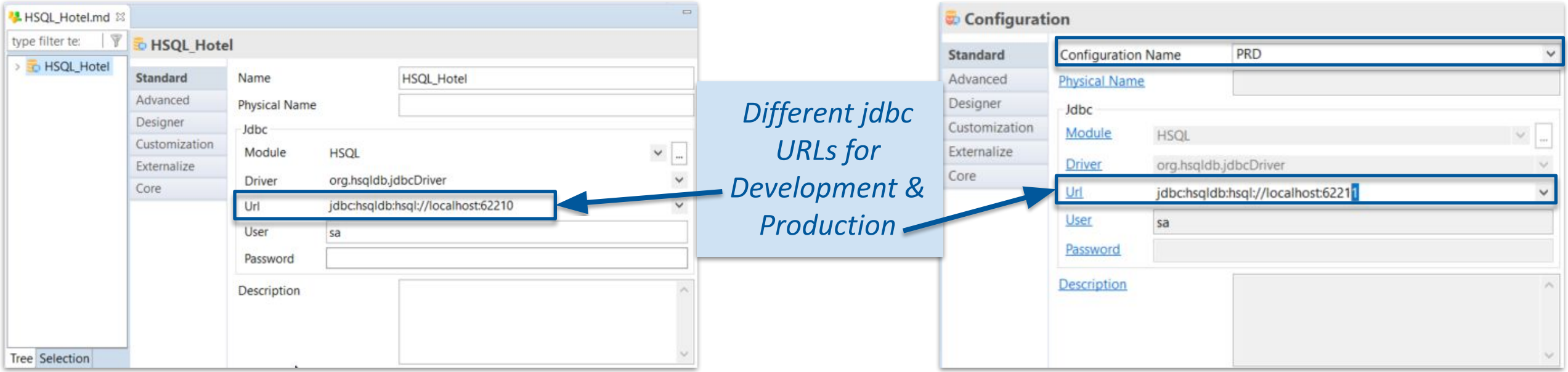
Semarchy

Configuration concept

The configuration

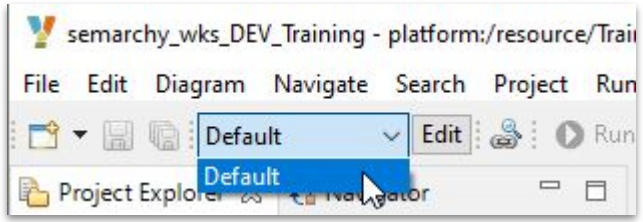
Configurations enable to overload the property of every Metadata for a specific environment (or context)

- Ex : when there are different connection URLs for Development & Production

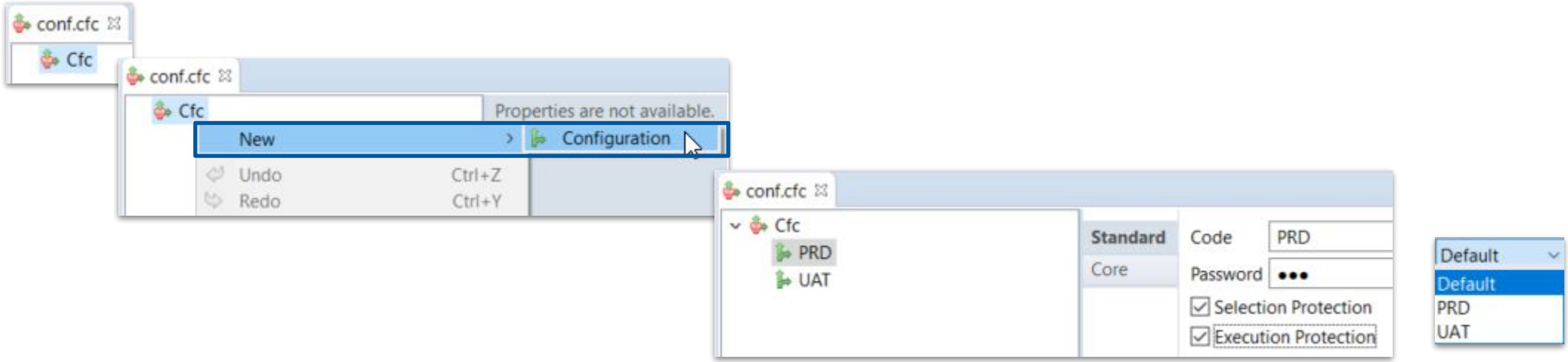


Creating and using configurations in xDI Designer

In xDI Designer, list of Configurations can be found in a combobox in the top left part of the screen :

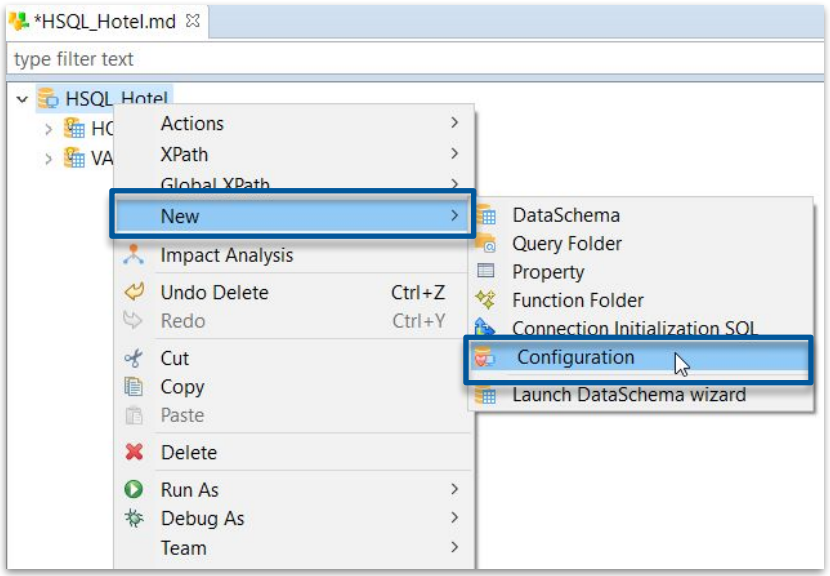


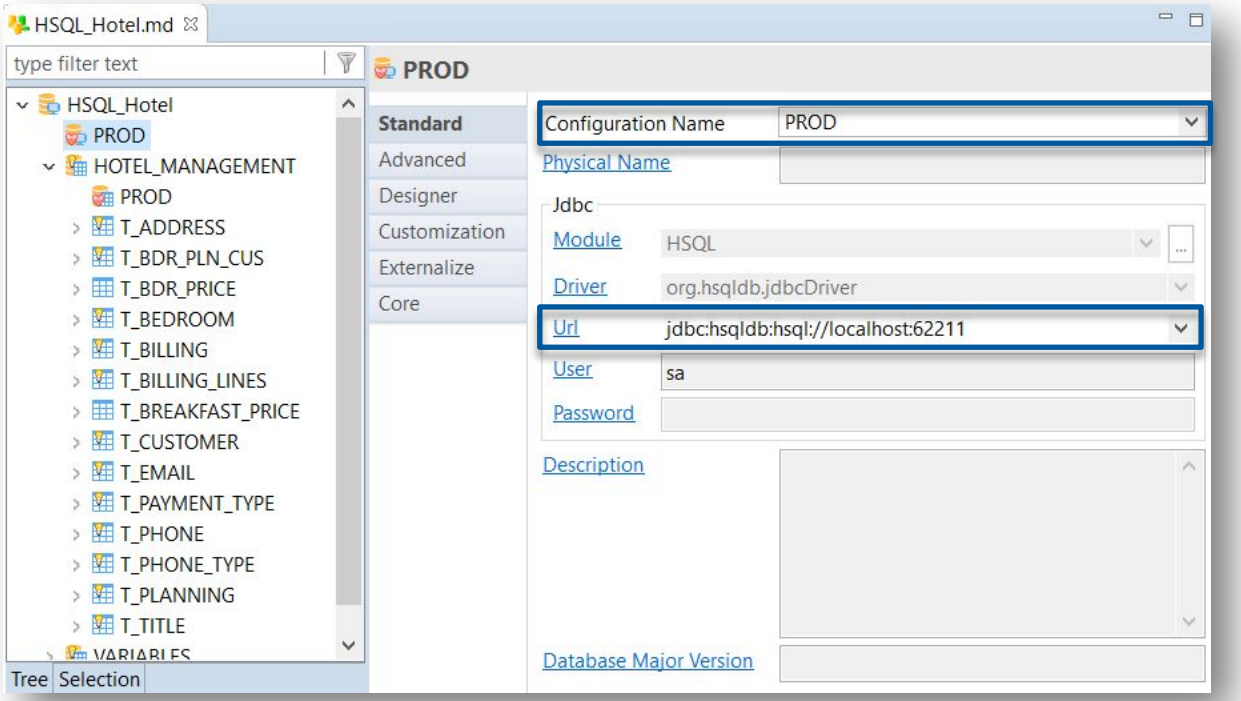
- Click on “Edit” button to open the configuration editor & create a new configuration



Using a configuration

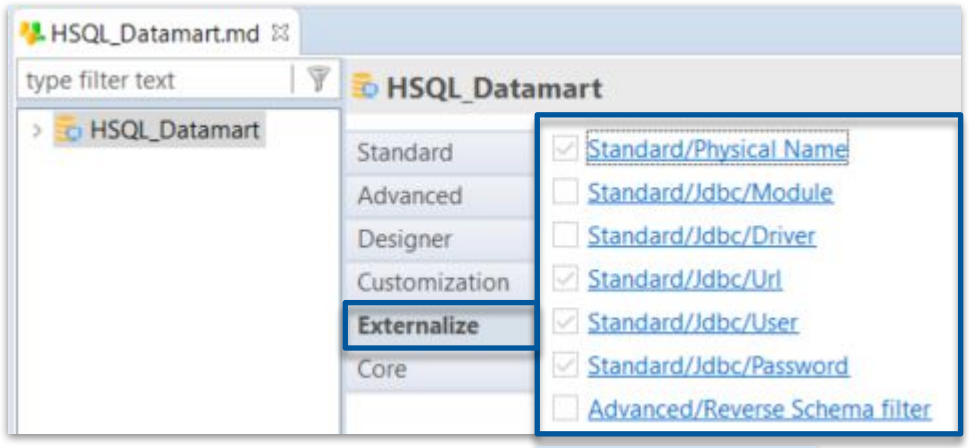
Inside every Metadatas, each node can be overloaded for a configuration:



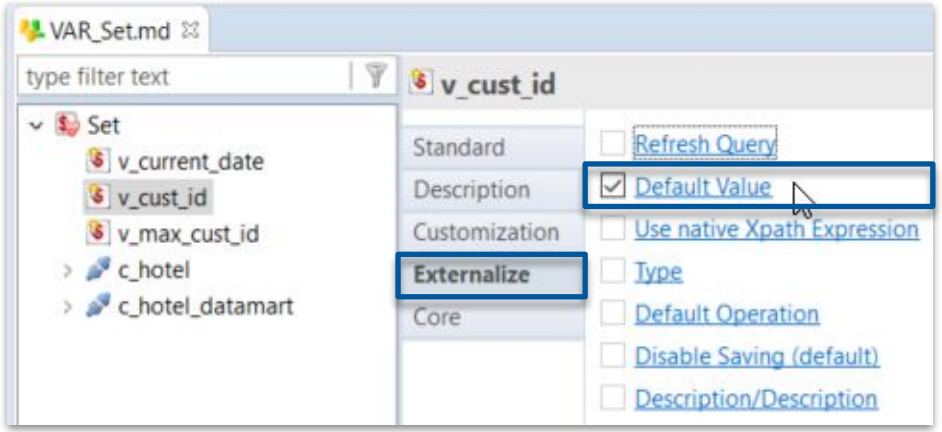


Externalize Metadata properties

Metadata properties are defined “To be externalized” inside packages *(concept explained in the next slides)* in the Externalize view



Default setting



Possible to change the default setting if it’s required to change the value of a property in another environment

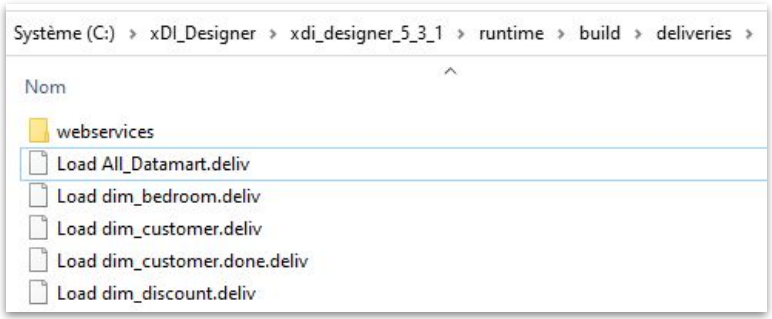

Semarchy
Deliveries & packages

Delivery principle

The object that will eventually be executed is a « deliverable » (.deliv file)

- **Already "configured"** in a defined configuration

- Each execution done by the Designer produces a deliverable which is stored in a Runtime subfolder:
  - runtime/build/deliveries

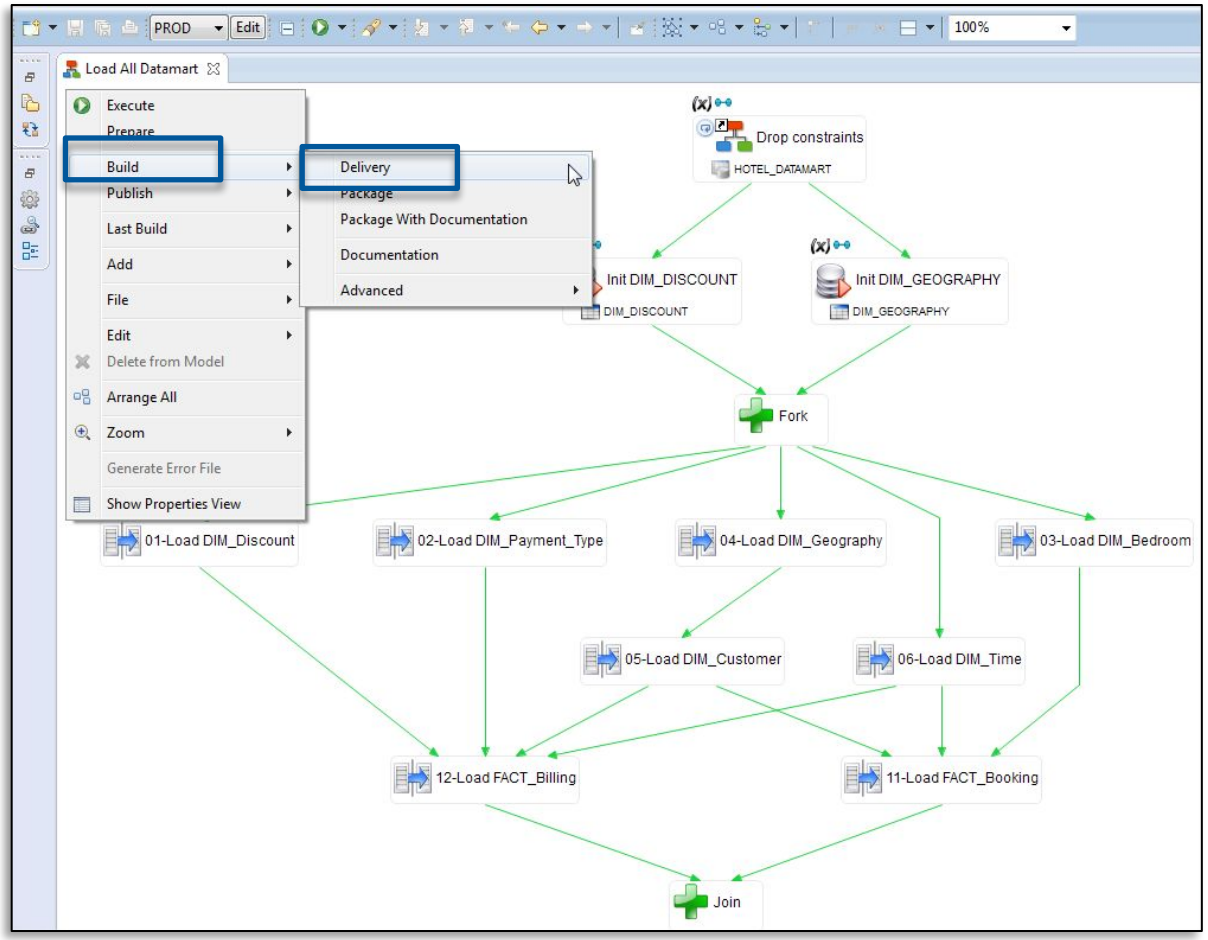


We could say that "*the easiest way to launch a production is to generate the deliverable with the right configuration and then deploy it on the production runtime*"

- But, **it's not recommended** to
  - Configure the Metadatas in the xDI Designer for Production environment
  - Use Production Runtimes
- Indeed, this can generate security breach and dangerous usage

Generate a deliverable

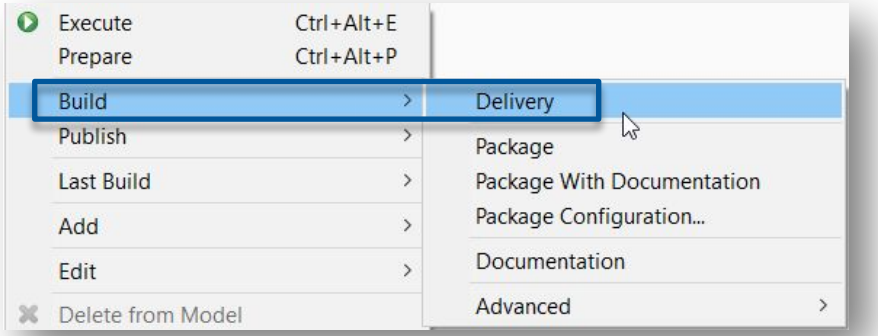
To build a Delivery, right click on a process/mapping and choose “**Build**”/ “**Delivery**” menu

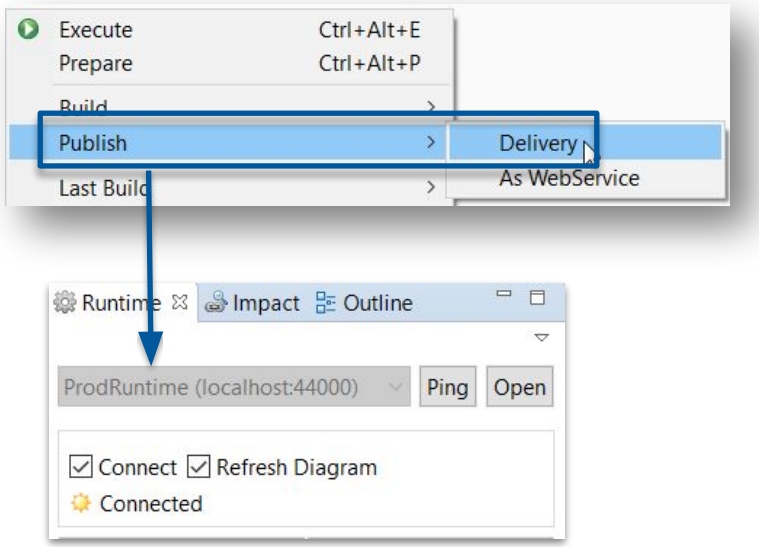


H-Deployment

Build or Publish Delivery

- **Build Delivery** will generate a delivery (.deliv) of the current mapping/process on the **local runtime** of the Designer
- **Publish Delivery** will generate a delivery (.deliv) of the current mapping/process on the **selected runtime** (on which the Designer is connected)

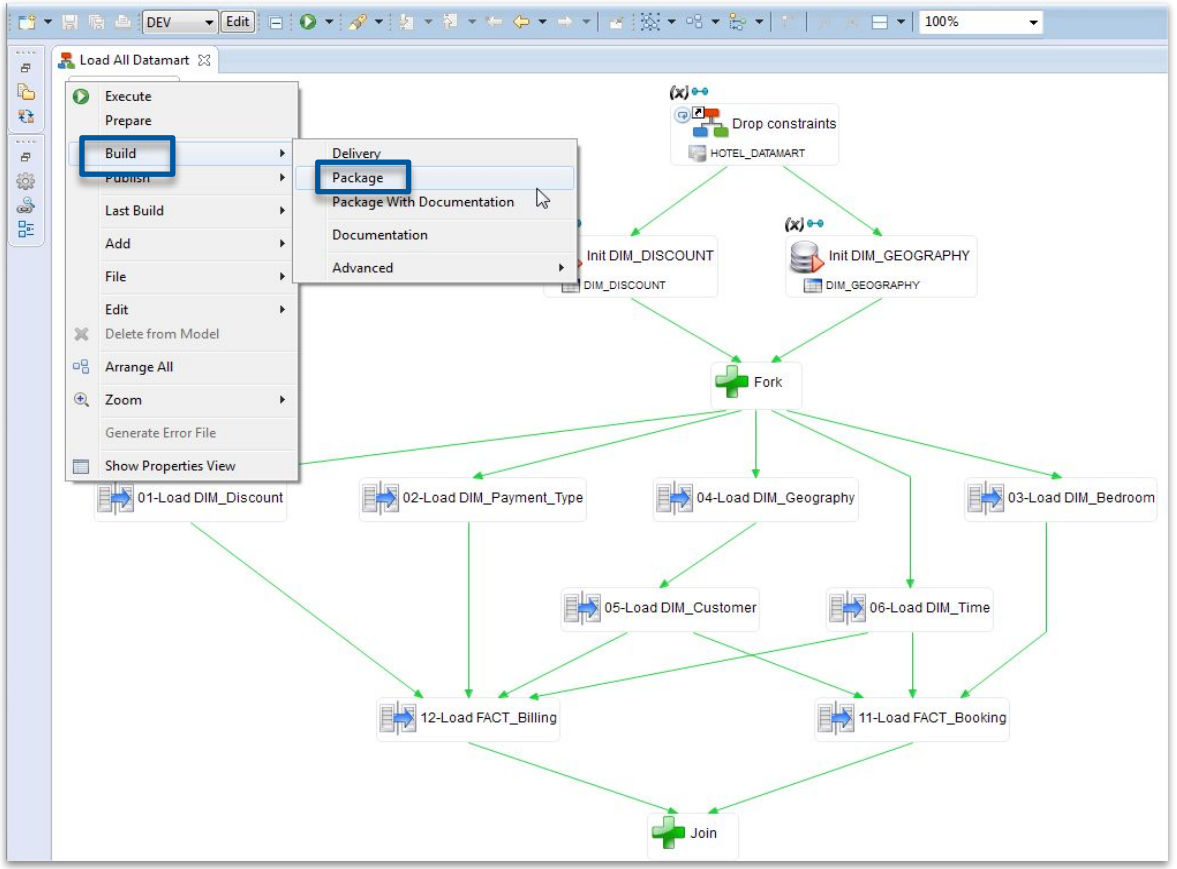




The Package

Used to allow the production or operating teams to configure Process or Mapping for other environments (change properties of Metadata):

- The solution is to **Build Package**

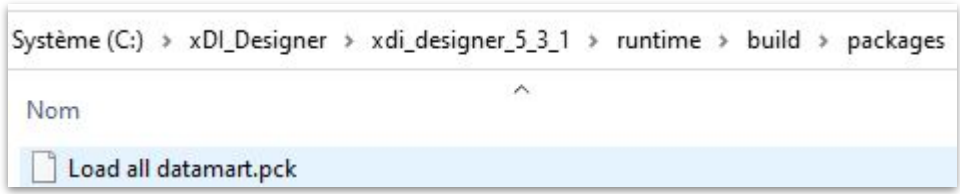


H-Deployment

Using packages

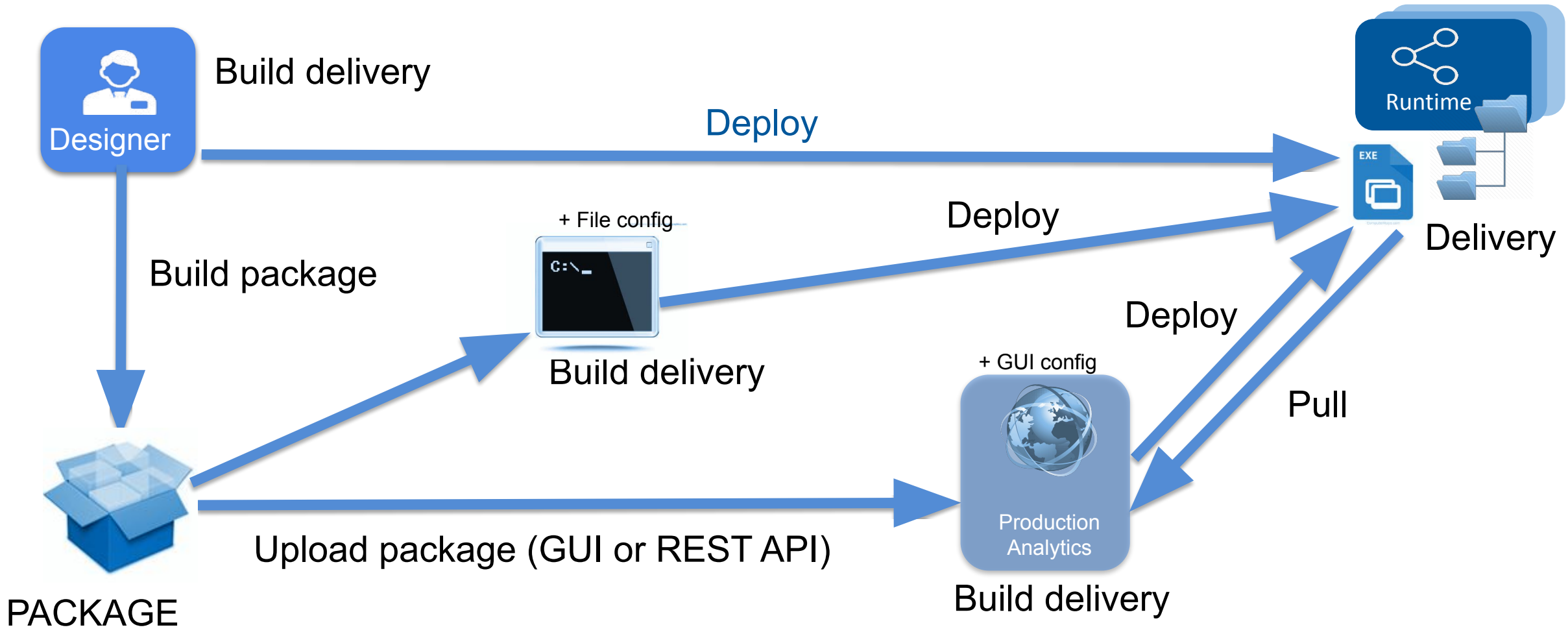
The packages are generated into a Runtime subfolder

- runtime/build/packages



- Packages are RAR archives
- They include everything required to generate a deliverable
- Two ways to use a package :
  - Command line
  - Stambia Analytics

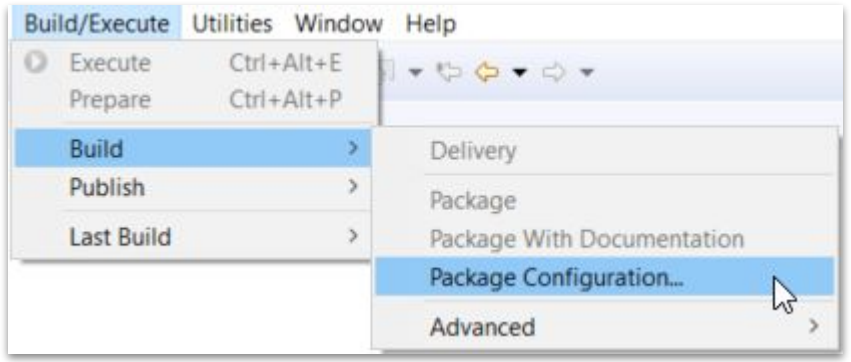
Different ways to deploy Delivery

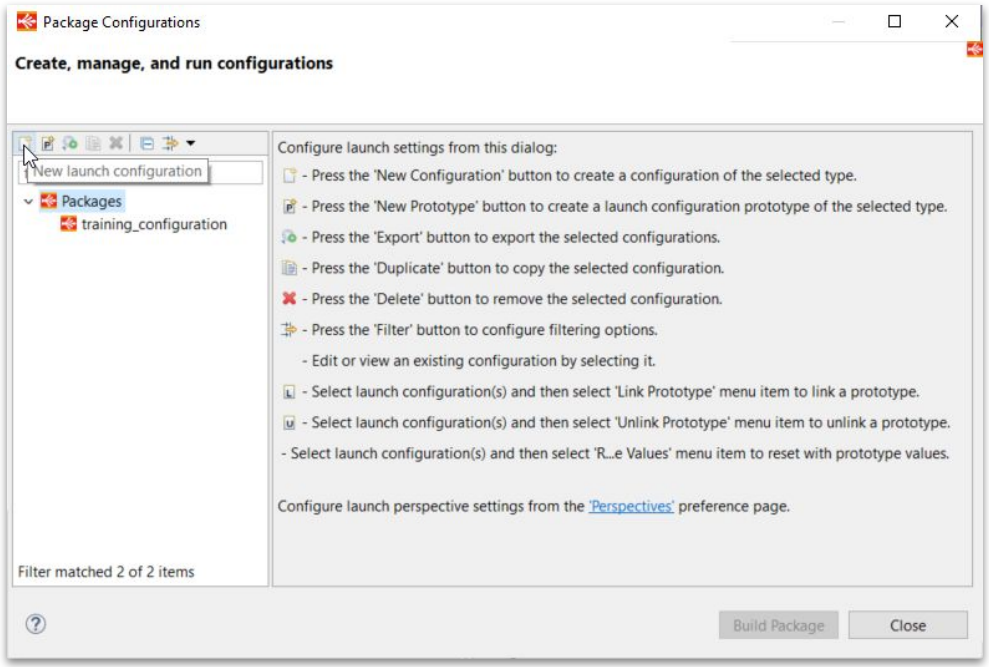


Build Package Configuration

xDI Designer give the availability to create a Package Configuration

- Goal : generate a unique Package file (.pck file) in which will be built all the Mappings and Processes specified
- Once configured, it can be saved and shared, giving the possibility to re-generate it with one click or through the command line utility

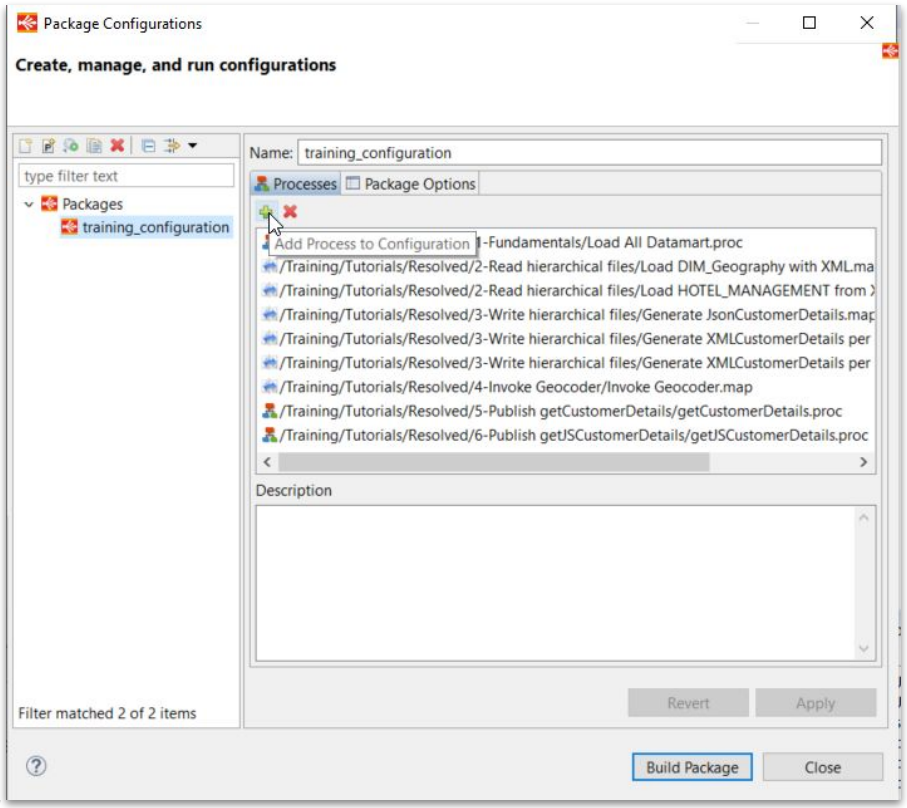


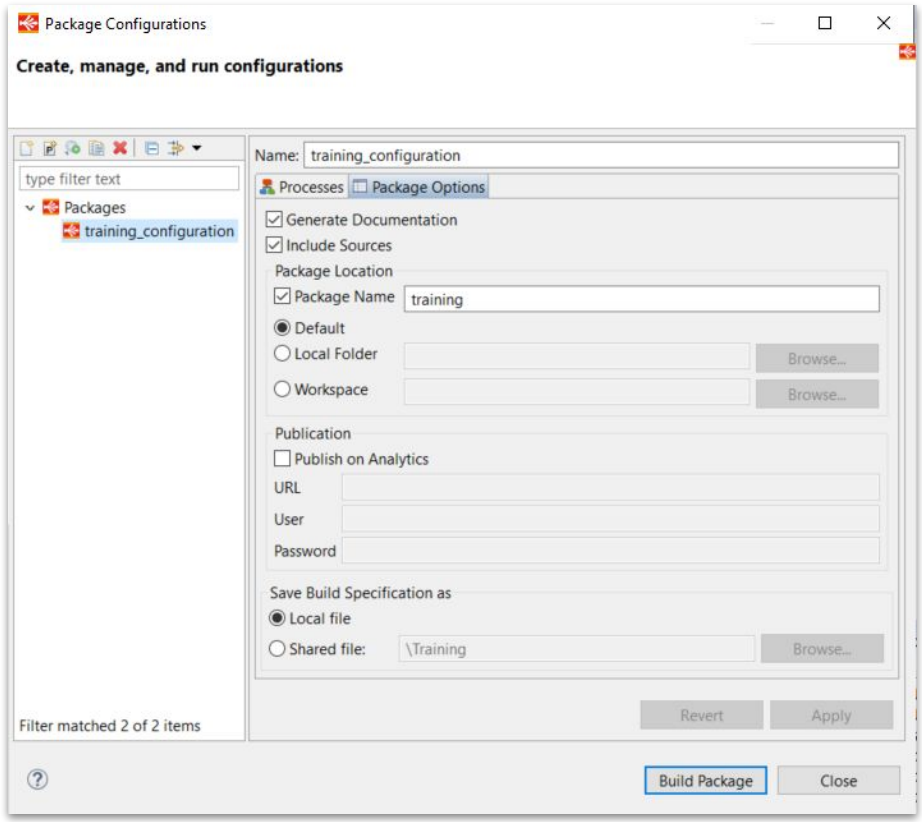


Build Package Configuration

Two tabs to define

- Processes tab to add Processes & Mapping to take
- Package Options tab to define the options





- Web Service

Semarchy

xDI DEV

H Deployment

H2 - Command Line & Scripting

Command line utility

xDI Designer has a command line utility used to build packages, build deliveries, extract information from packages, …

- A script text file containing commands (listed in the table below) to execute must be created and mentioned in the following java order:

```
cd <xDI_Designer_folder>
```

```
java -jar plugins/org.eclipse.equinox.launcher 1.5.100.v20180827-1352.jar  -application com.indy.shell.application
-console -noSplash -data "WORKSPACE_PATH" -script "SCRIPT_PATH"  [-logFile "LOG_FILE_PATH"]
```

Command	xDI Designer version	Description
betaBuildUberRuntime	S20.5.0 and higher	Create a JAR shipping a configured runtime containing all its deliveries
buildPackage	S19.0.19 and higher	Generate a package file from a list of Mappings and Processes.
buildDelivery	S19.0.19 and higher	Generate a delivery from a given package
launch	S19.0.0 and higher	Execute a launch configuration (defined in Stambia DI Designer from "Build > Package Configuration..." menu)
extractPackageInfo	S19.0.19 and higher	Extract various information from a given package
validateConfFile	S19.0.19 and higher	Check that a given Configuration file contains all the necessary externalized attributes for a given package
validatePackageSource	S19.0.19 and higher	Check that the sources of package matches specific user defined rules
patch	S19.0.19 and higher	Patch the sources of a given package
build package | packageWithDocumentation	*S18.3.2 and higher*	*deprecated command used to build packages*

Package generation in command line

To generate packages in command line:

- First, create a script file containing multiple build package orders based on the following syntax:

```
buildPackage -mainProcessWorkspacePath "<list_of_processes_to_include_in_package>" -package "<target_package_path>"
[-conf <configuration_name>] [-includeSources true | false] [-includeDocumentation true | false]
```

- Sample content of a scripting file named “buildPackage.txt”

```
buildPackage -mainProcessWorkspacePath "MyProject/Source Models/process_A.proc|MyProject/Source Models/indy.build/mapping_A.proc"
-package "D:/Built Items/myPackage01.pck"

buildPackage -mainProcessWorkspacePath "MyProject/Source Models/process_A.proc" -package "WKS:/Target Packages Project/myPackage01.pck"

buildPackage -mainProcessWorkspacePath "MyProject/Source Models/process_A.proc|MyProject/Source Models/process_B.proc"
-package "D:/Built Items/myPackage01.pck" -includeSources true -includeDocumentation true
```

Package extraction information in command line

The extractPackageInfo command allows to extract information from a package:

- List of Processes/sources models, the creation date, the documentation, …
- The following syntax must be used:

```
extractPackageInfo -package "<package_path>" -extract <information_to_extract> -outputFile <output_file_path>
```

- Sample of scripting file

```
extractPackageInfo -package "D:/Built/Generated Packages/myPackage.pck" -extract date
```

```
extractPackageInfo -package "D:/Built/Generated Packages/myPackage.pck" -extract conf -outputFile "D:/Extract/extractedConfiguration.conf"
```

```
extractPackageInfo -package "D:/Built/Generated Packages/myPackage.pck" -extract documentation -outputFile "D:/Extract/documentationFolder/"
```

Info to extract	Description
sourceList	When the package has been generated with the option to include sources inside, this option allows to extract the list of source models for a given package
mainProcessList	List of Processes contained in this package
manifest	Information about a given package, such as the creation date, the user which created it, the designer version used to generate, the operating system and JVM used to generate it, the list of Processes and sources contained inside
date	Date when the package has been created
user	User which created the package
packageId	Package internal id
conf	Extract package's configuration. This will extract all externalized attributes which values may need to be defined when generating deliveries from the package
documentation	Documentation included in the package

Package extraction information in command line

The extractPackageInfo command can be used to extract package's configuration:

```
extractPackageInfo -package "<package_path>" -extract conf -outputFile <output_config_file_path>
```

- The extracted file must be
  - renamed: using the name of the new configuration (e.g. PRD)
  - amended: uncomment the lines (i.e. remove hash character in first position)

DEV configuration generated with comments

```
#################################################################
### Name: super/Rdbms MetaData/Hypersonic SQL/HSQL_DatamartSampleDB
### Type: com.stambia.rdbms.server
#_wk1rMPxHEea_nbqmkK-3fQ/url=jdbc:hsqldb:hsql://localhost:62211
#_wk1rMPxHEea_nbqmkK-3fQ/user=sa
#_wk1rMPxHEea_nbqmkK-3fQ/password=3951C0D79B227B95C1DC348DD0BCE8F1
#################################################################
### Name: super/File//Reference_Files_Folder
### Type: com.stambia.file.directory
#_sLr3QIq4Eeq3wNFdv4f2nQ/path=C:\stb_wks\Training\Files_In\Reference_Files
#################################################################
### Name: super/File//Statistic_Report_Folder
### Type: com.stambia.file.directory
#_sLtseYq4Eeq3wNFdv4f2nQ/path=C:\stb_wks\Training\Files_Out\Statistic_Report
```

PRD configuration updated with removed comments

```
#################################################################
### Name: super/Rdbms MetaData/Hypersonic SQL/HSQL_DatamartSampleDB
### Type: com.stambia.rdbms.server
_wk1rMPxHEea_nbqmkK-3fQ/url=jdbc:hsqldb:hsql://localhost:62211
_wk1rMPxHEea_nbqmkK-3fQ/user=sa
_wk1rMPxHEea_nbqmkK-3fQ/password=3951C0D79B227B95C1DC348DD0BCE8F1
#################################################################
### Name: super/File//Reference_Files_Folder
### Type: com.stambia.file.directory
_sLr3QIq4Eeq3wNFdv4f2nQ/path=C:\stb_wks\Training\Files_In\Reference_Files\PROD
#################################################################
### Name: super/File//Statistic_Report_Folder
### Type: com.stambia.file.directory
_sLtseYq4Eeq3wNFdv4f2nQ/path=C:\stb_wks\Training\Files_Out\Statistic_Report
```

The configuration file can gradually be shared / consolidated with all the packages

Validate a configuration file in command line

The validateConfFile command allows to check if a given configuration file is well configured for a given package

- It will check that all the required attributes are well valuated in the configuration file and optionally write all the missing ones in a user defined output file
- The following syntax must be used:

```
validateConfFile -package "<package_path>" -confFile <configuration_name> [-missingConfFile
<file_path_for_writing_detected_missing_attributes>] [-missingConfBehavior MERGE | OVERWRITE]
```

- Sample of scripting file

```
validateConfFile -package "D:/Built/Generated Packages/myPackage.pck" -confFile "D:/configuration/myConfiguration.conf"
-missingConfFile "D:/validation/missingAttributes.conf" -missingConfBehaviour MERGE
```

If missingConfBehavior parameter has “MERGE” value, the missing attributes are appended to the existing file. If the value is “OVERWRITE”, the existing file is deleted and missing attributes are added in the new created file

Build a delivery in command line

The buildDelivery command allows to generate delivery files from a package

- It will check that all the required attributes are well valuated in the configuration file and optionally write all the missing ones in a user defined output file
- The following syntax must be used:

```
buildDelivery -package "<target_package_path>" [-buildmode <build_mode>] [-confFile <configuration_file_path>]
[-outputFolder <output_folder_path>] [-processName <Name_of_the_Process_to_build>] [-verbose true | false]
[-verifyModules <Folder_of_the_modules>]
```

- Sample of scripting file

```
buildDelivery -package "D:/Built Items/myPackage01.pck" -buildmode substitution -confFile "D:/Configuration Files/confA.conf" -processName "process_A"
```

build mode	Description
generation	Mode which launches a generation mechanism to build the delivery. The principal advantage of this mode is that as a new generation step is performed, user does not have to set all the parameters in the extracted configuration file, neither to use a configuration file. The default value with which the package has been generated is used for parameters which has not been valuated in the configuration file. Another advantage is that it allows to customize Metadata parameters which were not defined as externalized when generating the package. This offers the possibility to add extra parameters on the extracted configuration file used to build the delivery, to replace parameters which were not anticipated when generating the package.
substitution	Default mode which consists of simply replacing the externalized attributes with the values provided inside the extracted configuration file. The builder simply retrieves the pre-built delivery which is inside the package and replace the parameters in. This is also the method that uses Stambia Production Analytics for generating deliveries, and this offers better performances as no generation is performed to build the delivery. This also offers the possibility to generate deliveries for all the processes contained in a multi-process package.

Delivery execution in command line

The production teams are able to execute deliveries inside an external scheduler or directly on the operating system

- The following command line syntax:

```
startdelivery.[sh|bat] -name <delivery_name> [-var <variable_path> <variable_value>]
 [-sessionName <alternate_session_name>]
```

- Must be launched from the Runtime folder
- Sample with two input parameters

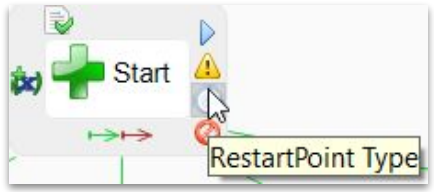
```
startdelivery.bat -name deliv1 -var ~/STATE_CODE CA -var ~/CITY_NAME "Los Angeles"
```

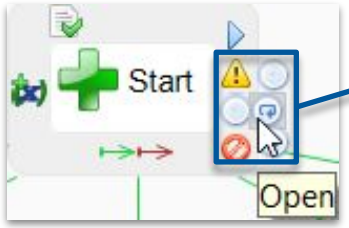
Restart an execution in command line

To restart an execution in command line, use the following syntax:

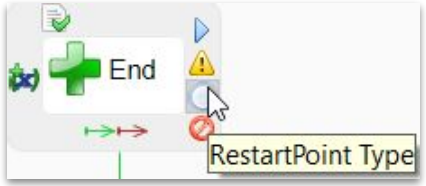
```
./startcommand.[sh|bat] connect to <Runtime_Host> port <Runtime_port>; restart session <Session_id>
```

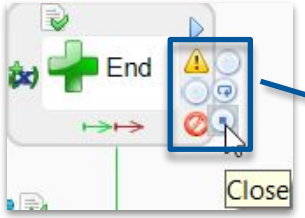
- The execution will restart from the restart point (if defined) or where the process failed
  - To “Open” a Restart Point:

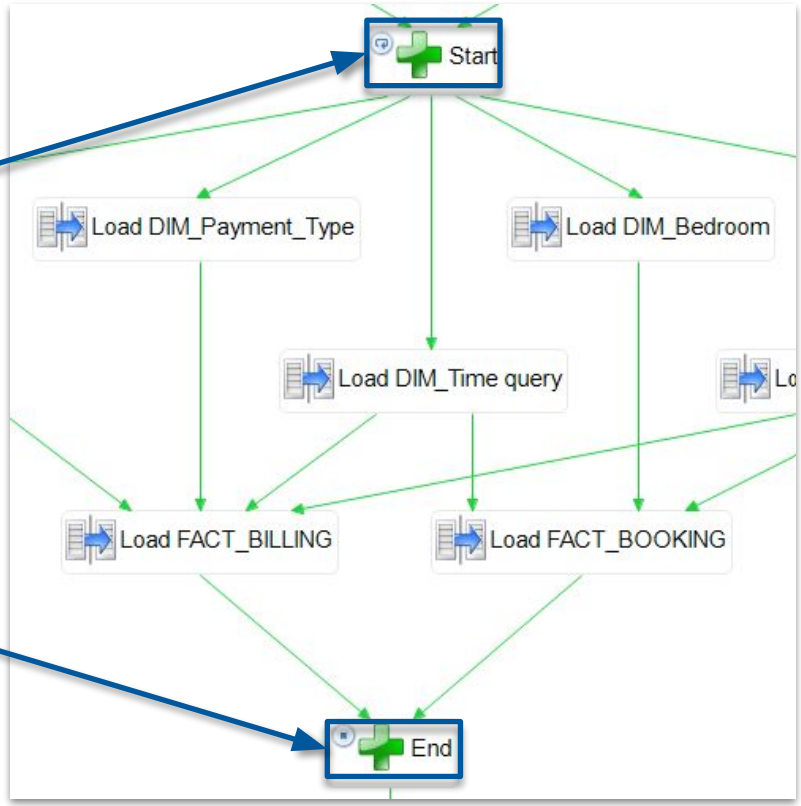




  - To “Close” a Restart Point:







To go further

Document Type	Link
Stambia.org article Build package configuration	https://stambia.org/doc/222-development-hints-and-tips/building-packages/540-build-package-configuration
Stambia.org article Building deliveries from build scripts	https://stambia.org/doc/58-stambia-di-software/runtime/how-to/602-building-deliveries-from-build-scripts
Stambia.org article Building deliveries with Designer Command Lines	https://stambia.org/doc/57-stambia-di-software/designer/how-to/723-getting-started-with-designer-command-lines


Semarchy
Questions?