



**energy components**

# **Energy Components**

## **Technical Overview**

1. Energy Components - Technical Overview .....	3
1.1 Energy Components Business Areas .....	4
1.2 Energy Components Basic User Guide .....	7
1.2.1 Energy Components: Getting Started .....	8
1.2.2 The Energy Components Screen .....	10
1.2.3 Help .....	17
1.3 Energy Components Product Concepts .....	18
1.3.1 Configuration overview .....	19
1.3.2 Calculation Framework .....	22
1.3.3 Classes and Objects .....	26
1.3.4 Group model .....	27
1.3.5 User roles and Access Rights .....	28
1.3.6 Reporting .....	29
1.4 Energy Components Technical Documentation .....	31
1.4.1 EC Production .....	32
1.4.1.1 EC Production System Attributes .....	33
1.4.1.2 EC Reporting Layer .....	37
1.4.1.3 EC Production Dashboard .....	40
1.4.2 EC Sales .....	55
1.4.2.1 EC Sales Technical Documentation .....	56
1.4.3 EC Transport .....	61
1.4.3.1 EC Transport Technical Documentation .....	62
1.4.3.2 EC Transport Dashboard .....	72
1.4.4 EC Revenue .....	77
1.4.4.1 EC Revenue System Attributes .....	78
1.4.4.2 EC Sales - EC Revenue Interface .....	85
1.4.4.3 EC Revenue Dashboards .....	93
1.4.5 EC Framework .....	109
1.4.5.1 PKI Technical Documentation .....	110
1.4.5.2 Advanced File Import - Technical Reference .....	115
1.4.5.3 EC FRMW Standard Reports Configuration Guide .....	148
1.4.5.4 EC IS Configuration Guide .....	153
1.4.5.5 EC IS Staging Table Extension .....	171
1.4.5.6 EC View Generator and Object Class Model .....	179
1.4.5.7 How to Setup Customer Defined Password Validation .....	257
1.4.5.8 How to Setup Timezone .....	258
1.4.5.9 How to Use Date Macro Parameter .....	261
1.4.5.10 Row-Level Security .....	264
1.4.5.11 How to Setup SSL/HTTPS .....	268
1.4.5.12 Smart Journaling .....	272
1.4.5.13 How to create, install and configure a Jasper Report .....	276
1.4.5.14 Messaging Technical Documentation .....	278
1.4.5.15 How to Encrypt Passwords in EC 11 .....	307
1.4.5.16 Integration Services Technical Documentation .....	311
1.4.5.17 EC Web Services .....	382
1.4.5.18 EC Dashboard Configuration Guide .....	386
1.5 BPM Technical Documentation .....	399
1.5.1 BPM Introduction .....	400
1.5.2 Architecture .....	401
1.5.3 Single Sign On .....	403
1.5.4 Repositories .....	405
1.5.5 Authoring Processes .....	412
1.5.6 Process Execution and Management .....	426
1.5.7 Business Action Invocation .....	434
1.5.8 Process Action .....	442
1.5.9 User Tasks and To-do List .....	454
1.5.10 Samples .....	460
1.5.11 API .....	472

# Energy Components - Technical Overview

This document provides a Technical Overview about Energy Components (EC). It contains chapters describing the Business Areas and related functionality, a basic user guide about how to get started, technical documentation for the different modules, and information about the process automation in EC.

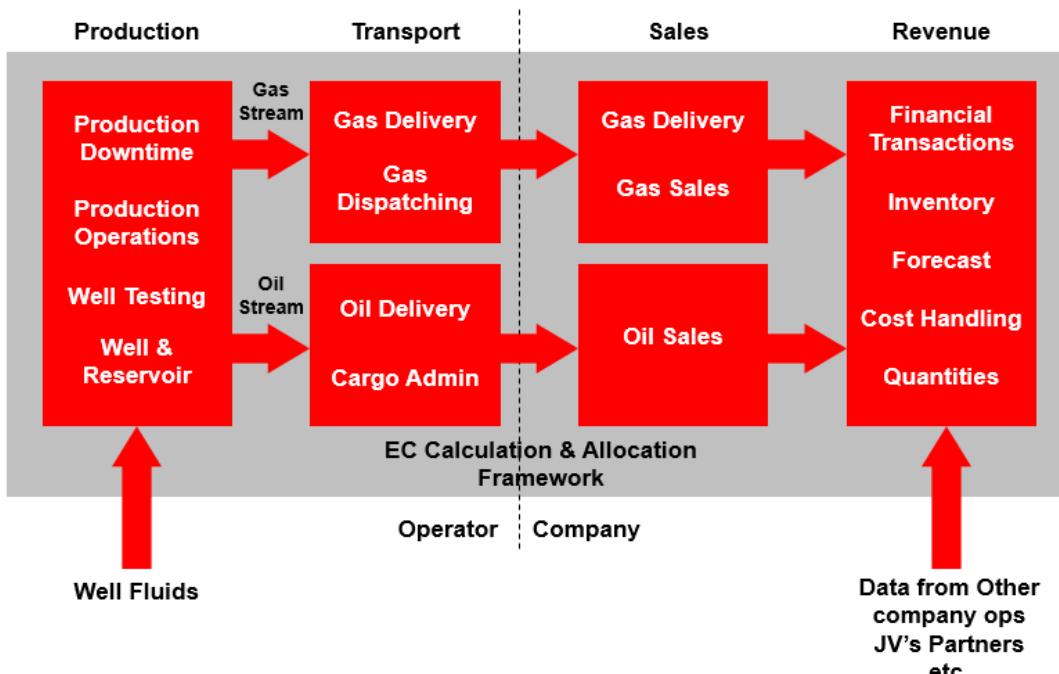
- Energy Components Business Areas
- Energy Components Basic User Guide
- Energy Components Product Concepts
- Energy Components Technical Documentation
- BPM Technical Documentation

## Energy Components Business Areas

This Chapter contains information about Energy Components (EC) and provides information about the different business areas covered by EC, and details about the specific functionality that is present within each module.

Energy Components (EC) covers four major business areas: **Production; Transport; Sales and Revenue**. Each of the four business areas has its own module and relating screens.

Below is a summary of each business area within EC:



### Overview of the Energy Components System

## EC Production

The EC Production module supports all aspects of production management from forecasting to full production. The module is able to take into account specific production requirements including gas/condensate fields, coal-bed methane fields, heavy oil and bitumen production and onshore pumped wells (without any automation and tank farms). This module is also able to track actual performance against targets and identify bottlenecks in order to maximise production from your assets. With EC Production users can:

- Automatically acquire data from DCS, data historians and other sources
- Perform comprehensive validation and quality control on all data
- Record well tests and calculate test results for individual wells
- Estimate individual well rates based on a range of methods, taking into account enhanced recovery mechanisms, including full support for so-called "intelligent" wells
- Allocate each phase back to individual wells and completions
- Allocate sales products back to individual wells and completions
- Allocate production down to reservoir zones
- Manage use and inventory of chemicals
- Perform emission reporting and environmental accounting
- Record all data required for daily production and operation reports
- Perform inventory management including volumetric calculations based on e.g. tank strapping tables
- Execute full 'ownership allocation' including royalty and production sharing arrangements
- Manage production allowances, forecasts, targets and potentials
- Adopt a systematic approach to production downtime and deferments based on widely accepted principles,

- including the allocation of deferment quantities to events
- Perform calculations of key performance indicators
- Report internally and externally to JV partners, as well as all regulatory reporting

## EC Transport

The EC Transport module covers the transportation of hydrocarbons either through pipeline systems or by vessels. EC transport is used for a variety of transportation requirements including oil terminals, LNG Export/Import terminals, gas plants and pipeline transport operations. With EC Transport users can:

- Perform cargo scheduling for multiple crude blends, condensates, LNG and other products. Issue official and tentative lifting programme. EC can generate a lifting program based on entitlements and fixed cargo sizes (often the situation with offshore liftings) or build the lifting program based on nominations.
- Record relevant data from tanker inspections and keep history record per ship
- Issue all official cargo documents as defined in relevant Lifting Procedure, including e.g. Bill of Lading, Certificate of Quantity, Certificate of Quality, Time sheet, etc.
- Demurrage handling including invoicing and payment tracking
- Handle claims and fail-to-lift situations
- Perform complete lift accounting per lifter per field and per product. Handle any swap arrangements between lifters
- Calculate entitlements according to JV arrangements and prevailing royalty/PSA terms
- Perform crude oil value adjustment supporting both in-kind and in-cash settlements
- Perform scheduling LNG vessel arrivals, and storage evolution at LNG import terminals
- Manage nominations and re-nominations from shippers. EC offers a message broker solution ensuring safe and controlled data exchange with shippers and other parties.
- Assess available capacity against aggregate nominations at each network point and handle curtailment situations
- Implement and manage gas storage, balancing and other flexibility arrangements
- Handle title transfer and arrangements where parties buy/sell from each other
- Perform matching of nominations with any adjacent transport network operators
- Record measurements of quantity and quality on a continuous basis
- Perform allocation and reconciliation of all data at periodical intervals

## EC Sales

The EC Sales module includes functionally that supports the sales organisation in all aspects of selling/trading hydrocarbons including managing complex gas sales contracts. With EC Sales the user can:

- Manage nominations, re-nominations and requests from individual buyers, and validate against contractual terms
- Manage gas availability from own production, gas storage, balancing positions, substitution arrangements, etc.
- Assess contractual obligations against gas availability and deal with any shortfall situations
- Place shippers nominations to all relevant gas transport service operators
- Attribute actual deliveries to sales contracts according to priority and user-defined rules
- Perform full contractual accounting according to contract clauses. For take-or-pay contracts this involves handling e.g. carry-forward gas, make-up gas, shortfall allowances, etc.
- Calculate prices in multiple currencies according to published reference prices, consumer price indexes and applicable contract terms

## EC Revenue

The EC Revenue module covers the valuation of hydrocarbon Sales and Purchases as well as valuation of Tariff Income and Tariff Cost for infrastructure usage, including invoicing and revenue allocation and distribution of these financial transactions. Energy Components is the only solution that addresses the complete value chain from reservoir to revenue. It is specifically designed for the upstream industry. The system can handle some of the most complex revenue arrangements, including lifting agreements between joint venture parties, royalty/PSA agreements and multi-product, multi-currency invoicing. The module offers full traceability and auditability - supporting SOX404 compliance. With EC Revenue users can:

- Valuate all hydrocarbon deliveries using prevailing prices and applicable contractual terms
- Generate and submit invoices, calculate interest rates and track payments
- Perform accrual and preliminary postings

- Perform prior period adjustments
- Fully automated invoicing based on contractual rules and settings
- Integrate with financial accounting systems for booking of cost and revenue
- Valuate inventories and book value of hydrocarbon inventory positions
- Calculate and book remaining reserves and production related depreciation keys (UOP)
- Perform production and revenue forecasting and budgeting, including regular closure forecast for remaining months in year (both calendar and gas-year supported)
- Interface with financial accounting system for booking of all transactions. The interface will update status of each booking in EC once the accounting system has confirmed the booking.
- Allowing of uploading and manipulating (grouping and splitting values and quantities) for the ERP system to produce amounts and values that can be used for reporting such as allowable deductible costs.
- Perform complex inventory valuations through a value change, allowing tracking production, purchase, transportation and other costs associated to each unit.

## Energy Components Basic User Guide

This Basic User guide does not contain all the information about the system, but provides a description of how to get started. It also includes information about The Energy Component screen and details about how to navigate and use the different windows and panes.

Through the integrated online help, the system will automatically provide the user with more detailed information based upon the current screen and the customers system setup.

- Energy Components: Getting Started
- The Energy Components Screen
- Help

## Energy Components: Getting Started

Energy Components is a module-based application software with fully configurable screens and calculation models.

It is a web-based application, accessed through Internet Explorer - this means you do not need to install any software. To access the system you will need to get the Energy Components URL and a username and password from your IT Department.

### Basic System Requirements

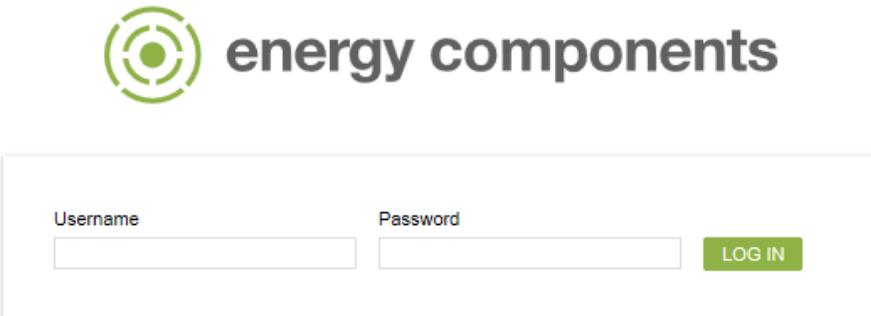
- Browser : IE v11 or later, best-effort-basis support (i.e verified to run on) latest versions of Firefox and Chrome.
- Operating System: No specific requirement
- Screen Resolution: minimum 1280 x 1024



**Because Energy Components is web based, any data you enter into a screen exists on your screen until you save it. Once you have saved the information it goes into the central database and is available to anyone who has access to the screen or record.**

### Logging On

Energy Components is a web-based application. To access the software, open your web browser (which must be according to basic system requirements). In the address bar of your web browser, type in the Energy Components web address (contact your IT department for the web address). The login screen (below) will appear. You will then need to type in your User Name and Password and click the Login button.



**Once you have logged in, do not use the Internet Explorer back, forward or refresh buttons. Use the controls within the Energy Components screens. If you use the Internet Explorer buttons you are likely to be logged out and might lose unsaved data.**

### Changing Your Password

You will be required to change your password every 30 days or the period set by your system administrator. When your password is nearing its renewal date you will see this screen when you login:



Your password will expire in 0 day(s)

Current Password

New Password

Retype New Password

To change your password, enter your existing password in the first box. Then, enter your new password in the second box and again in the third box. Press the 'Save' icon on the Tool Bar. In EC the password rules are configurable. Typically there will be rules like: your password must be at least six characters long and can contain both numbers and letters.

## The Energy Components Screen

The Energy Component screen is made up of seven different windows or panes: Screen Bar, Tool Bar, Tree View, Data Window, Navigator Pane, Status Area and the Internet Explorer Status Line. These panes will always stay the same, with the exception of the Data Window and the Navigator Pane, which change depending on the type of operations being carried out.

Below is a summary of the different windows or panes within the Energy Components screen:

**Screen Bar** Displays the screen name and login id for the person who is logged in

**Tool Bar** Displays the standard Energy Components functions as icons e.g. 'Save', 'Delete' and 'Insert'

**Tree View** Used to navigate to the specific Energy Components screen or business function you require

**Navigator Pane** Used to filter what data to display in the Data Window, e.g. select period, assets, etc.

**Data Window** Displays the data you have selected in the Tree View and the Navigator Pane

**Status Area** Displays information about record creation / updates, record revisions, record status, validation info, trending info and hints and tips

The screenshot shows the 'Daily Oil Stream Status' screen. The Data Window contains two tables: 'MEASURED FIGURES' and 'DERIVED AND CALCULATED FIGURES'. The 'MEASURED FIGURES' table has columns for Stream Name, Grs Vol [m³], BSW [m³], Calc Net Vol [m³], Alloc Net Vol [m³], Alloc Factor, Client [N], Client Vol [m³], Shrink Vol [m³], Shrinkage, Calc Diluent [N], Grs Mass [kg], Total Vol [m³], 20d Avg Vol [m³], Gas Oil Ratio [bbl/tonnes], and Oil Spec Gravity. The 'DERIVED AND CALCULATED FIGURES' table has columns for Stream Name, Grs Vol [m³], Net Vol [m³], Alloc Net Vol [m³], and Alloc Factor. The Status Area at the bottom shows record status, revision info, approval status, hints & tips, validation, and trending.

### Screen Bar

The Screen Bar displays the Login ID of the user logged into the session (in this case sysadmin), logout icon and the system menu bars:



The icons on the Tool Bar have specific meanings as described in the following table:

Icon	Function
	<b>Logout</b> Ends the current user session and returns the user to the application Login screen
	<b>Help</b> Context sensitive help for the selected functional area

### Tool Bar

The Energy Components Tool Bar has eight icons (to use the icons, left click on them):

new



The icons on the Tool Bar have specific meanings as described in the following table:

Icon	Function
	<b>New</b> Insert fields, ready for new registration of data
	<b>Save</b> Saves updated data in the Data Window to the database
	<b>Retrieve/Refresh</b> Refreshes the display
	<b>Delete</b> Marks the selected row for deletion
	<b>Hide/Show Tree View and Status Area</b> This shows or hides the Tree View area and the Status Area of the screen to make more horizontal space available for the display of data. This is particularly useful if you have to scroll across the screen to see a complete record

When a function on the Tool Bar is not available for use it is 'greyed out' to show that it is deactivated. In some screens when you start to add data, icons which have previously been deactivated will become activated.

## Tree View

The Tree View is the basic navigation mechanism by which you can access screens and functions within the Energy Components system. It is similar in structure to Windows Explorer file menu. It is an expandable tree, which contains all of the menu options you are allowed to access. The menu supports several levels of indentation, so that screens can be grouped together under meaningful headings and in a way that supports the workflow process of the function. Select the relevant folder and file from the Tree View menu depending on the operation you wish to carry out.

The Tree View window has two distinct areas. The top portion of the Tree View contains configuration and maintenance functions. The lower portion contains business functions which are grouped together by specific function, i.e. Allocation, Terminal Operation, etc.

For ease of use, there is also a search field, which allows you to search for the Energy Components screen or business function you are searching for. You can also drag and drop your favorite screens into the favorite section for easy access.

The screenshot below ha been taken from a system that has access to all screens. The information displayed in the Tree View (as shown below) will vary depending on the access rights you have been granted and your business area. Your access rights will be set by the system administrator.



## SEARCH

system|

- EC Codes - Non-System Codes
- EC Codes - System Codes
- HCB System
- HCB System Items
- Maintain System Reference Value
- Maintain System Settings
- System Attributes

## FAVORITES

- Daily Dashboard
- Daily Production Well Status 1

## MENU

- > CONFIGURATION
- > EC PRODUCTION
- > EC SALES
- > EC REVENUE
- > REPORTING
- > PROCESS AUTOMATION
- > MESSAGING
- > TASK LIST
- > EC INTEGRATION SERVICE

**Navigator Pane**

After selecting the appropriate screen on the Tree View menu, you can use the Navigator to select the class of data or date range you wish to view. There are two types of Navigator Panes, depending on the type of data you want to view: Standard Navigator and Filter Navigator.

**Standard Navigator**

The Standard Navigator consists of options to select the date range by clicking on the calendar icon. You can select the asset by clicking on any of the other drop-down menus. Click 'Go', and the information you require will appear in the data pane.

Date 2011-1-1	Production Unit P1 Production Unit	Area P1 Area	Facility Class 1 P1 Facility 1	
------------------	---------------------------------------	-----------------	-----------------------------------	--

## Data Window

The Data Window has several different layouts. The Data Window layout will depend on the specific screen you have selected from the Tree View window and the information selected in the Navigator Pane. The image below shows an example of a typical Data Window layout. This screen shows all the detail of a record in one list.

MEASURED FIGURES															
Stream Name	Stream Volume Data			Stream Volume Data			Stream Mass Data			Historical Gross Volume					
	Grs Vol [Sm³]	BSAW [%]	Calc Net Vol [Sm³]	Alloc Net Vol [Sm³]	Alloc Factor	Diluent [%]	Diluent Vol [Sm³]	Shrunk Vol [Sm³]	Shrinkage	Calc Diluent [%]	Grs Mass [kg]	Yday Vol [Sm³]	30d Avg Vol [Sm³]	Gas Oil Ratio [Nm³/m³]	Oil Spec Gravity
P1_S001_M_OIL.PO.0001	1,350.0	2.00	1,323.0												
P1_S024_M_OIL.WR.0038			1,000.0												
P1_S072_M_OIL_PP.0028												1,200.0			
P1_S074_M_OIL_PP.0028	1,400.0		1,400.0												
P1_S075_M_OIL_PP.0028	1,600.0		1,600.0									1,300.0			
P1_S076_M_OIL_PP.0028	1,700.0		1,700.0												
P1_S083_M_OIL.PO.0005_04	1,500.0		1,500.0												
P1_S008_M_OIL.PO.0001	1,500.0		1,058.4												

DERIVED AND CALCULATED FIGURES																
Stream Name	Grs Vol [Sm³]	Net Vol [Sm³]	Alloc Net Vol [Sm³]	Alloc Factor												
P1_S009_M_OIL.PO.0040	43.5	33.6														
P1_S011_M_OIL.PO.0056	10.0	9.8														
P1_S012_M_OIL.PO.0056	9.8	9.7														
P1_S013_M_OIL.PO.0045	45.0	34.6														
P1_S014_M_OIL.PO.0046	432.0	427.7														
P1_S023_D_OIL.PO.0001_02	13,500.0	13,230.0														
P1_S063_M_OIL.PO.0100_F	0.0	0.0														
P1_S078_M_OIL.PO.0100_F	72.4	55.7														
P1_S082_M_OIL.PO.0056		12.7														
P1_S107_D_OIL.PO.0021	50.0	50.0														
P1_S081_M_OIL.PO.0056	12.0															

To update the data:

- Place your cursor in the record you wish to update
- Type in the new value or use the drop down lists to update information



- If a data entry field is white you have the option to add data to the field to complete an action
- If a data entry field is yellow it is mandatory that you add data to the field before an action can be completed
- If a data entry field is blue the field is read only

## Status Area

The Status Area (below) appears at the bottom of the main screen. It displays information about the specific screen that have been accessed in the Tree View menu. This is a standard section at the foot of each screen and includes five standard tab pages, including: Record Status, Revision Info, Hints & Tips, Validation and Trending.

RECORD STATUS	REVISION INFO	APPROVAL STATUS	HINTS & TIPS	VALIDATION	TRENDING
Created by <input type="text" value="ECKERNEL_ECRCS5_DB"/>	<input type="text" value="2014-12-12T21:52:00"/>	<input type="text" value="Record status"/>	<input type="text" value="Provisional, with 0 revision(s)."/>		
Last updated by <input type="text" value="sysadmin"/>	<input type="text" value="2014-12-16T18:50:21"/>	<input type="text" value="Revision text"/>	<input type="text" value="Revision Event"/>	<input type="button" value="Revision Event"/>	

## Record Status

The Record Status Tab (below) tells you who created a record and when and who made the most recent changes to the record.

RECORD STATUS	REVISION INFO	APPROVAL STATUS	HINTS & TIPS	VALIDATION	TRENDING
Created by <input type="text" value="ECKERNEL_ECRCS5_DB"/>	<input type="text" value="2014-12-12T21:52:00"/>	<input type="text" value="Record status"/>	<input type="text" value="Provisional, with 0 revision(s)."/>		
Last updated by <input type="text" value="sysadmin"/>	<input type="text" value="2014-12-16T18:50:21"/>	<input type="text" value="Revision text"/>	<input type="text" value="Revision Event"/>	<input type="button" value="Revision Event"/>	

The Record Status Tab page displays the following information for a selected item of data on the screen:



- User who created the item of data
- Date and time when the item of data was created
- User who last updated the item of data
- Date and time when the item of data was last updated
- Record status of the item of data, e.g. provisional or approved data, number of revisions
  - Provisional status is the default status for a record until a job is run and the status updates
  - Verified status is typically used when data has been automatically loaded into Energy Components using the EC data capture features or other interfaces
  - Approved status is where a job is run on the data, updating the status to the highest security level
- Revision text associated with the current revision of the data item

## Revision Info

This Revision Info Tab (below) lists all the changes that have been made to a record since its creation. This list is updated each time a user makes a change to the record.

REVISION INFO APPROVAL STATUS HINTS & TIPS VALIDATION TRENDING												
User Name	Journal Date	Record Status	Rev Event	Well Name	On Stream [h]	Choke 1	Choke UOM	Well Head		Downhole		
								WHT [°C]	WHP [barg]	DHT [°C]	DHP [barg]	Mess Diluent [Sm³]
admin	2014-12-16 21	Provisional		P1 W001 OP		100		75.0	100.0	100.0	125.0	
admin	2014-12-16 21	Provisional		P1 W001 OP		100				100.0	125.0	
admin	2014-12-16 21	Provisional		P1 W001 OP		100						
KERNEL_E	2014-12-12 21	Provisional		P1 W001 OP								

The Revision Info Tab page displays the following information for a selected item of data on the screen:

- List of all revisions associated with the data item
- The actual value the item has in each revision (highlighted the changed values)
- For each revision, the name of the user who updated the data
- For each revision, the date/time when the data was updated
- For each revision, the text associated with that revision

## Hints & Tips

The Hints & Tips Tab can be used to display and edit helpful notes relating to a particular screen. An example of the Hints and Tips section of the Status Area is shown here:

RECORD STATUS REVISION INFO APPROVAL STATUS HINTS & TIPS VALIDATION TRENDING
User defined hint and tips for the screen   <input type="button" value="SAVE"/>

To add a Hint or Tip:

- I. Click on the Hints & Tips tab
- II. Enter the text
- III. Click on Save
- IV. When you go back to visit this screen, the hint/tip will be displayed

## Validation

The Validation Tab is a log of the results of validation checks, created from using of the 'Check Rules' function. Any validation failures will appear in the Validation Tab screen. To clear the validation log, correct the record/s by re-running the validation process or alternatively, delete the errors from the log. You can re-run a single record by selecting the 'Re-run selected' button or re-run an entire process by selecting the 'Run all' button.

Object Name	Ignore	Daytime	Severity	Message	Validation Time	RE-RUN SELECTED
P1 W006 OP	<input type="checkbox"/>	2011-01-01T00:00:00	ERROR	Well P1 W006 OP is missing theoretical oil rate, cannot allocate	2014-12-16T21:40:37	<input type="button" value="RUN ALL"/>
P1 W007 OP	<input type="checkbox"/>	2011-01-01T00:00:00	ERROR	Well P1 W007 OP is missing theoretical oil rate, cannot allocate	2014-12-16T21:40:37	<input type="button" value="SAVE"/>
P1 W015 OP	<input type="checkbox"/>	2011-01-01T00:00:00	ERROR	Well P1 W015 OP is missing theoretical oil rate, cannot allocate	2014-12-16T21:40:37	

## Trending

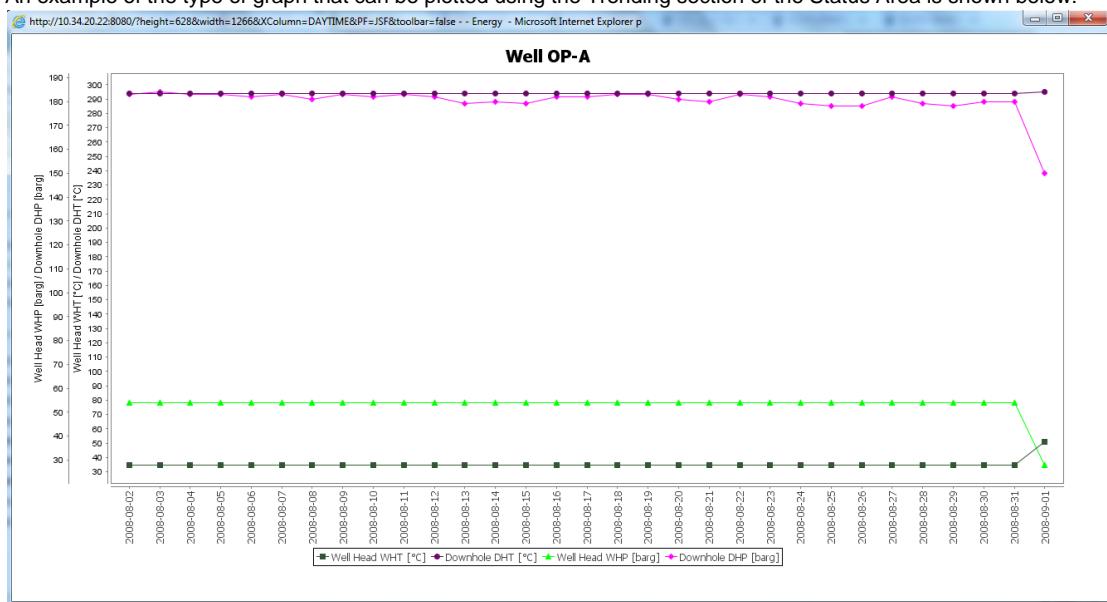
The Trending Tab allows the user to select one or more data items from the screen to display in a graph, shown in an Internet Explorer window, or to export the figures into a file (for example, an Excel spreadsheet). The Trending Tab also allows users to specify the date range that the graph is plotted over. Trending is a useful way of viewing how data has changed over a period of time. An example screen showing the Trending section of the Status Area, is shown below:

RECORD STATUS	REVISION INFO	APPROVAL STATUS	HINTS & TIPS	VALIDATION	TRENDING
From Date 2010-12-02	<input type="checkbox"/> Grs Vol	<input type="checkbox"/> Alloc Factor		DRAW...	Format pdf
To Date 2011-01-01	<input checked="" type="checkbox"/> Net Vol				EXPORT...

EC allows you to produce a Trending graph. To produce a Trending graph:

- I. Select the required record (make sure the cursor is actually on the record – if the record is not properly selected a message saying 'No columns selected' will appear)
- II. Then, select the Trending Tab
- III. On the Trending Tab, enter the 'From Date' and 'To Date' date parameters using the drop-down calendar
- IV. Next, select the check boxes with the data items for the graph to be plotted against. In the example above, three data areas have been selected: 'On Strm', 'Gas Vol' and 'Temperature'
- V. Select the draw button on the Trending Tab:
- VI. A new Internet Explorer window will pop up with the completed graph

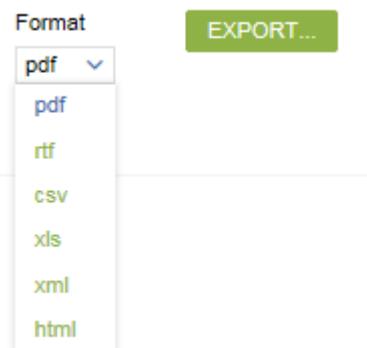
An example of the type of graph that can be plotted using the Trending section of the Status Area is shown below:



If you use a large date range and select several columns the graph will take longer to produce as the system retrieves the data

Once the graph has been produced it is possible to print or save the image. Place your cursor on the graph and right click the mouse button. A Windows menu will appear allowing you to copy or print the graph. If you choose to copy, use the "Paste Special" option and paste as a 'bitmap' or a 'PNG' file.

EC also allows you to export the data as a table into an Excel spreadsheet as a .csv or .xls file. To perform this function, follow steps I to V of the process described above to produce a Trending graph and set the 'Format' option to .csv using the drop-down menu (as shown below).



Then select the 'Export' button:

A Windows dialog box will appear and prompt you to save the file. Once the file is saved as a .csv or .xls file, it can be opened in Excel.

## Help

The 'Help' system in Energy Components is contextualised depending on which screen you are using. For example, if you are in the Daily Gas Stream Status screen, the help information provided will relate specifically to the Daily Gas Stream Status screen.

Each help file contains the following:

- An image of the screen
- A description of the business function for the screen
- Information on the functionality of the screen
- Information on interdependencies i.e. other screens that rely on information from the particular screen and screens that need to be completed before the screen can properly complete its function

Users with valid access are allowed to update the help content. To do so, open the help screen and click inside the description area. You will then receive a text editor where you can do your modifications, and tick the accept button (v) when you are done. It is also possible to add new screenshots.

## Energy Components Product Concepts

This chapter contains information about the EC product Concepts and how the system can be configured to the customers specific needs, defining back-end functionality, adapting the Graphical User Interface, applying rules for the clients business processes through the calculation framework, managing users and access, and how reports can be defined through the reporting layer.

- Configuration overview
- Calculation Framework
- Classes and Objects
- Group model
- User roles and Access Rights
- Reporting

## Configuration overview

The major concept behind Energy Components is that it is configurable without the need for programming expertise or system downtime. Both the backend functionality and the user interface can be configured. A unique feature of Energy Components is the ability to allow clients to re-configure the system without the need for high-level programming skills. The configuration changes are quick to make and take effect immediately. New configurations can be added while the system is in use – avoiding the need for system or user downtime. This makes system changes financially efficient as well as time efficient. Configurable areas include:

- User interface, screens and navigation features (referred to as 'Business Functions' in Energy Components)
- The methods the system uses to react, calculate and process data (referred to as 'Business Processes' in Energy Components)

With the configuration capability, Energy Components can:

- Include or hide new columns
- Configure calculated numbers in the reporting layer that can then be displayed as and when required
- Be configured for multiple languages
- Be configured by role or record status

As illustrated in the screen shown below, the following areas within Energy Components can be configured by Tieto prior to roll-out, or by the client after roll-out:

- Tree View
- Navigator
- Access to Objects
- Units
- Audit Tracking
- Check Rules
- Record Status

## Screen Configuration

Well Name	Opening Daytime	Closing Daytime	Totalizer Reading			Calculated			Reference Factors		
			Opening	Override	Closing	Adj Vol [Sm³]	Net Vol [Sm³]	Daily Rate	Meter	Conversion	Temp
P1_W020_OP	2011-01-01 12:00	2011-01-02 12:00	5000	11000	0.0	594.0	1,188.0	11000	0.1000	0.9000	0.9000
P1_W020_OP	2011-01-01 00:00	2011-01-01 12:00	0	5000	0.0	495.0	990.0	11000	0.1000	0.9000	0.9000
P1_W021_OP	2011-01-01 00:00	2011-01-01 12:00	0	4000	0.0	398.0	792.0	11000	0.1000	0.9000	0.9000
P1_W085_CP	2011-01-01 00:00	2011-01-01 12:00	0	6000	0.0	594.0	1,188.0	11000	0.1000	0.9000	0.9000
P1_W086_CP	2011-01-01 00:00	2011-01-01 12:00	0	7000	0.0	693.0	1,386.0	11000	0.1000	0.9000	0.9000

- Tree View menu can be configured
- Navigator can be configured
- Access to Objects can be configured
- Units can be configured

## Configuration Options

### Tree View

The Tree View can be tailored to meet the requirements of each customer. Energy Components comes with a standard menu layout where business functions are grouped together. These can easily be changed to reflect the customers' workflow and processes. The Tree View will contain different functions for different users depending on their business area and the users' level of access rights. When a user does not have access to a given screen or business function, the screen will not be visible on the Tree View.

For ease of use, there is also a search field, which allows you to search for the Energy Components screen or business function you are searching for. You can also drag and drop your favorite screens into the favorite section for easy access.

### Navigator

The Navigator can be configured for each screen. Most screens in Energy Components use a dynamic Group Model Navigator which reads the Group Model Configuration and populates the columns dynamically. If the Group Model Configuration is changed, the Navigator (and subsequently what is shown on the screen) will change to reflect the new model and hierarchy.

### Units

The Units used in the application can be changed. There are two types of Units used in Energy Components, these are:

**Display Unit** - Indicates what unit type should be displayed in the screens

**Storage Unit** - Indicates what unit type should be stored in the database

Each Unit is configurable for each measurement type. For instance once pressure measurements have been configured they will be stored and displayed using the same unit measurement throughout the system.



**Although it is possible to change Units within the system careful consideration should be given before changing storage units e.g. kg to tonnes. Changing the storage unit will NOT trigger a recalculation and conversion of stored data.**

### Object Access

Object Access components can be configured to limit user access to specific assets such as network points. This feature prevents users from seeing specific objects anywhere in the system either in screens or on any dropdown menus. Object Access can be set up to filter any object types. This way the system can be configured to meet the needs of specific users or groups of users (i.e. roles).

### Columns

The columns in the screens can be configured. New columns can be added and existing columns can be hidden from view. Screen labels and column sort orders can also be changed to meet your requirements.

### Calculated Numbers

Columns containing calculated values are configurable and new ones can be added. These can be hidden in certain screens, yet still made visible and available in reporting views for use in reports.

### Screen and Data Access

Screen and data access can be configured for each individual screen. The screen access is set up according to user roles, so that users can only see the screens that are relevant to their roles function. You can also control the type of access the user has within a screen.

The access levels defined in Energy Components are:

<b>No access</b>	Screen will be invisible for the users with this role
<b>Read</b>	Screen will be read only
<b>Change</b>	Only the 'Save' button will be activated in the screen where applicable
<b>New</b>	The 'Save' and 'New' buttons will be activated in the screen where applicable
<b>Delete</b>	The 'Save', 'New' and 'Delete' buttons will be activated in the screens where applicable
<b>Edit on VERIFIED data</b>	Access to change data with status VERIFIED
<b>Edit on APPROVED data</b>	Access to change data with status APPROVED

### New Objects

New objects can quickly be added to the application using the configuration screens. As soon as the new object has been saved to Energy Components it is immediately available in the appropriate screens with no requirement for additional programming.

### Audit Tracking

Energy Components automatically keeps a tracking journal of all data with a record status of 'VERIFIED' or higher. Again, this feature is configurable and can be changed to keep full information on all records with status of 'PROVISIONAL' or higher.

### Check Rules

Each screen can have one or more Check Rules or validation hints applied to it (as described in section 6.4 Data Validation) The Check Rules can be configured to automatically monitor the system for correctness of data and can prompt the user to correct data and re-run processes if necessary.

### Record Status Processes

System processes can be configured to run and update the record status automatically, or by manual intervention. These processes can be tailored to contain specific rules for each implementation project and customer. Access to these processes can be setup to prevent users from running unauthorised approval jobs for example from V (Verified) to A (Approved) and thereby prevent data from being accidentally changed by users.

### Language

The default language in Energy Components is English. However, the entire application can be configured to use the language of your choice. The language template is applied globally to Energy Components, but can also be configured and tailored for individual users so the system appears in their native language.

## Calculation Framework

The Calculation Framework in Energy Components is used to deploy client defined business logic applicable (in the form of process diagram) for the business processes. The client can create process diagram which may contain sub process diagram, equation process, excel workbook process and/or implemented calculation library process as they require. This unique feature allows specific process diagram to be defined for each operation, and applied in a consistent and controlled manner without the need for a compiler program. Each of the calculations is configurable - any changes can be added to the calculation process diagram which will then be applied consistently across the system.

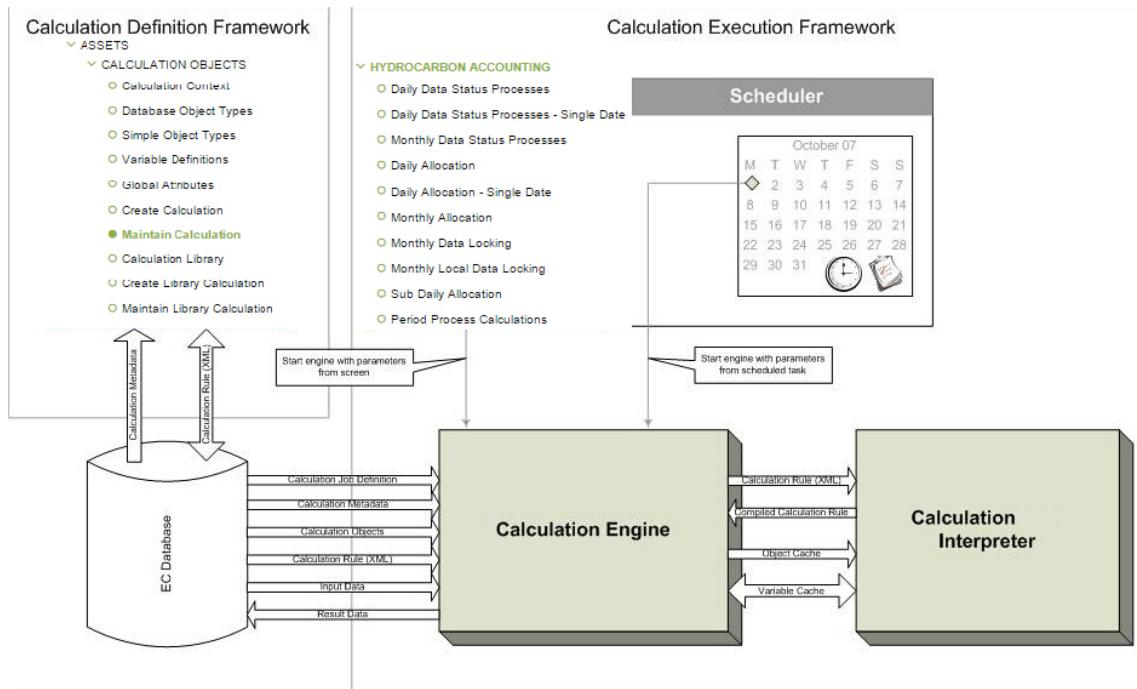
The framework allows the user to build up and/or modify complex process diagram containing sub process diagram, equation process, excel workbook process and/or implemented calculation library process which influences the existing calculations. This is particularly useful when new assets are being added onto the system, enabling the user to manage varying calculations and scenarios. The framework has functionality allowing the user to make complex changes in a test environment before being launched in the live system - removing the risk of incorrect calculations entering the live system.

The framework can be used across most business functions including:

- Hydrocarbon Accounting / Allocation calculations
- Sales / Contract calculations
- Price calculations
- Cargo Scheduling

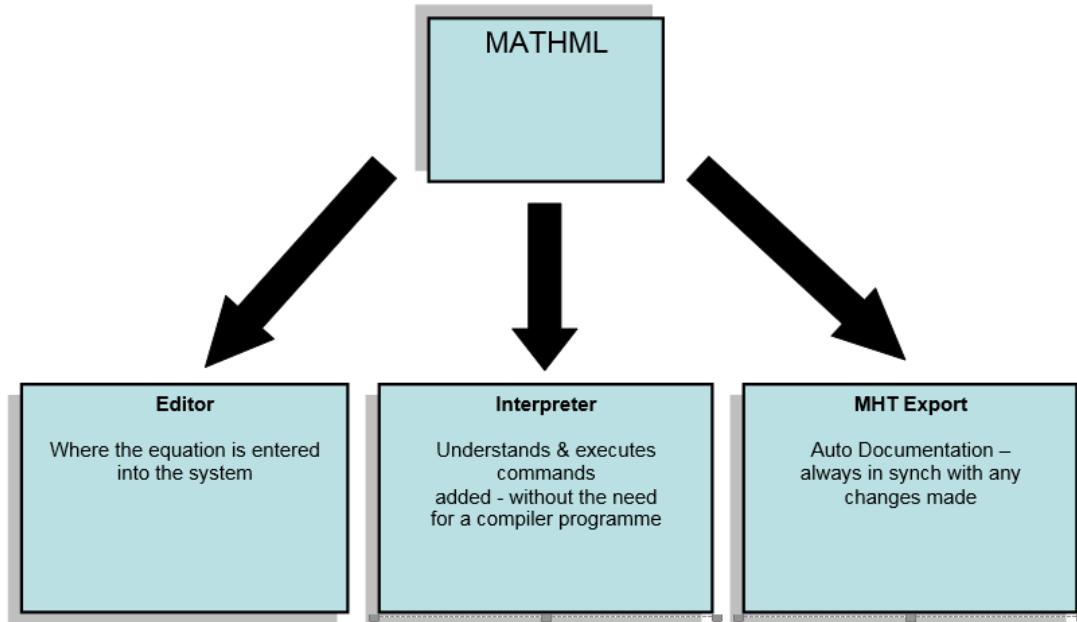
The Calculation Framework consists of two main components; the Calculation Definition Framework and the Calculation Execution Framework. The Calculation Definition Framework is used to create new and maintain existing calculation process diagram. The Calculation Execution Framework reads and understands the calculation put into the calculation editor and then executes it in the system.

The basic building elements for calculation process diagram are attributes and variables. They refer directly to values already stored in the Energy Components database. Equations, sets and conditions combine attributes and variables with arithmetic and logical operators and functions. The equations then arithmetically deploy the calculation process diagram, and the sets define the group of objects that the equation relates to.



## Equation Editor

Equations are stored in the Energy Components database in MATHML code. MATHML code which is a form of XML (Extensible Mark-Up Language) - code designed for mathematic expressions. As illustrated below, MATHML comprises an editor, an interpreter and a MHT export function.



Calculations (process diagram) are created using the Equation Editor, found on the Maintain Calculation screen (as shown below). In the top half of the screen the user selects the date, calculation context and calculation.

The screenshot shows the "Maintain Calculation" screen. On the left is a navigation menu with sections like "FAVORITES", "MENU", and "ASSETS". The main area has tabs for "FLOWCHART", "LOCAL SETS", "INHERITED SETS", "LOCAL VARIABLES", and "INHERITED VARIABLES". The "FLOWCHART" tab is selected, displaying a process diagram with nodes: "Start", "Allocate all production phases to PI and RBF", "Allocate all injection phases to PI and RBF", and "Stop". Arrows show the flow from Start to the first allocation node, then to the second, and finally to Stop. Below the diagram is a "RECORD STATUS" section with fields for "Created by", "Last updated by", "Record status", "Revision text", and "Revision Event".

**Maintain Calculation - Process Diagram**

The user then can build iterations, condition and equation etc. The definitions for the iterations, condition and equation

are also set by user.

Eqn #	Disable	Iterations	Condition	Equation
1	<input type="checkbox"/>	?	?	INFO = 'Allocate production phases for Resv Block Formation (' + Name_n + ')'
2	<input type="checkbox"/>	?	?	REMARK = 'Allocate Oil Production'
3	<input type="checkbox"/>	?	?	$RBF\_OP\_NStdVol_{n,d} = \sum_{w \in setWellsProducingOil} (RBF\_OP\_Vfrac_{n,w,d} * RBF\_OP\_InitialStdVol_{n,w,d})$
4	<input type="checkbox"/>	?	?	$RBF\_OP\_Mass_{n,d} = \sum_{w \in setWellsProducingOil} (RBF\_OP\_Vfrac_{n,w,d} * RBF\_OP\_InitialMass_{n,w,d})$
5	<input type="checkbox"/>	?	?	INFO = 'Allocated Oil to RBF (' + Name_n + ') is: Volume = ' + RBF_OP_NStdVol_{n,d} + ' Mass = ' + RBF_OP_Mass_{n,d}
6	<input type="checkbox"/>	pi ∈ setPerInterval	isValid(OP_Vfrac_{pi,d}) ∧ isVal	PerOP_NStdVol_{pi,d} = OP_Vfrac_{pi,d} * OP_InitialStdVol_{pi,d}
7	<input type="checkbox"/>	pi ∈ setPerInterval	isValid(OP_Vfrac_{pi,d}) ∧ isVal	PerOP_Mass_{pi,d} = OP_Vfrac_{pi,d} * OP_InitialMass_{pi,d}
8	<input type="checkbox"/>	pi ∈ setPerInterval	?	INFO = 'Allocated Oil to Perforation Interval (' + Name_pi + ') is: Volume = ' + PerOP_NStdVol_{pi,d} + ' Mass = ' + PerOP_Mass_{pi,d}
9	<input type="checkbox"/>	?	?	REMARK = 'Allocate Gas Production'
10	<input type="checkbox"/>	?	?	$RBF\_GP\_NStdVol_{n,d} = \sum_{w \in setWellsProducingGas} (RBF\_GP\_Vfrac_{n,w,d} * RBF\_GP\_InitialStdVol_{n,w,d})$
11	<input type="checkbox"/>	?	?	$RBF\_GP\_Mass_{n,d} = \sum_{w \in setWellsProducingGasMass} (RBF\_GP\_Vfrac_{n,w,d} * RBF\_GP\_InitialMass_{n,w,d})$
12	<input type="checkbox"/>	?	?	INFO = 'Allocated Gas to RBF (' + Name_n + ') is: Volume = ' + RBF_GPNStdVol_{n,d} + ' Mass = ' + RBF_GPMass_{n,d}
13	<input type="checkbox"/>	pi ∈ setPerInterval	isValid(GP_Vfrac_{pi,d}) ∧ isVal	PerGP_NStdVol_{pi,d} = GP_Vfrac_{pi,d} * GP_InitialNStdVol_{pi,d}
14	<input type="checkbox"/>	pi ∈ setPerInterval	?	PerGP_Mass_{pi,d} = GP_Vfrac_{pi,d} * GP_InitialMass_{pi,d}

RECORD STATUS: REVISION INFO: APPROVAL STATUS: HINTS & TIPS: VALIDATION: TRENDING  
Created by: ECKERNEL\_111\_TC4\_12EE | 2015-12-05T17:16:39 | Record status: Provisional, 0 revision(s).  
Last updated by: ECKERNEL\_111\_TC4\_12EE | 2015-12-05T17:16:42 | Revision text: Revision Event: [dropdown]

## Maintain Calculation - Equation Process

Calculations can be changed and are available immediately for execution without the need for programming or compiling code. The Equation Editor is a visual system which uses a series of configurable process diagram such as equation process conforming to the EC Calculation Syntax. A basic equation in the EC Calculation Syntax would have the variable or object components on the left hand side, and the arithmetic expression on the right hand side, as shown in the example given below in Equation Syntax 1:

**Equation Syntax 1:**  $variable_{lhs-object-ident-list} = arithmetic\_expression$

An equation can also have an associated condition in the form of a logical expression, as shown in the example given below in Equation Syntax 2:

**Equation Syntax 2:**  $logical\_expression \quad variable_{lhs-object-ident-list} = arithmetic\_expression$

The equation given below sets the allocated daily net volume for all incoming oil streams (that are not fixed), equal to the measured daily stream volume, times the daily oil reconciliation factor:

$$DayNStdVol_{\text{setInAdjusOil}, d \in Days} = DayInitialNStdVol_{\text{set}, d} \cdot \frac{\sum_{l \in strOutOil} DayNStdVol_{l,d} - \sum_{l \in strInFixedOil} DayNStdVol_{l,d}}{\sum_{l \in strInAdjusOil} DayNStdVol_{l,d}}$$

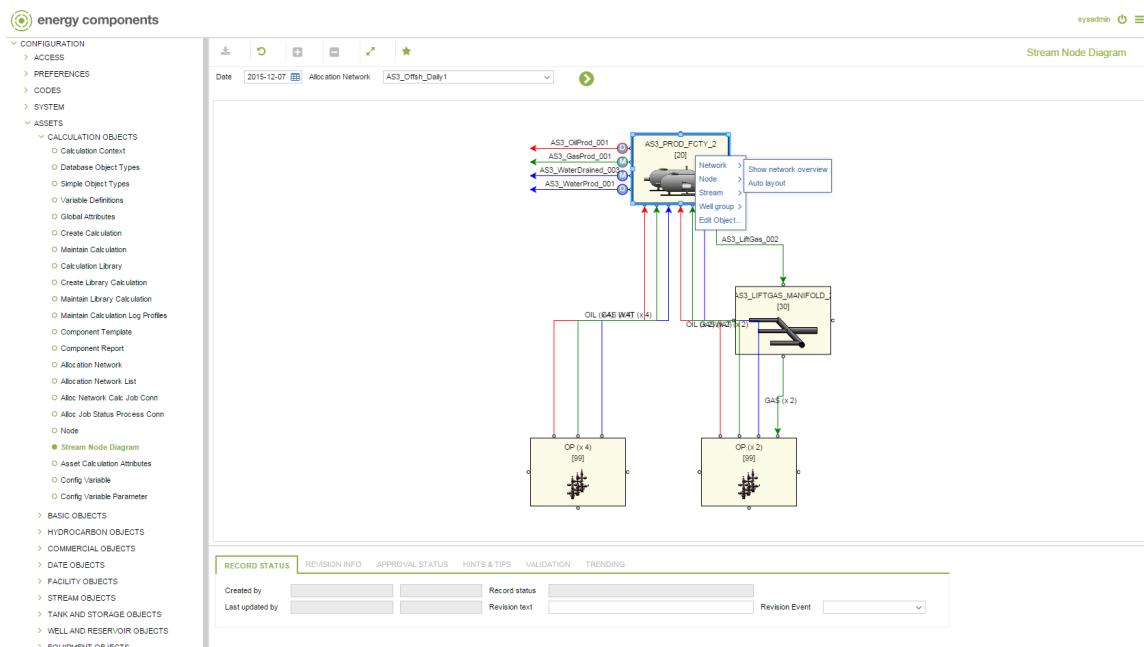
## The Stream Node Diagram

Within Energy Components, a Stream represents a flow from one point to another. An example would be a pipeline transporting gas. The points connecting a Stream are called Nodes. Examples of Nodes would be well, gathering station, platform, terminal. The main purpose of the Stream / Node concept is to model infrastructure and hydrocarbon flow. Further, to have a structure for applying quantities and calculations. The quantities are related to the Streams, e.g. a gas stream holding a quantity for the mass flowing for a certain period. The calculations are related to the Nodes, e.g. a calculation summarising the quantities for two incoming gas streams and putting the result on the outgoing gas stream.

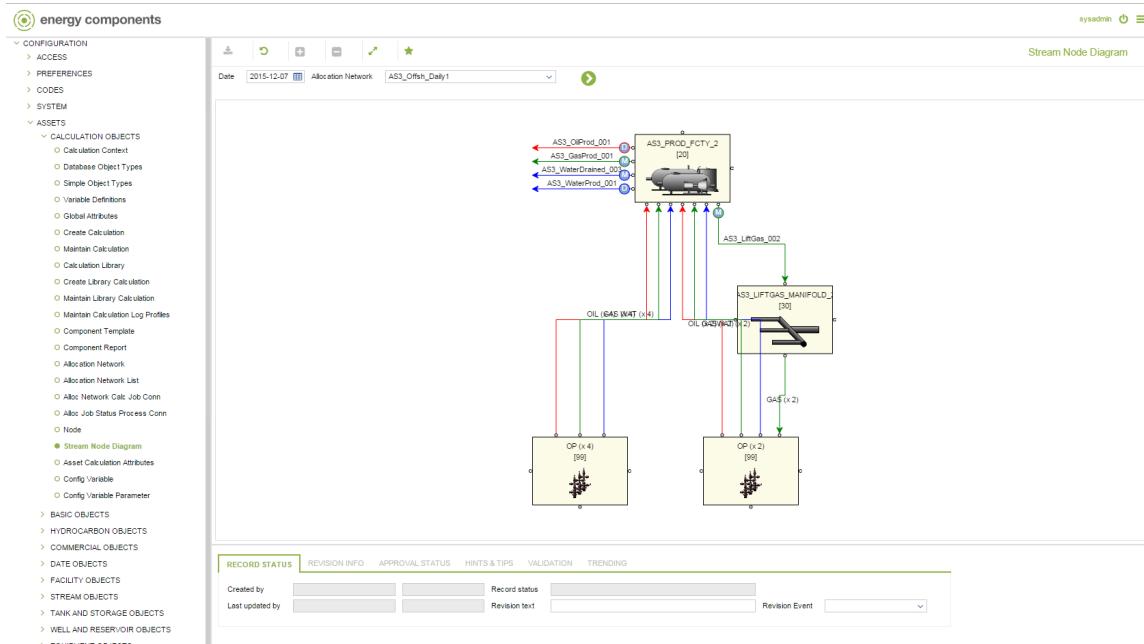
The Stream Node Diagram is used to show the structure of the production of oil and gas through pipelines and terminals. The visual representation is useful in situations where there is a large network, incorporating many streams and nodes (or



network points and delivery points). The Stream Node Diagram is configurable using the Visualization Tool that's provided alongside Energy Components. This means that any changes to the diagram can be made using the configuration screens and can be implemented immediately without the need for programming or system downtime. There are many functions can be chosen from the context menu of Stream Node Diagram: Auto layout, Edit Object etc.



### Stream Node Diagram Editor - Context Menu



### Example of a Stream Node Diagram

new

## Classes and Objects

The EC system provides a large degree of flexibility to cater for different business scenarios. To be able to do this, the system has been designed with an abstraction view layer in the database that separates the business logic from the actual table structure used to store the data. This abstraction is defined as a set of what we call classes - very similar to the class concept used in object oriented design and programming languages.

This concept allows us to:

- Have common functionality for data validation and integrity constraint without mixing it with the business logic
- Choose to make some class attributes virtual (so that they are calculated when needed), or physically store them if needed without changing the table structure
- Hide or show product attributes and also add their own class attributes if needed
- Adjust and define the screen navigation model to be suitable for both small and larger operations
- Enforce data protection with ringfencing and data locking
- Support workflow operations, such as four eye approval and control point validation, as generic concepts that can be added to any class
- Replicate some data for performance reasons without having the business layer writing special code for it
- Change the table structure without interfering with the business logic

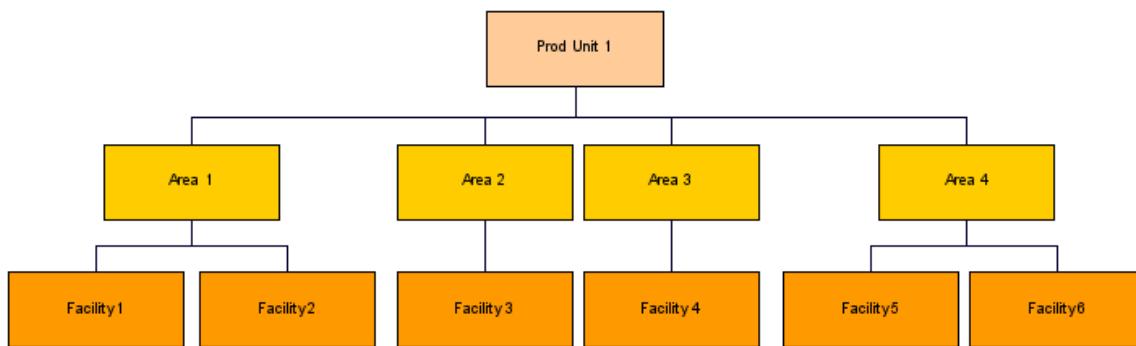
The class concept distinguishes between four different class types:

- **Object classes**
  - Object classes are usually something static such as a physical object.
  - Typical Object classes are for example: Facility, Tank, Separator and Well.
- **Data classes**
  - Data classes are typically owned by an object class, and represent a set of measurements or events linked to the owner object.
  - Daily tank volume readings and exported oil and gas volumes are very often part of a Data class.
- **Interface classes**
  - Interface classes are abstractions over several object classes with a common subset of attributes.
  - They are used, among other things, for representing specific physical objects as nodes in a diagram showing how the hydrocarbons are passing through the system.
- **Table classes**
  - Table classes are similar to data classes, but have less support for validation, row level security, etc.
  - A table class typically does not have an object owner or a timestamp as a part of the primary key.

## Group model

Information from Group Model is used to populate the navigators on any screens that are configured to use the group model and therefore control what is viewed on the screen. Each different navigator represents a different Group Model. The Group Model is a hierarchical organisational structure of assets and classes within Energy Components, which again can be configured to meet the needs of the client.

There are parent and child relationships between objects. In the example below, the parent is the Production Unit; next in the structure is the area or field, and finally the child in the structure is the Production Facility i.e. the individual installation. There can only be one 'Parent' in a Group Model but there can be several interconnecting 'Child' relationships. The example below shows a full Group Model, where all areas in the structure link to the production unit.



In the example below the navigator is looking for information on a particular offshore facility. The structure of the Group Model used to populate this navigator is very simple. It contains the Production Unit at the top (which is the Parent) and the Offshore Area (which is a child object) and finally the specific Offshore Facility (which is also a child object). There are two predefined Group Models within Energy Components - Geographical and Operational. All other Group Models can be configured by the client to meet their requirements. The Group Model configuration screen is found in the configuration menu, under 'System'. You can select the object and classes you wish to appear in the model and the navigator and you can decide if fields in the navigator are mandatory or optional.

Group Model

Group Type: operational

Class Name	Parent Class Name	Is it Role Name	Label	Context	Mandatory
AREA	PRODUCTIONUNIT	<input checked="" type="checkbox"/>	OP_PRODUCTIONUNIT	Op Production Unit	EC_PROD
AREA	PROD_SUB_UNIT	<input checked="" type="checkbox"/>	OP_PROD_SUB_UNIT	Op Prod Sub Unit	EC_PROD
CHEM_INJ_POINT	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
CHEM_TANK	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
CHOKE_MODEL	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
EOPR	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
FCTY_CLASS_1	AREA	<input checked="" type="checkbox"/>	OP_AREA	Op Area	EC_PROD
FCTY_CLASS_1	FCTY_CLASS_2	<input checked="" type="checkbox"/>	OP_FCTY_CLASS_2	Op Facility Class 2	EC_PROD
FCTY_CLASS_2	SUB_AREA	<input checked="" type="checkbox"/>	OP_SUB_AREA	Op Sub Area	EC_PROD
FLOWLINE	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
SHIFT	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
STORAGE	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
STREAM	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
PIPELINE	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
PROSEPARATOR	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
PROD_SUB_UNIT	PRODUCTIONUNIT	<input checked="" type="checkbox"/>	OP_PRODUCTIONUNIT	Op Production Unit	EC_PROD
SUB_AREA	AREA	<input checked="" type="checkbox"/>	OP_AREA	Op Area	EC_PROD
TESTSEPARATOR	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
TEST_DEVICE	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD
WELL_HOLE	FCTY_CLASS_1	<input checked="" type="checkbox"/>	OP_FCTY_1	Op Facility Class 1	EC_PROD

(1 of 2) [1] [2] [3]

**DB MAPPING**

Mapping Type: ATTRIBUTE	PRESENTATION
Label: OP_PU_ID	Op Production Unit
Syntax: 2000	Sort Order: 100
Sort Order: 2000	Static Presentation:

```

viewwidth=120 PopupURL=FrontController!com_ecm_screens_object_popup?
CLASS_NAME=PRODUCTIONUNIT.PopupDependency@RetrieveAll@DATIMESCREEN.this.currentRow.DATIMESCREEN
OBJECT_START_DATE=Screen.this.currentRow.OBJECT_START_DATE@SCREEN.this.currentRow.OP_PRODUCTIONUN
T_ID=ReturnFields.OBJECT_ID@PopupReturnColumn=2@PopupWidth=250@PopupHeight=300
  
```

Generic Presentation

Group Model Configuration Screen

## User roles and Access Rights

User roles and access rights are configured in Energy Components to control or limit which screens a user can view and control the users' ability to insert, update or delete data. These controls are enabled by assigning one or more roles to a user, then defining the level of access they will have within any screen. User roles are found on the Tree View menu, under configuration and access.

### User Roles

User roles define the activities a particular business role can access – e.g. well tests, configuration etc. They also determine the level of access – e.g.: insert and update well tests, view well configurations, hide allocation configuration screens, etc.

### Partitioning

Partitioning defines the fields and facilities a role can have access to. For example, an operator role for "platform A" can be restricted to only entering or viewing data for platform A.

### Users

This is where individual users are created in the system and their access roles (one or more) allocated, e.g. Fred Jones OPT/32 – "Data Entry role for Facility B".

## Reporting

Energy Components contains a framework for reporting, supporting both product specific- and customer specific reports. The framework is designed to work with both internal reporting capabilities as well as 3rd party report tools.

The EC Products offer standard reports delivered with the product (e.g. Cargo documentation). Additional packages including reports for specific operations- and areas (e.g. Regulatory Reporting) can be installed in addition.

Customers can create their own reports, both ad-hoc- and fixed reports. Most of the configuration is done through screens in EC ('Reporting' in the Tree View Menu).

### Fixed Reporting

A fixed report will be based on a report definition. The report definition will include one- or several report templates, and can have additional configuration. The report templates will tell what 'system' to use for generating the report, and what input parameters are needed. The report template represent one report artifact (pdf, excel etc.).

EC offers the following report systems:

EC Jasper Report	This system supports JasperReporting 6.0. A jasper report can be exported to several formats, like PDF, Excel, CSV, etc.
EC Excel Report	EC has a system for configuration of Excel reports, with mapping of data between spreadsheet and the database. It can be used together with the calculation engine, as part of a calculation.
External System	This system enables EC to handle reports generated by external systems.
EC Internal	This system is deprecated, and will be removed in future versions of EC. It supports an old version of Jasper (version 3.1). The system is also used for plugging in customer specific solutions for report generation. Note: The system is by default set to inactive (The EC Code REPORT_SYSTEM).

Customer specific integration of any 3rd party reporting tool can be implemented. The only requirement is that the tool can connect to the oracle database, and that there are a way for the system to interact with the tool. The configuration and generation of these reports will use the same screens, as for fixed reports.

Note that Tieto offers an out-of-the-box solution for integration with Business Objects (BO-Integration).

### Ad-hoc Reporting

For ad-hoc reporting, the screen 'Export to Excel Express' can be utilized. End-user can create their own reports using screens in EC. Any database view generated from an EC class can be used as basis for a report. Data can be filtered, and the user can select what columns to include.

### Workflow of Reporting

Configuration	Reports are configured through screens in EC. These are found under 'Reporting' in the Tree View. External tools like Excel and JasperReport tool need to be used to create the report templates. The reports can be added to report sets.
Generation	Reports can be generated through the 'Report Administration' screen. Some product screens will offer report generation as well (e.g. cargo documents). Scheduled tasks can be created for reports to be generated automatically at some time and frequency. Generation can also be added to a business process (BPM).
Search and View	Reports can be viewed/downloaded from 'Report Administration' screen. The Report Archive screen provide search by parameter value.
Distribute	Reports can be connected to messages configured in the EC Messaging System. By clicking the 'Send' button for a generated report, the report can e.g. be sent as email internally or to partners and government systems.
Publish	Generated reports can be registered as published, so that they appear as published reports for a specific period. The 'Display Published Reports' screen will show the published reports.
Verify and Approve	A generated report will initially have the status Provisional. It can then be set to Verified and Approved. Users will need to have specific access to be able to approve reports.

## Energy Components Technical Documentation

The Technical Documentation is separated per module, providing information that describes handling, functionality and architecture that is relevant for each specific module.

- EC Production
- EC Sales
- EC Transport
- EC Revenue
- EC Framework

## EC Production

This chapter contains the Technical Documentation for EC Production

- EC Production System Attributes
- EC Reporting Layer
- EC Production Dashboard

## EC Production System Attributes

EC Production uses a number of System Attributes to control system behavior. This document explains these system attributes.

For a new EC installation all of these should be checked and set according to the required behavior of EC Production.

### **System Attribute 'ADJUST\_POTENTIAL\_DST'**

This attribute controls whether the well potential will be adjusted for daylight saving dates. "Y" means that the well potential will be adjusted. Used in EcBp\_Well\_Potential.

Options are Y | N.

The Default EC installation value is 'Y'.

### **System Attribute 'ALLOW\_ALLOC\_LOCK\_MONTH'**

This attribute indicates whether monthly allocation is allowed on locked months. "Y" means that monthly allocation is allowed on locked months.

Options are Y | N.

The Default EC installation value is 'N'.

### **System Attribute 'ALLOW\_DUP\_PROD\_DEF\_EVENT'**

This attribute indicates whether overlapping non-well group deferment events will be allowed. This applies to the obsolete deferment version Low and Off Deferments.

Options are Y | N.

The Default EC installation value is 'N'.

### **System Attribute 'ALLOW\_WT\_MULTISELECT'**

This attribute indicates whether it is allowed to select multiple well tests at a time in BF PT.0013, Single Production Well Test. When this attribute is set to 'Y' multiple well tests can be selected and Accept, Reject, Preprocess or Calculate PVT procedures can be executed.

Options are Y | N.

Default EC installation value is 'N'.

### **System Attribute 'API\_CALC\_BIT\_DENSITY\_MAX'**

This attribute indicates the default maximum bitumen density.

Default EC installation value is 1050

### **System Attribute 'API\_CALC\_BIT\_DENSITY\_MIN'**

This attribute indicates the default minimum bitumen density.

Default EC installation value is 950

### **System Attribute 'API\_CALC\_DIL\_DENSITY\_MAX'**

This attribute indicates the default maximum diluent density.

Default EC installation value is 750

### **System Attribute 'API\_CALC\_DIL\_DENSITY\_MIN'**

This attribute indicates the default minimum diluent density.

Default EC installation value is 500

### **System Attribute 'API\_CALC\_MAX\_ITERATIONS'**

This attribute indicates the default max iteration used in the loop to calculate blend/diluent volume in EcBp\_Vcf.calcAPIBlendShrinkage.

Default EC installation value is 20.

#### **System Attribute 'API\_CALC\_TOLERANCE'**

This attribute indicates the default tolerance value used in the loop to calculate blend/diluent volume in EcBp\_Vcf.calcAPIBlendShrinkage.

Default EC installation value is 1E-6

#### **System Attribute 'BITUMEN\_DENSITY'**

Default value for Bitumen Density Default EC installation value is 1020.

#### **System Attribute 'DAILY\_DEFERMENT\_LEVEL'**

This attribute indicates the level in the operational group model events are stored against. This is applicable for the PD.0004 Daily Deferment, deferment version.

Valid choices : SUB\_AREA, FCTY\_CLASS\_2, FCTY\_CLASS\_1.

Default EC installation value is 'FCTY\_CLASS\_1'.

#### **System Attribute 'DECIMAL\_COMP\_NORMALIZE'**

This attribute indicates the number of decimals to be stored when component normalization to 100 is performed.

Default EC installation value is NULL (Blank)

#### **System Attribute 'DEFERMENT\_VERSION'**

This attribute indicates the deferment version to use. Note that with the introduction of the new deferment version PD.0020 (Well Deferment), PD.0001 / PD.0001.02 and PD.0006 are now obsolete and will not be supported in future EC versions.

Choose between PD.0001 / PD.0001.02 / PD.0004 / PD.0006 / PD.0020.

Default EC installation value is 'PD.0020'.

#### **System Attribute 'DEF\_WELL\_CALC\_RULE'**

This attribute is obsolete

#### **System Attribute 'DILUENT\_DENSITY'**

This attribute indicates the default value for Diluent Density.

Default EC installation value is 800.

#### **System Attribute 'MAX\_ALLOC\_MONTHS'**

This attribute indicates the maximum number of months the Monthly Allocation BF can process at a time.

Default EC installation value is 1.

#### **System Attribute 'MAX\_SHORT\_DAYS'**

This attribute indicates the number of days before a SHORT deferment event is moved to LONG. This is only applicable for the Low/Off Deferment version which is now obsolete.

Default EC installation value is '1000'.

#### **System Attribute 'MONTHLY\_STATUS\_PROMPT'**

This attribute indicates whether the user will be prompted before proceeding on the non-reversible task Monthly Data Status Processes in Hydrocarbon Accounting.

Options are Y | N.

Default EC installation value is 'N'.

#### **System Attribute 'MOVE\_TO\_LONG\_TERM'**

This attribute indicates the behavior of the Move to Long term button in the obsolete BF's PD.0001.02 and PD.0002.02.

Options are COPY\_MOVE | MOVE\_ONLY.

Default EC installation value is 'MOVE\_ONLY'.

#### **System Attribute 'NBR\_TRUCK\_LOADS\_AVG\_BSW'**

This attribute indicates the number of truck loads used to calculate the average bsw from well tanks.

Default EC installation value is 10.

#### **System Attribute 'ON\_STRM\_DAYS\_METHOD'**

This attribute indicates how on stream days will be calculated in Monthly Gas Injection Well Data (WR.0046) and Monthly Water Injection Well Data (WR.0047).

If HRS\_DIV\_24, then calls CalcOnStrmHrsMonth() / 24

If DAYS\_GREAT\_0, then count number of days where inj\_vol>0

Options are HRS\_DIV\_24, |DAYS\_GREAT\_0

Default EC installation value is 'HRS\_DIV\_24'

#### **System Attribute 'PD.0001/2\_HIDE\_REPORT'**

This attribute is obsolete.

#### **System Attribute 'PERF\_CURVE\_RATE\_AXIS'**

This attribute indicates which axis should display the performance curve rate (X or Y).

Options are X | Y

Default EC installation value is 'X'

#### **System Attribute 'PROD\_FCST\_SCENARIO'**

This attribute indicates the system default production forecast scenario

Default EC installation value is 'OFFICIAL'

#### **System Attribute 'PROD\_FCST\_TYPE'**

This attribute indicates the system default production forecast type

Default EC installation value is 'SHORT\_TERM'

#### **System Attribute 'PROD\_FCST\_PERIOD'**

This attribute indicates the system default production forecast period.

#### **System Attribute 'PT.0013\_FORMULA'**

This attribute indicates which shrinkage calculation to use in PT.0013 Single Production Well Test

Options are PT.0013 | PT.0010

PT.0013 -> New formula, PT.0013 -> use old formula from PT.0010.

Default EC installation value is 'PT.0010'

#### **System Attribute 'PT\_COMBINE\_TIMESCOPE'**

This attribute indicates the maximum number of days of result history that should be available in the production test combine context

Default EC installation value is 90

#### **System Attribute 'PT\_RESULT\_TIMESCOPE'**

This attribute indicates the maximum number of days of result history that should be available in the production test result context

Default EC installation value is 10

#### **System Attribute 'REF\_AIR\_PRESS'**

This attribute indicates the reference value used to calculate air density. The unit is mBar.

Default EC installation value is 1013.25

#### **System Attribute 'REF\_AIR\_TEMP'**

This attribute indicates the reference value used to calculate air density. The unit is Celsius.

Default EC installation value is 15

#### **System Attribute 'REF\_PRESS\_BASE'**

This attribute indicates the default pressure base in BAR

Default EC installation value is 1.01325.

#### **System Attribute 'REF\_REL\_HUMIDITY'**

This attribute indicates the reference value used to calculate air density. The unit is %

Default EC installation value is 0.

#### **System Attribute 'REF\_SITE\_ATM\_PRESS'**

This attribute indicates the default site atm. pressure in BAR

Default EC installation value is 1.01325.

#### **System Attribute 'REF\_TEMP\_BASE'**

This attribute indicates the default temperature base in C.

Default EC installation value is 15.

#### **System Attribute 'REF\_WATER\_TEMP'**

This attribute indicates the reference value used to calculate water density. The unit is Celsius.

Default EC installation value is 15.

#### **System Attribute 'STANDARD\_CODE'**

This attribute indicates the system standard.

Default EC installation value is 'ISO6976\_STD'

#### **System Attribute 'VF\_ROUTINE\_TAB'**

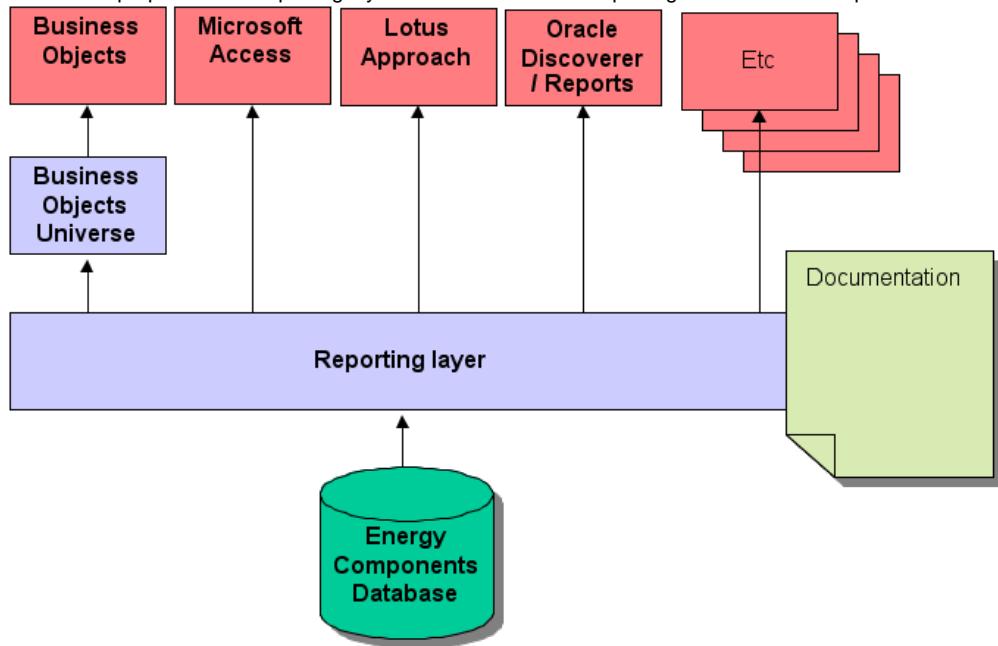
This attribute indicates whether to disable the Generated Vapors and Recovered Vapors tab in PO.0086 Daily Gas Stream- vent and Flare Calc. If set to Y, these two tabs will be not be activated in the screen for Routine streams.

## EC Reporting Layer

This document gives a high-level description of the generated reporting layer shipped with Energy Components

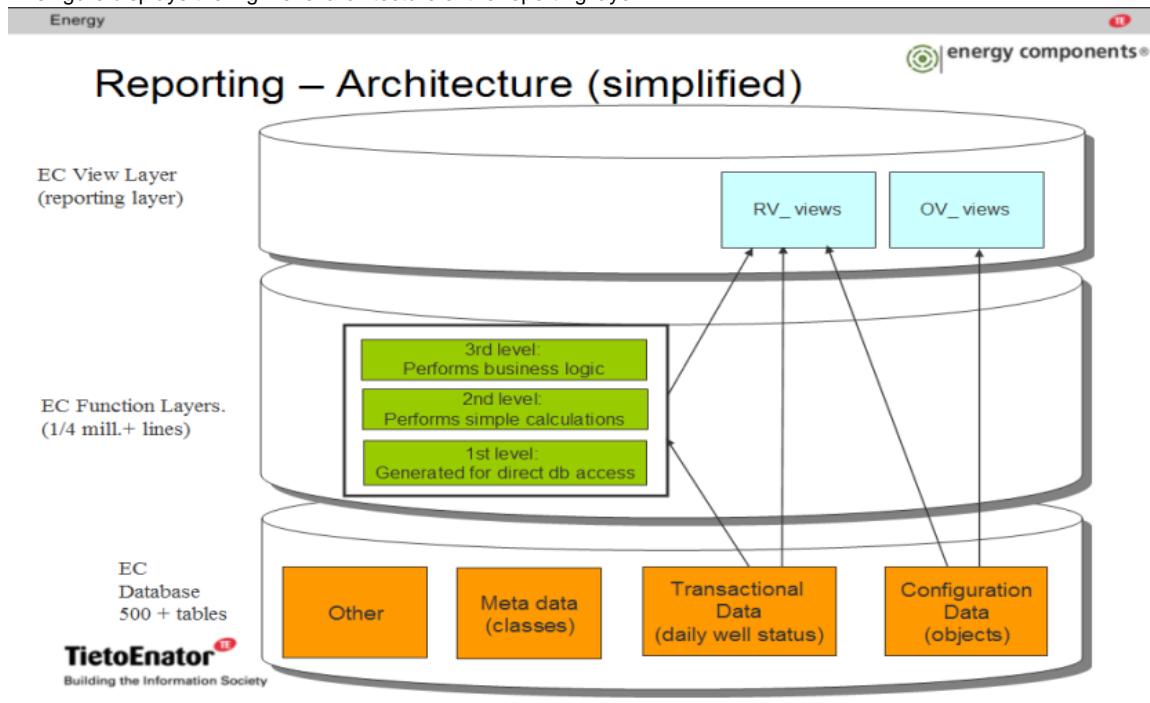
### Overview

The overall purpose of the reporting layer is to make end user reporting much easier and quicker.



### Architecture

This figure displays the high level architecture of the reporting layer.



## Reporting views (RV views)

There will be generated one RV view for each data class, object class and table class configured in EC. The RV view will include:

- All attributes from the owner class (object class)
- All attributes from the data class
- Reporting views always shows correct configuration for the selected production day(s)
- All unit conversions configured (see screen shot)
- A dedicated REPORTING\_<operation> Oracle database user is created. This user id has read access to all RV views.

Measurement Type	Description	Format Mask
STD_LIQ_VOL_RATE_	Standard liquid volume rate per month	###.###.##0.0
STD_LNG_DENS	Standard LNG Density	#.##0.00
STD_LNG_VOL	Standard LNG Volume	###.###.##0.0
STD_OIL_DENS	Standard oil density	#.##0.00
STD_OIL_RATE	Standard oil rate per day	###.###.##0.0
STD_OIL_SG	Specific Gravity	
STD_OIL_VOL	Standard oil volume	###.###.##0.0
STD_WATER_DENS	Standard water Density	#.##0.00
STD_WATER_RATE	Standard water rate per day	###.###.##0.0
STD_WATER_VOL	Standard water Volume	###.###.##0.0
STEAM_RATE_CWE	Cold Water Equivalents Steam Rate	###.###.##0.0
STEAM_VOL_CWE	Cold Water Equivalents of Steam	###.###.##0.0
STROKE_LENGTH	Beam Pump stroke length	#.##0.00
STROKE_SPEED	Beam Pump stroke speed	
TEMP	Temperature	#.##0.0
THERMAL_EXPANSIO	Thermal Expansion	0.000000
TOTAL_ACID_NO	Total Acid Number (ph)	
ULLAGE	Ullage	#0.00
VIBRATION	Vibration	
VISCOSITY_KIN	Kinematic viscosity	#.##0.0000

Any temperature attributes will get two columns in the reporting view. One for C (Celsius) and one for F (Fahrenheit). The columns will get a \_C and \_F postfix to the attribute name. The end who creates the reports does not need to know which unit is stored in the database, the reporting layer will automatically perform conversion between units.

Reporting views (RV\_) are not updatable

## Other generated views

EC will also generate data views, object views, interface views and table views.

### Data views (DV\_)

Data views are 100% generated and they are updatable. There is one or several data views for each business function in EC. The web client uses data views for select, insert, update and delete. Data views does not contain unit of measure conversion or information (no \_C postfix). End

should not use data views for reporting, all information in data views are also available in reporting (RV) views.

### Object views (OV\_)

Object views are 100% generated, and they are updateable. There is one object view for each object class in EC. The web client uses object views for select, insert, update and delete of objects (maintain screens). Object views only lists records where there has been a change in the configuration.

End users can use object views if the purpose is to get information about configuration changes. Object views should not be used if the purpose is to get configuration for a selected production day, for this purpose RV views should be used.

### Table views (TV\_)

Table views are 100% generated and they are updatable. Some configuration screens uses table classes (e.g. EC

Codes). A table class hold data that cannot be modelled as object or data classes.

End users should not use table views, all information in table views are also found in reporting (RV) views.

### **Interface views (IV\_)**

Interface views are 100% generated and they join objects of same type. The interface view IV\_FACILITY joins objects from OV\_FCTY\_CLASS\_1 and OV\_FCTY\_CLASS\_2.

End users can use interface views.

### **Journal views (\_JN)**

Journal views lists old records (being either deleted or updated). These views are read only.

### **Other views**

There are also some views that are maintained manually. These views are named V\_<view name>. Some views are also named DAO\_<view name> (internal EC usage) or TRANS\_<view name>, these are also maintain manually.

### **Customer specific views (ZV\_)**

If there is a need to write additional views to meet specific customer requirements, they should always be prefixed ZV\_<view name>. It will be easier to upgrade to a newer version of EC if all custom specific code easily can be identified.

## **Query guidelines**

Reporting views can potentially return a large amount of records and columns from the database. The number will depend upon how many objects that are defined in the database, and how many years of history data are available.

As a general rule always

- List which columns you want to extract from the view (avoid "select \* from").
- Add where condition for production day
  - Where production\_day between.....
  - Where production\_day = .....
- Add where condition for objects to extract if not all objects are of interest
  - Where op\_fcty\_1\_code = 'AAAA' (e.g. get all objects which are connected to operational facility 1 code 'AAAA' only)
  - Where .....

## **View listing**

The sql below can be used to list all DATA, OBJECT and TABLE reporting views

```
Select class_type, 'RV_'||class_name, label from class where class_type in ('DATA','OBJECT','TABLE') order by class_type, class_name
```

## EC Production Dashboard

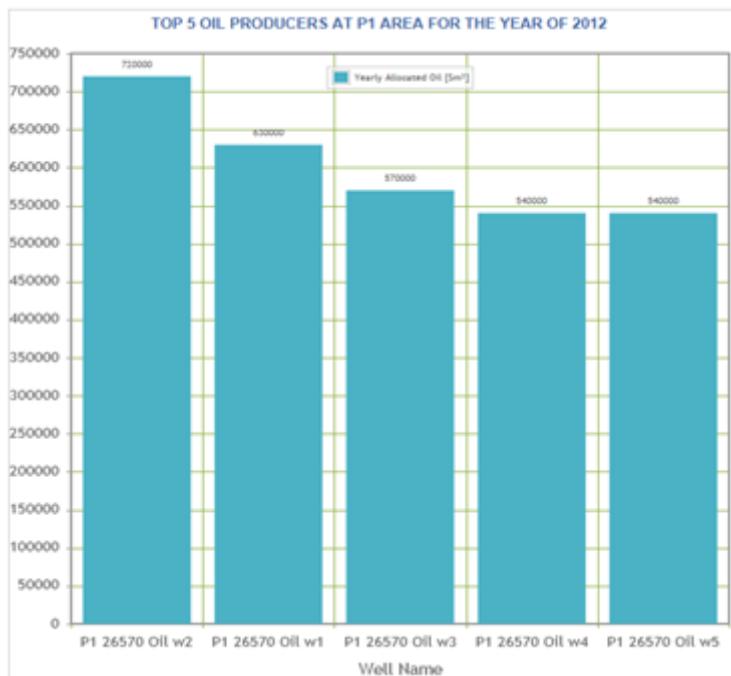
This document describes the EC Production Dashboard widgets that comes as part of the standard EC product. The EC Production Dashboard consists of two line widgets, two bar widgets and two pie chart widgets. They are predefined. However, it is possible for projects to create or modify the dashboard query in order to support their business needs. The query is stored in the QUERY parameter in CTRL\_DASHBOARD\_PARAM table. To define these widgets from the dashboard, there is a menu consisting of different input parameters on the top right hand corner of the widget.

Production widgets that are available in EC-11.1 are:

- Top 5 Yearly Producers at Area (separate widgets for Oil/Gas/Condensate)
- Top 5 Yearly Producers at Facility (separate widgets for Oil/Gas/Condensate)
- Top 5 Monthly Producers at Area (separate widgets for Oil/Gas/Condensate)
- Top 5 Monthly Producers at Facility (separate widgets for Oil/Gas/Condensate)
- Daily Actual Vs Planned For Area (separate widgets for Oil/Gas/Condensate/Water)
- Daily Actual Vs Planned For Facility (separate widgets for Oil/Gas/Condensate/Water)
- Monthly Actual Vs Planned For Area (separate widgets for Oil/Gas/Condensate/Water)
- Monthly Actual Vs Planned For Facility (separate widgets for Oil/Gas/Condensate/Water)
- Oil Production For Area
- Oil Production For Facility
- Gas Utilization For Area
- Gas Utilization For Facility

### Widget 1: Top 5 Yearly Oil Producers at Area:

This dashboard widget displays top 5 Oil Producers at a selected Area and Year. In the example below, it displays the Top 5 Oil Producers at P1 Area in 2012.



The parameters that needs to be configured for Top 5 Yearly Oil Producers at Area is shown in the screenshot below:



This widget can be filtered by Year and Area.

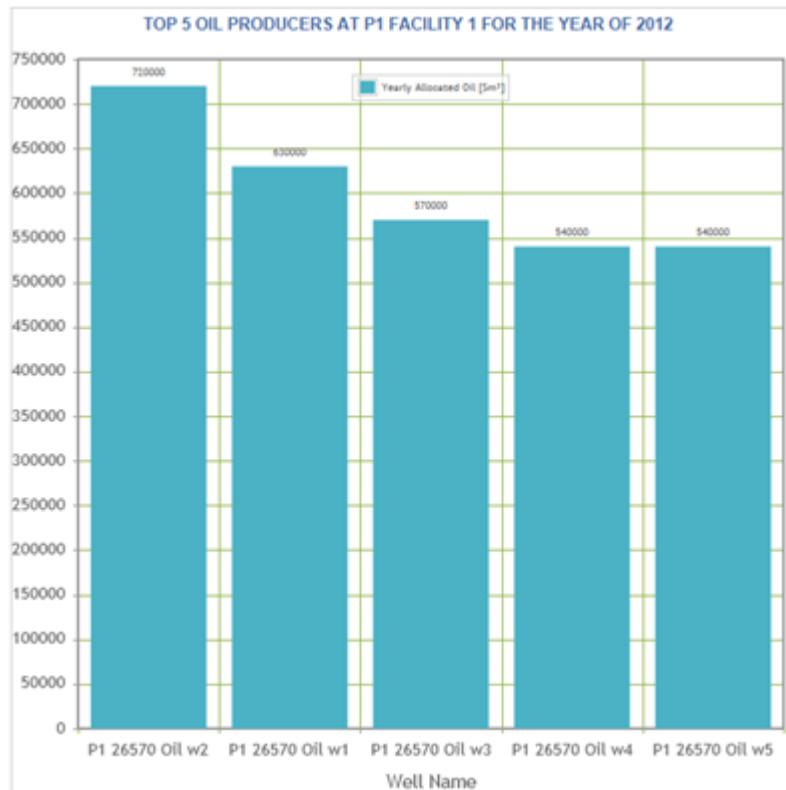
The function of each chart parameters is defined below:

- Legend position: The position of the legend in the chart.
- Show Point Labels: Each series will be labeled with the produced Oil volume value.

Similar widgets and configuration exists for Gas and Condensate.

#### Widget 2: Top 5 Yearly Oil Producers at Facility:

This dashboard widget displays top 5 Oil Producers at a selected Facility and Year. In the example below, it displays the Top 5 Oil Producer at P1 Facility in 2012.



The parameters that needs to be configured for Top 5 Yearly Oil Producers at Facility is shown in the screenshot below:



This widget can be filtered by Year and Facility.

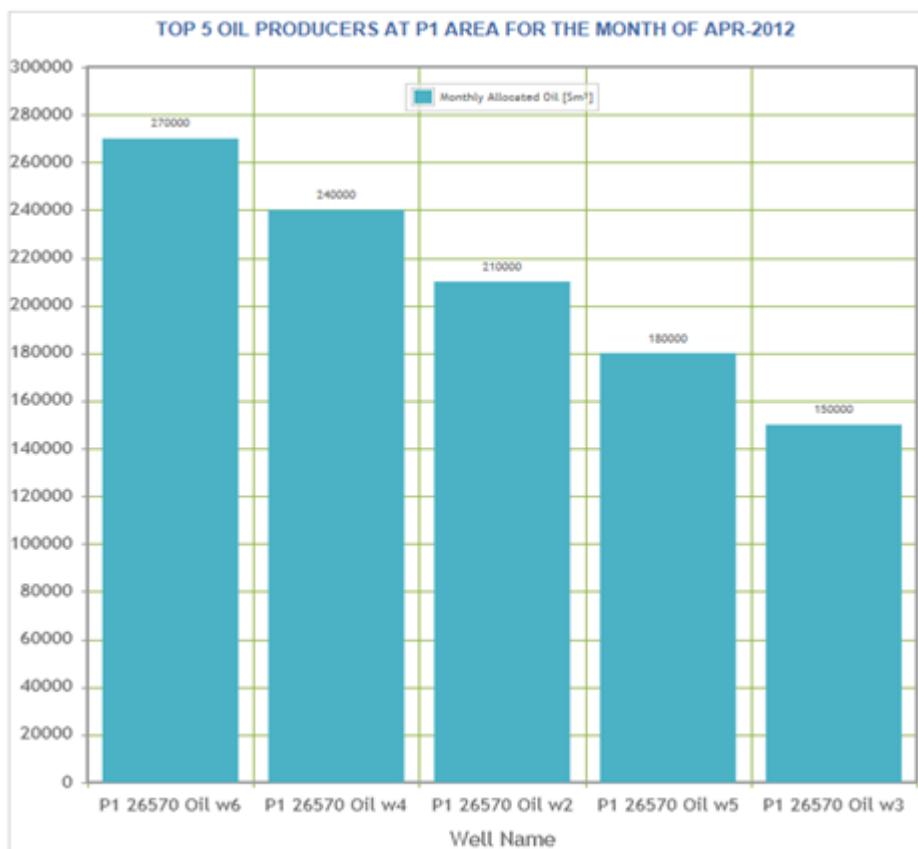
The function of each chart parameters is defined below:

- Legend position: The position of the legend in the chart.
- Show Point Labels: Each series will be labeled with the produced Oil volume value.

Similar widgets and configuration exists for Gas and Condensate.

#### Widget 3: Top 5 Monthly Oil Producers at Area:

This dashboard widget displays top 5 Oil Producers at a selected Area and Month. In the example below, it displays the Top 5 Oil Producers at P1 Area for April, 2012.



The parameters that needs to be configured for Top 5 Monthly Oil Producers at Area is shown in the screenshot below:



This widget can be filtered by Month and Area

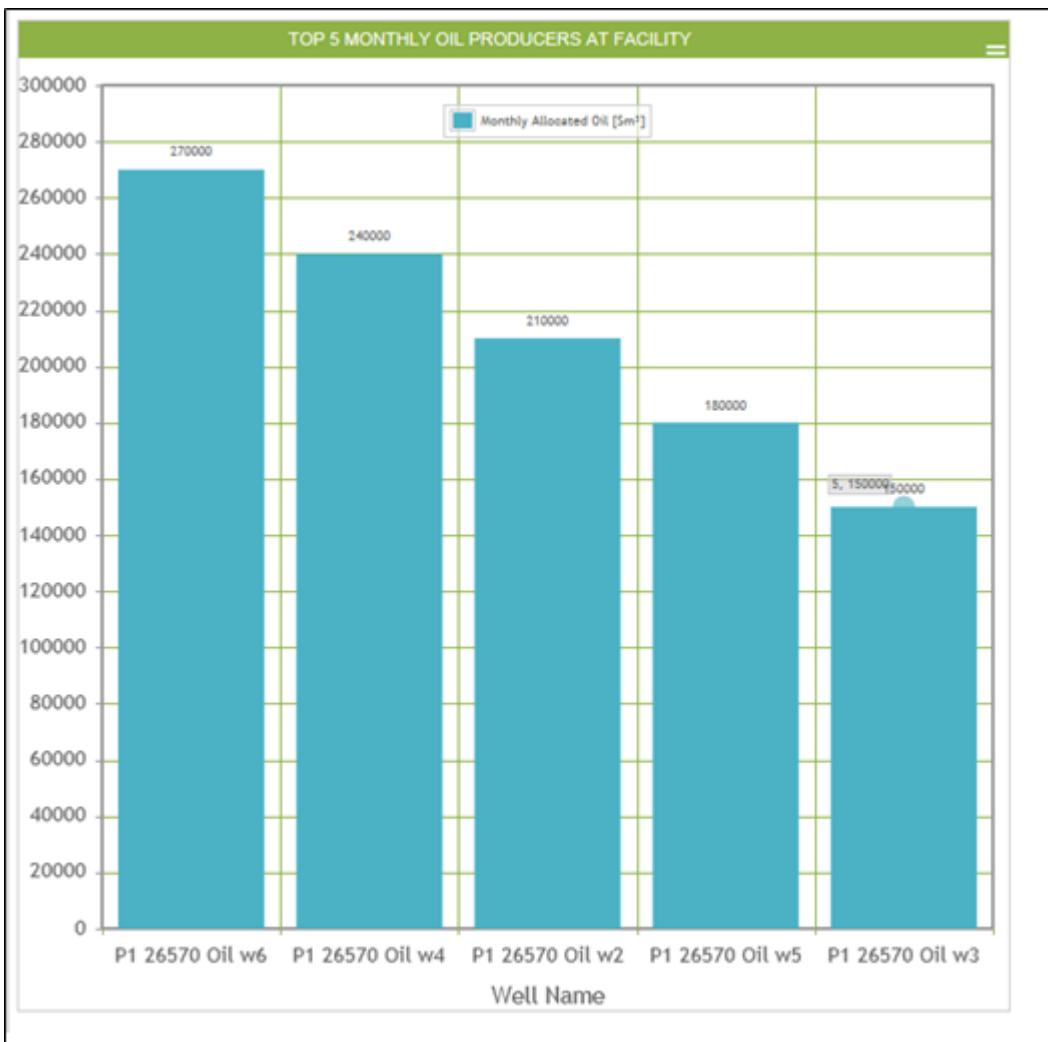
The function of each chart parameters is defined below:

- Legend position: The position of the legend in the chart.
- Show Point Labels: Each series will be labeled with the produced Oil volume value.

Similar widgets and configuration exists for Gas and Condensate.

#### Widget 4: Top 5 Monthly Oil Producers at Facility:

This dashboard widget displays top 5 Oil Producers at a selected Facility and Month. In the example below, it displays the Top 5 Oil Producer at P1 Facility for April, 2012.



The parameters that needs to be configured for Top 5 Monthly Oil Producers at Facility is shown in the screenshot below:



This widget can be filtered by Month and Facility.

The function of each chart parameters is defined below:

- Legend position: The position of the legend in the chart.
- Show Point Labels: Each series will be labeled with the produced Oil volume value.

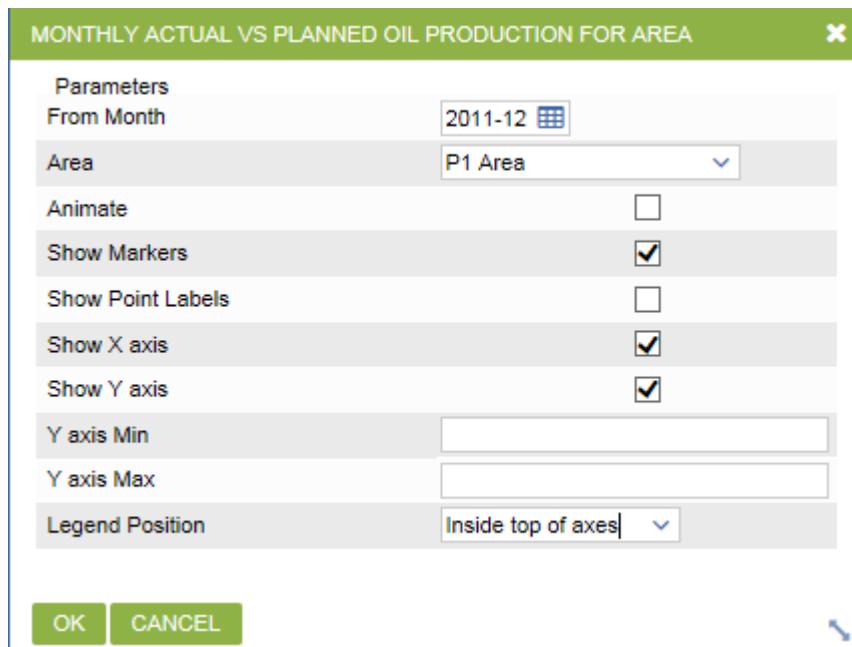
Similar widgets and configuration exists for Gas and Condensate.

#### Widget 5: Monthly Actual Vs Planned Production For Area:

This dashboard widget displays Monthly Actual vs Planned Oil Production for selected Area and Month. In the example below, it displays the Monthly Actual vs Planned Oil Production at Area for the past 12 months when the selected month is December, 2011.



The parameters that needs to be configured for Monthly Actual vs Planned Oil Production for Area is shown in the screenshot below:



This widget can be filtered by Month and Area.

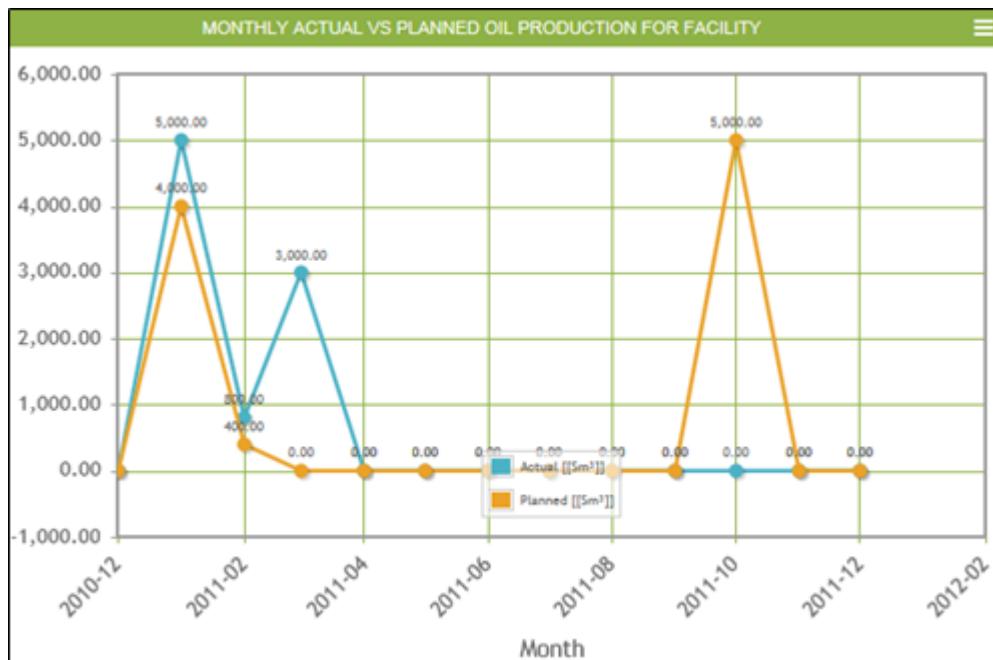
The function of each chart parameters is defined below:

- Animate: Will show animation of the line being drawn
- Show Markers: Each series will be marked with a circle
- Show Point Labels: Each series will be labeled with the Oil volume
- Show X Axis: Shows the x axis value and its label (Month).
- Show Y Axis: Shows the y axis value (Oil volume).
- Y axis Min: The minimum value of the Y axis to be displayed on the chart
- Y axis Max: The maximum value of the Y axis to be displayed on the chart
- Legend position: The position of the legend in the chart.

Similar widgets and configuration exists for Gas, Condensate and Water.

#### **Widget 6: Monthly Actual Vs Planned Production For Facility:**

This dashboard widget displays Monthly Actual vs Planned Oil Production for selected Facility and Month. In the example below, it displays the Monthly Actual vs Planned Oil Production at P1 Facility for past 12 months when the selected month is December, 2011.



The parameters that needs to be configured for Monthly Actual vs Planned Oil Production for Facility is shown in the screenshot below:

**MONTHLY ACTUAL VS PLANNED OIL PRODUCTION FOR FACILITY**

Parameters	
From Month	2011-12 <input type="button" value="..."/>
Facility	P1 Facility 1 <input type="button" value="..."/>
Animate	<input type="checkbox"/>
Show Markers	<input checked="" type="checkbox"/>
Show Point Labels	<input type="checkbox"/>
Show X axis	<input checked="" type="checkbox"/>
Show Y axis	<input checked="" type="checkbox"/>
Y axis Min	<input type="text"/>
Y axis Max	<input type="text"/>
Legend Position	Inside top of axes <input type="button" value="..."/>
<b>OK</b> <b>CANCEL</b>	

This widget can be filtered by Month and Facility.

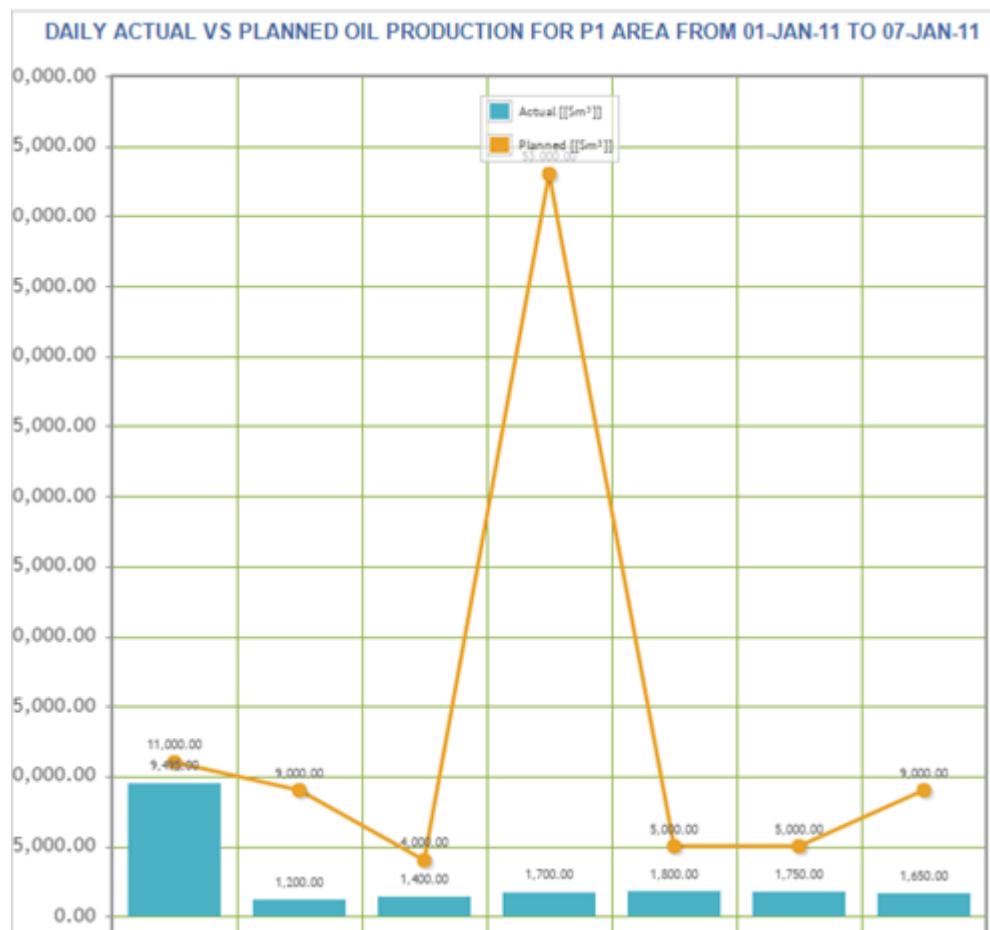
The function of each chart parameters is defined below:

- Animate: Will show animation of the line being drawn
- Show Markers: Each series will be marked with a circle
- Show Point Labels: Each series will be labeled with the Oil volume
- Show X Axis: Shows the x axis value and its label (Month).
- Show Y Axis: Shows the y axis value (Oil volume).
- Y axis Min: The minimum value of the Y axis to be displayed on the chart
- Y axis Max: The maximum value of the Y axis to be displayed on the chart
- Legend position: The position of the legend in the chart.

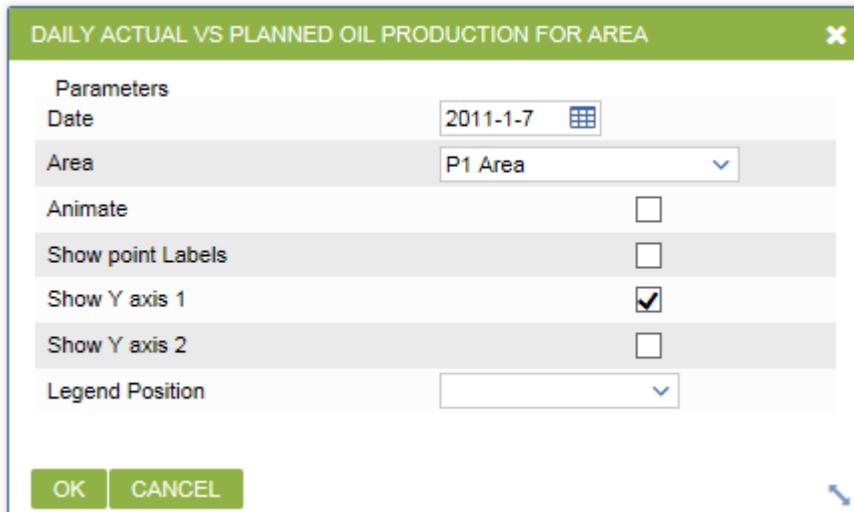
Similar widgets and configuration exists for Gas, Condensate and Water.

#### Widget 7: Daily Actual Vs Planned Production For Area:

This dashboard widget displays Daily Actual vs Planned Oil Production for selected Area and Day. In the example below, it displays the Daily Actual vs Planned Oil Production at P1 Area for the past 7 days when selected date is 7-Jan-2011.



The parameters that needs to be configured for Daily Actual vs Planned Oil Production for Area is shown in the screenshot below:



This widget can be filtered by Date and Area.

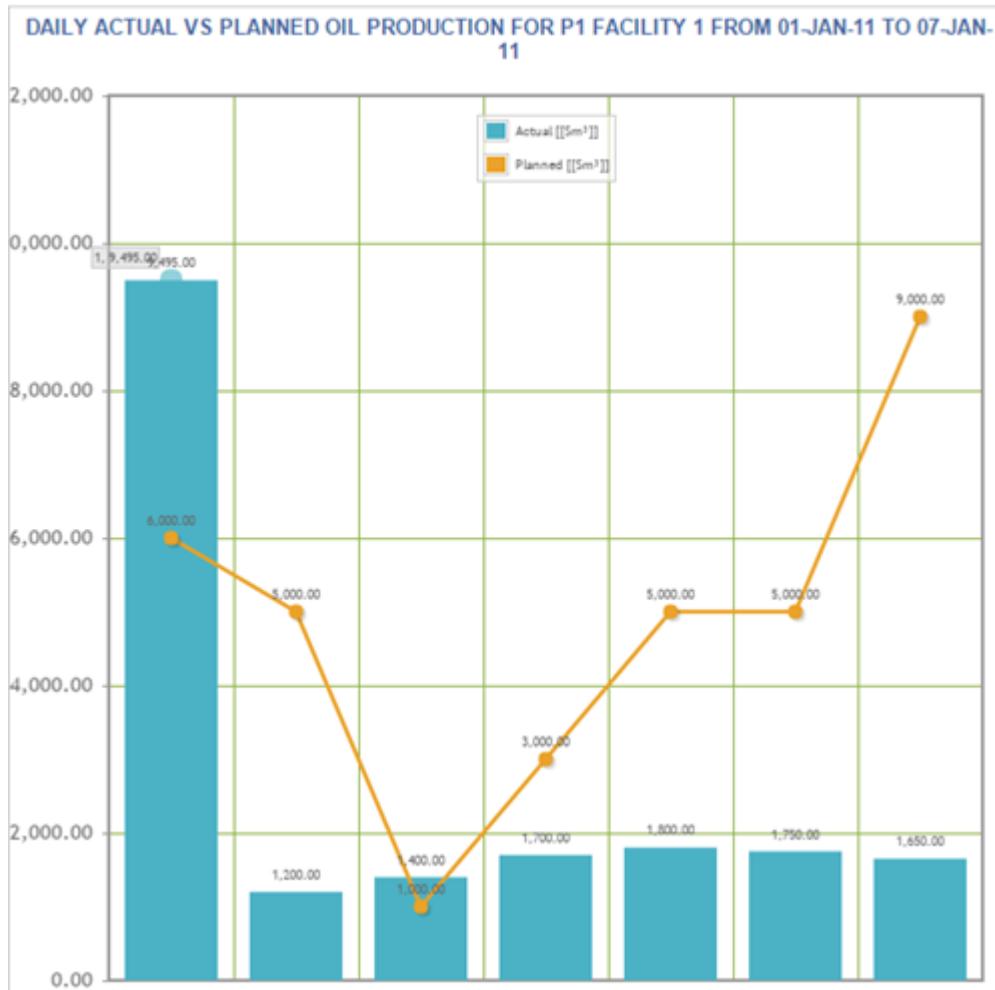
The function of each chart parameters is defined below:

- Animate: Will show animation of the line being drawn
- Show Point Labels: Each series will be labeled with the Oil volume
- Show Y Axis 1: Shows the Y axis value (Actual).
- Show Y Axis 2 Shows the Y axis value (Planned).
- Legend position: The position of the legend in the chart.

Similar widgets and configuration exists for Gas, Condensate and Water.

#### **Widget 8: Daily Actual Vs Planned Production For Facility:**

This dashboard widget displays Daily Actual vs Planned Oil Production for selected Facility and Date. In the example below, it displays the Daily Actual vs Planned Oil Production at P1 Facility for past 7 days when selected date is 7-Jan-2011.



The parameters that needs to be configured for Daily Actual vs Planned Oil Production for Facility is shown in the screenshot below:

DAILY ACTUAL VS PLANNED OIL PRODUCTION FOR FACILITY

Parameters		
Date	2011-1-7	<input type="button" value=""/>
Facility	P1 Facility 1	<input type="button" value=""/>
Animate	<input type="checkbox"/>	
Show point Labels	<input type="checkbox"/>	
Show Y axis 1	<input checked="" type="checkbox"/>	
Show Y axis 2	<input type="checkbox"/>	
Legend Position	Inside top of axes	
<input type="button" value="OK"/> <input type="button" value="CANCEL"/>		

This widget can be filtered by Date and Facility.

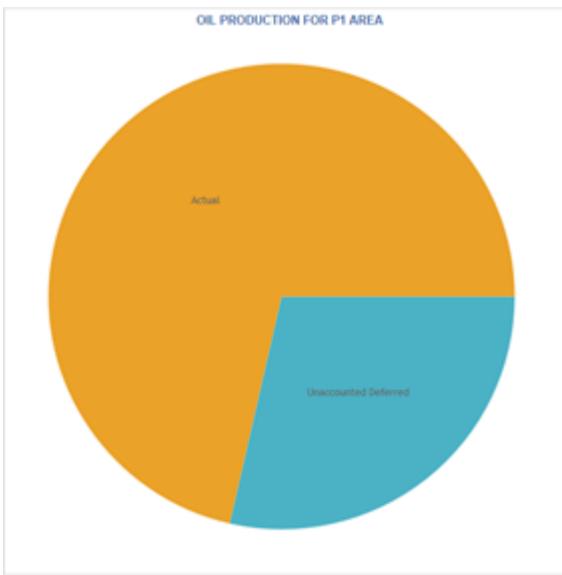
The function of each chart parameters is defined below:

- Animate: Will show animation of the line being drawn
- Show Point Labels: Each series will be labeled with the Oil volume
- Show Y Axis 1: Shows the Y axis value (Actual).
- Show Y Axis 2 Shows the Y axis value (Planned).
- Legend position: The position of the legend in the chart.

Similar widgets and configuration exists for Gas, Condensate and Water.

#### Widget 9: Oil Production For Area

This dashboard widget displays total Oil Production volume (Actual/Deferred/Unaccounted Deferred) for selected Area and Date. In the example below, it displays the total Oil Production at P1 Area when selected date is 1-Jan-2011.



The parameters that needs to be configured for Oil Production for Area is shown in the screenshot below:



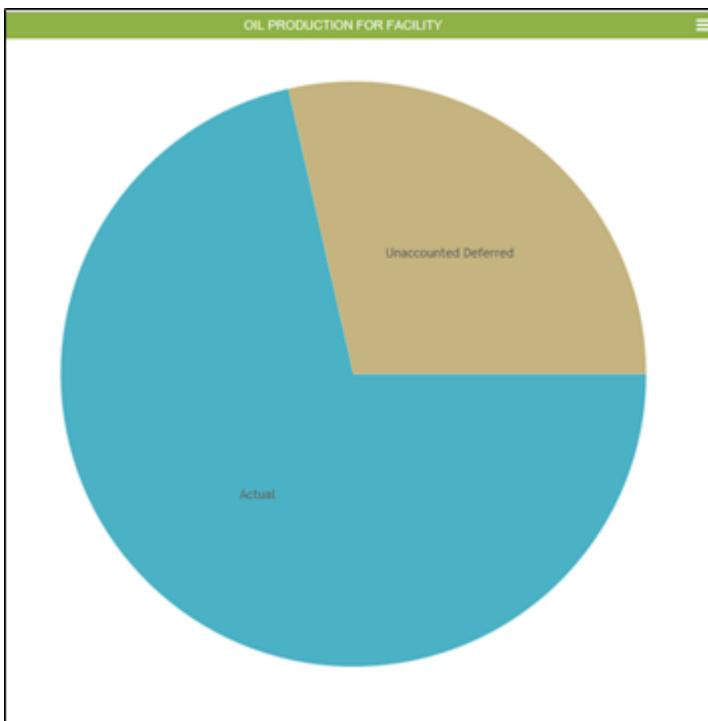
This widget can be filtered by Date and Area.

The function of each chart parameters is defined below:

- Show Data Labels: Each series will be labeled with the Oil volume value.

#### Widget 10: Oil Production For Facility

This dashboard widget displays total Oil Production volume (Actual/Deferred/Unaccounted Deferred) for selected Facility and Date. In the example below, it displays the total Oil Production at P1 Facility when selected date is 1-Jan-2011.



The parameters that needs to be configured for Oil Production for Facility is shown in the screenshot below:

**OIL PRODUCTION FOR FACILITY**

Parameters

Date: 2011-01-01

Show Data Labels:

Facility: P1 Facility 1

**OK** **CANCEL**

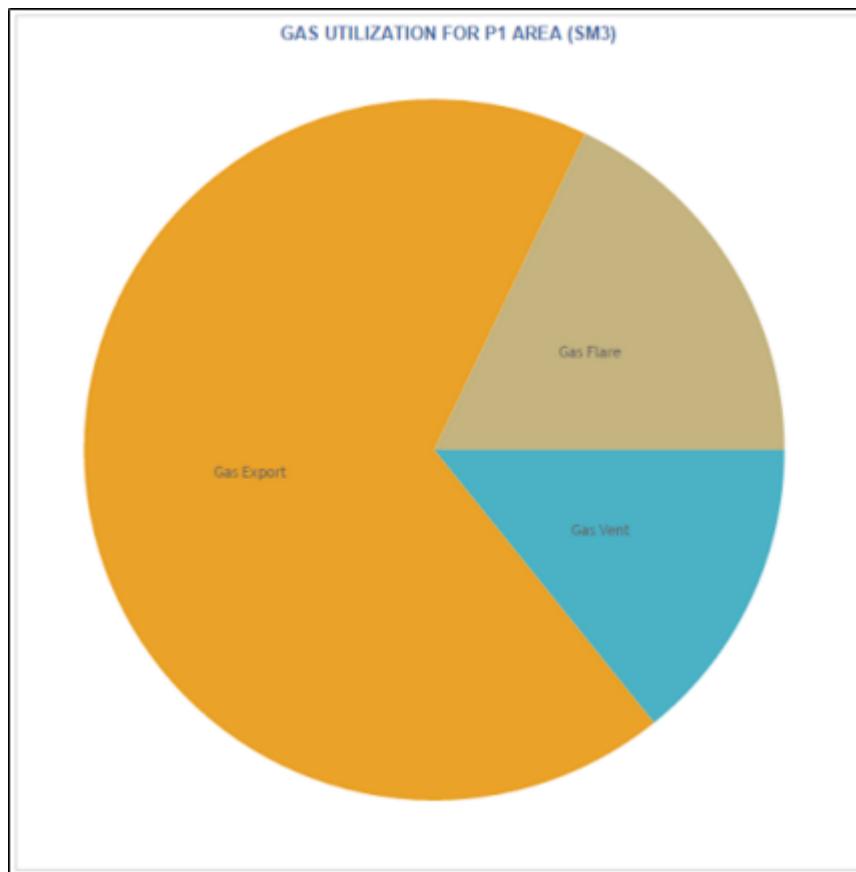
This widget can be filtered by Date and Facility.

The function of each chart parameters is defined below:

- Show Data Labels: Each series will be labeled with the Oil volume value.

#### Widget 11: Gas Utilization For Area

This dashboard widget displays Gas Utilization volume for selected Area and Date. This widget will show data for selected date only. In the example below, it displays the Gas Utilization at P1 Area when selected date is 7-Jan-2011.



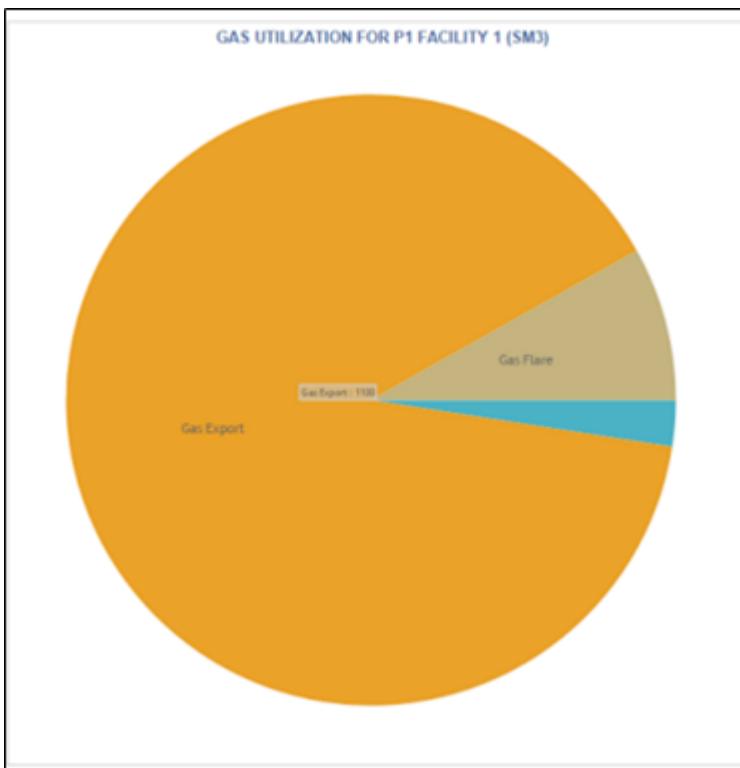
The parameters that needs to be configured for Gas Utilization for Area is shown in the screenshot below:



This widget can be filtered by Date and Area.

#### Widget 12: Gas Utilization For Facility

This dashboard widget displays Gas Utilization volume for selected Facility and Date. This widget will show data for selected date only. In the example below, it displays the Gas Utilization at P1 Facility when selected date is 7-Jan-2011.



The parameters that needs to be configured for Gas Utilization for Facility is shown in the screenshot below:

GAS UTILIZATION FOR FACILITY

Parameters

Date: 2011-1-7

Facility: P1 Facility 1

OK CANCEL

This widget can be filtered by Date and Facility.

## EC Sales

This chapter contains the Technical Documentation for EC Sales

- EC Sales Technical Documentation

## EC Sales Technical Documentation

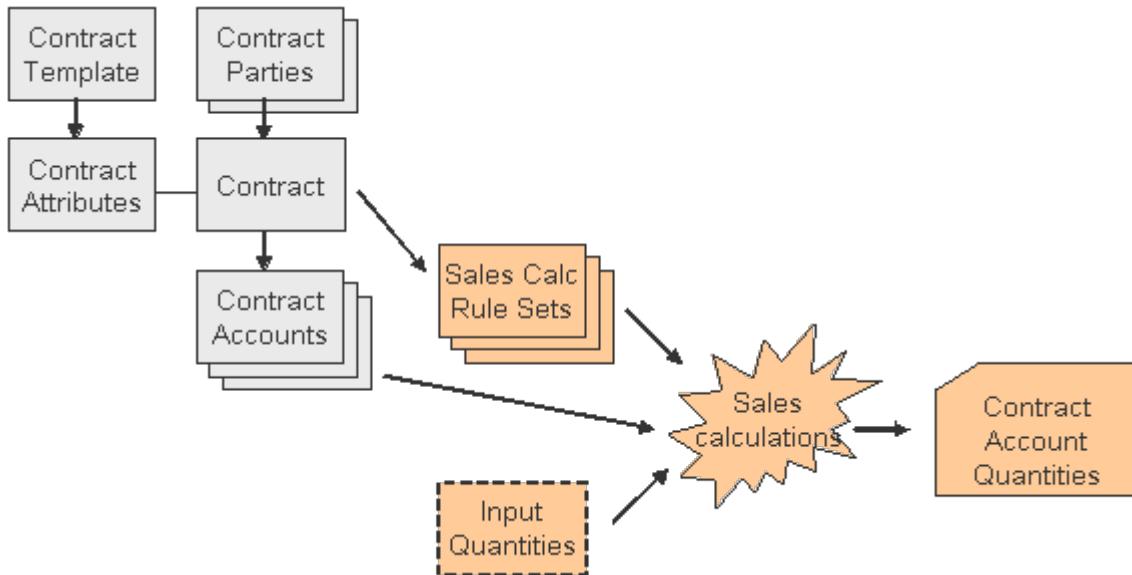
EC Sales is divided into several business areas (BA), which are as follows:

Business Area	Description
GS (Gas Sales)	Supporting the Gas Sales processes
SD (Sales Dispatching)	Sub area within GS, supporting capacity booking, nominations, nomination rules, delivery events and follow-up of actual deliveries
PR (Price Determination)	Sub area within GS, supporting the calculation of unit prices based rules typically defined by a contract. Such rules will normally relate to price indices as basis for the price determination.
SA (Sales Allocation)	Sub area within GS, supporting the calculation of hydrocarbon sales quantities according to rules and conditions stated in the commercial sales contracts.
TR (Trading)	Sub area within GS, supporting portfolio management, risk management, deal capturing / validation and price simulation
GP (Gas Purchase)	Supporting the gas purchasing processes
OS (Oil Sales)	Supporting the oil sales processes

This document is mainly addressing the Sales Dispatching, Price Determination and Sales Allocation functionality. For enquiries about other EC Sales capabilities, please contact Tieto directly.

### EC Sales – Sales Allocation (SA)

Sales Allocation is the EC concept for following up contractual commitments relating to transactional quantities.



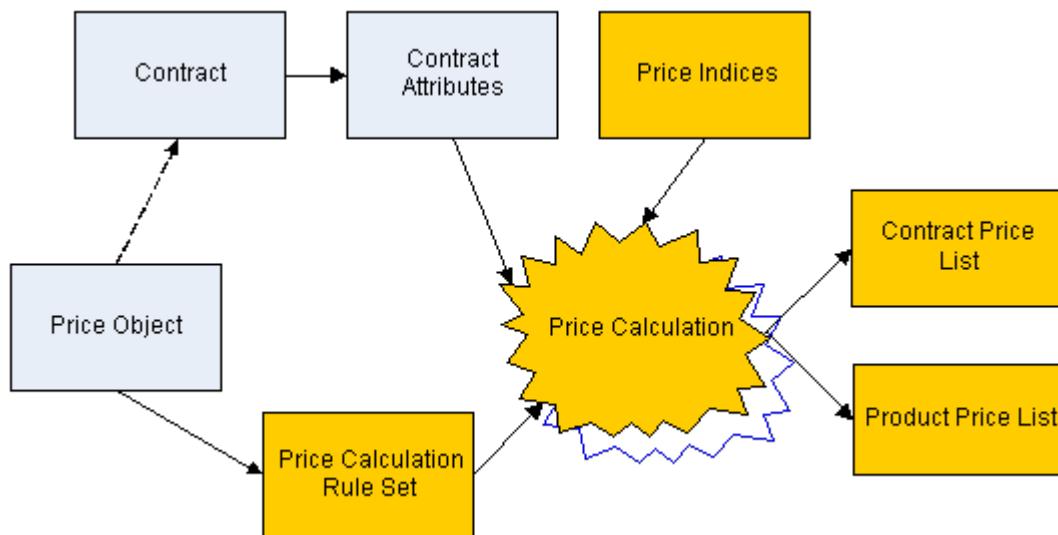
As shown in the figure, the sales allocation (sales calculation) concept relies on the EC Contract Concept. For a contract, it is possible to associate Sales Calculation Rule Sets. Such rule sets are defined and executed in by the EC Calculation Framework. The calculation rules will relate to Contract Accounts associated to the given contract, and the results of the calculations will be stored as Contract Account Quantities.

As an example, a gas sales agreement defines that a monthly off-spec gas quantity shall be recorded, according to described rules. To configure this in EC, a contract account: Monthly Off Spec Qty would typically be defined. Further, the rules described by the contract will be configured as equations (Sales Calculation Rule Sets) in the EC Calculation Framework. The defined equations may also relate to contract attributes for the relevant contract. To execute the actual

Sales Allocation for the Monthly Off Spec Qty account, the calculation will be run, using available Input Quantities (e.g. delivery quantities combined with Off Spec Events), and the result will be stored for the appropriate period in the Contract Account Quantities "table".

A complex contract will typically have a number of contract accounts, typically also operating on different time spans, e.g. daily, monthly and yearly level.

## EC Sales – Price Determination (PR)



The EC Price Determination capabilities support the determination of product unit prices, typically based on price indices and contractual price definition rules.

There are two types of prices. Prices that relate to a specific contract only, and more generic prices that are valid for a given product but multiple contracts.

As shown in the figure, the price calculations relate to Price Objects as well as the EC Contract Concept. The price calculation rules defined by the Price Object will be associated to Price Calculation Rule Sets. Such rule sets are defined and executed by the EC Calculation Framework. Further, the price calculations rely on price indices, typically available on daily, monthly and yearly level. The price calculations may also relate to contract parameters or contract account quantities. The results of the calculations will be stored as Contract Price List or Product Price List values.

### Price Concept definition example

In the next paragraphs the process defining a price concept is described step by step. Two price concepts are constructed to illustrate the process from defining the price concept until the price is ready for use for a given product or contract. The price concepts that will be used as examples are:

- ABC price concept (is used for illustration in each step and included in example data)
- CIF price concept (is included in example data)

### Step 1 - define the price concept

The price concept can be any concept that the customer wants to define. The price concept will have to be defined with a name and a code. Note that the price concept is not time driven, meaning that there are no date information related to a price concept. Example data:

Price concept code	Name
ABC	ABC Price Concept
CIF	CIF Price concept

### Step 2 – define price elements

The next step is to define all the elements that the price concept consists of. In the constructed example '**ABC price concept**' each price element in the price concept has to be defined and they are **A**, **B** and **C**. First of all we have to relate each price element to a price concept which is the '**ABC price concept**'. Furthermore we have to define a price element

code and a price element name. Example data:

Price concept code	Price element code	Name
ABC	A	A element
ABC	B	B element
ABC	C	C element
CIF	Cost	Cost element
CIF	Insurance	Insurance element
CIF	Freight	Freight element

### Step 3 – define the price object

The price object is defined based on a price concept and the price object has to point to either a product or the combination product/contract. When creating the price object the following information can be defined:

- Price object code and name
- Start and end date
- Product name
- Price concept
- Contract name
- Currency
- UOM
- Time span
- Calculation rule
- Calculation sequence number
- Description

Example data:

Price object code	Name	Start date	End date	Product	Price concept code	Contract	Currency	UOM	Time span	Calc rule	Calc seq	Description
EX_1	Example_1	01.01.2005		Prod_Ex_1	ABC	Contr_Exa_mple_1	EUR	SM3	MTH	EX_RULE_1	1	Example_1
EX_2	Example_2	01.01.2005		Prod_Ex_2	CIF	Contr_Exa_mple_2	EUR	SM3	MTH	EX_RULE_2	1	Example_2

When the necessary input has been provided the price calculation can be run.

### Step 4 – Price calculation

The Price Calculation business function is used to calculate the price values based on the definitions described above. The individual rules for each customer are implemented in the calculation engine. Based on the example we construct a formula based on the price elements that are defined for the price concept:

$$C = A + B$$

The formula has to be defined in the calculation engine and each of the price elements will be defined as a read or write variable. In our case price elements A and B will be defined as READ variables since they consist of input to the calculation while price element C is defined as WRITE variable since it is calculated based on the formula above. Price elements that are input to the formula will typically be price indices or contractual attributes. For more information about how to set up the formulas, see the documentation covering the calculation engine.

For results of the price calculation see the next section.

### Step 5 – price lists (contract/product)

The price lists (for contract and product) will allow display and update of price values. Based on selection of month, contract and product in the navigator data will appear in the data area in this business function. Example data:

Valid from	Price product name	Price object name	Price concept	Price element	Calculated price	Adjusted price	UOM	Comment	Contract
01.01.2005	CRUDE	Example _1	ABC	A	100	100	SM3	Example _1	Contr_Ex ample_1
01.01.2005	CRUDE	Example _1	ABC	B	10	10	SM3	Example _1	Contr_Ex ample_1
01.01.2005	CRUDE	Example _1	ABC	C	110	110	SM3	Example _1	Contr_Ex ample_1
01.01.2005	BUTANE	Example _2	CIF	Cost	5,12	5,12	SM3	Example _2	Contr_Ex ample_2
01.01.2005	BUTANE	Example _2	CIF	Insurance	0,65	0,65	SM3	Example _2	Contr_Ex ample_2
01.01.2005	BUTANE	Example _2	CIF	Freight	2,88	2,88	SM3	Example _2	Contr_Ex ample_2

### System properties

The following properties are maintained in the ctrl\_property\_meta table

<b>Label</b>	<b>Description</b>	<b>Default value</b>
Instantiate Comment at Period Sales Nomintion	Flag for whether or not to instantiate a new empty comment to Period Sales Nomination	Y
Instantiate Daily Contract Delivery	Flag for whether or not to instantiate Daily Contract Delivery records	Y
Instantiate Daily Contract Status	Flag for whether or not to instantiate records used to list values in Daily Contract Account Status	Y
Instantiate Daily Price Index	Flag for whether or not to instantiate Daily Price Index records	Y
Instantiate Daily Price Rate	Flag for whether or not to instantiate Daily Price Rate records	Y
Instantiate Monthly Contract Delivery	Flag for whether or not to instantiate Monthly Contract Delivery records	Y
Instantiate Monthly Contract Status	Flag for whether or not to instantiate records used to list values in Monthly Contract Account Status	Y
Instantiate Monthly Expenditure	Flag for whether or not to instantiate Monthly Expenditure records	Y
Instantiate Monthly Expenditure Forecast	Flag for whether or not to instantiate Monthly Expenditure Forecast records	Y
Instantiate Monthly Price Index	Flag for whether or not to instantiate Monthly Price Index records	Y
Instantiate Monthly Price Rate	Flag for whether or not to instantiate Monthly Price Rate records	Y
Instantiate Sub Daily Contract Status	Flag for whether or not to instantiate records used to list values in Sub Daily Contract Account Status	Y
Instantiate Wet Gas Hourly Profile	Flag for whether or not to instantiate Wet Gas Hourly Profile records	Y
Instantiate Yearly Contract Status	Flag for whether or not to instantiate records used to list values in Yearly Contract Account Status	Y

## EC Transport

This chapter contains the Technical Documentation for EC Transport

- EC Transport Technical Documentation
- EC Transport Dashboard

## EC Transport Technical Documentation

A transport operation in Energy Components is defined as one oil / gas pipeline or group of pipelines which share processing, terminal and storage facilities and have at least one interconnecting hydrocarbon stream. Basically the transporters role is to redeliver hydrocarbons at an exit point according to an agreement (transport agreement). In the case of transporting hydrocarbons in the liquid phase, such a role might also include managing the off-take lifting's according to agreement (lifting procedure). Such a redelivery service may include processing and storage of hydrocarbons (processing / storage agreement).

In many cases the transport is clearly distinguished from the producer of the hydrocarbons, where the transporters offers services according to a fee (often referred to as "tariff") for one or more producers of hydrocarbons. In other cases the transport operation forms an integral part of the production operation by, e.g. performing the cargo lifting and inventory management of crude or other products shipped on tankers (or trains / trucks).

Tieto provides complete hydrocarbon accounting solutions for any type of transport operation, including commercial gas pipelines, oil pipelines, LNG/NGL plants and off-take terminals.

EC Transport is divided into several business areas (BA), which are as follows:

Business Area	Description
CA (Cargo Administration)	Supporting the Cargo Administration process (also known as tanker scheduling, shipment planning, etc)
CP (Cargo Planning)	Sub area within CA, supporting the planning of the cargo lifting activities
TO (Terminal Operation)	Sub area within CA, supporting the terminal activities, irregularities and production of cargo documents
LA (Lifting Account)	Sub area within CA, supporting lifting account transactions
OD (Oil Delivery)	Supporting the Oil Delivery process
GD (Gas Dispatching)	Supporting the Gas Dispatching process
GD (Gas Delivery)	Supporting the Gas Delivery process
FC (Forecast)	Supporting the matching of field forecast volumes with plant production capacities

*This document is restricted to defined cargo administration, gas dispatching and forecasting functionality, i.e. the CP, TO, LA, GD and FC business areas. For enquiries about other EC Transport capabilities, please contact Tieto directly.*

## EC Transport Cargo Status Rules

In addition to the general record status concept, the Cargo Administration part of EC Transport do include a dedicated cargo status. This section explains, in quite detail, the cargo status concept.

### Cargo Status Values and Workflow

Cargo status can be customised as required by the customers. *All customer cargo statuses must be mapped against system cargo statuses.*

This can be done in Cargo Status Mapping (CO.2006). For more information about the mapping see the Business Function document.

The next sections describe the different system statuses that a Cargo can have. This is also the general workflow. Not following these steps will have impact on how the system behaves.

The available system cargo statuses are described below. The natural workflow is top down (Not cancelled).

- T – Tentative (part of future plan, cargoes are not available in terminal operation screens)
- R – Ready for Harbour (cargoes are available in terminal operations)
- C – Closed (lifting completed)
- A – Approve (warrantee expired)
- D – Cancelled (lifting will not take place)

The only requirement for customer cargo statuses is that the Cancelled status must have the code D.

There are four screens where the user can change system cargo status.

- Nomination Entry (CP.0001) – The cargo is created in this business function with status set to 'Tentative'
- Cargo Information (CP.0003)
- Nomination Details (CP.0004)
- Lifting Instruction (CP.0005) – Typically used to set to 'Ready for Harbour'
- BLMR Info (TO.0005) – Typically used to set to 'Closed'

When a system cargo status is set to **Closed** the record status is set to V (Verified) for these tables in the database. This will prevent the user from doing updates after a cargo is closed.

- CARGO\_TRANSPORT
- STORAGE\_LIFT\_NOMINATION
- STORAGE\_LIFTING

When a system cargo status is set to **Approved** the record status is set to A (Approved) for these tables in the database. This will prevent the user from doing updates after a cargo is approved.

- CARGO\_TRANSPORT
- STORAGE\_LIFT\_NOMINATION
- STORAGE\_LIFTING
- CARGO\_ACTIVITY
- CARGO\_ANALYSIS, CARGO\_ANALYSIS\_ITEM
- CARGO\_LIFTING\_DELAY
- CARRIER\_INSPECTION

### **Rules for System cargo status**

The matrix below describes what a system cargo status can change to

From/ To	Tentative	Ready for Harbour	Closed	Approved	Cancelled
<b>Tentative</b>		Y	N	N	Y
<b>Ready for Harbour</b>	Y*		Y*	N	Y
<b>Closed</b>	N	Y		Y	N
<b>Approved</b>	N	N	N		N
<b>Cancelled</b>	N	N	N	N	

See validation section for special rules.

### **Validation**

Below is a list of additional validation rules used when system cargo status changes.

From Status	To Status	Validation
Ready For Harbour	Tentative	Not allowed if there are storage liftings (BLMR Info (TO.0005) got data)
Ready For Harbour	Closed	Not allowed if no storage lifting (BLMR Info (TO.0005) misses data)
Closed	Ready For Harbour	Not allowed if lifting account on nominations is closed (Monthly Account Status LA.0002)

### **Error Messages**

From Status	To Status	Validation
Tentative	Closed or Approved	The Cargo Status cannot change from Tentative to Closed or Approved
Ready for Harbour	Approved	The Cargo Status cannot change from Ready for Harbour to Approved
Closed	Tentative, Cancelled	The Cargo Status cannot change from Closed to Tentative or Cancelled
Approved	Tentative, Closed or Cancelled	The Cargo Status cannot change from Approved to Tentative, Closed or Cancelled
Cancelled	Tentative, Closed or Approved	The Cargo Status cannot change from Cancelled to Tentative, Closed or Approved
Ready for Harbour	Tentative if storage lifting	The Cargo Status cannot change to Tentative if nominated lifted value is different than null for one of the nominations
Ready for Harbour	Closed if no storage lifting	The Cargo Status cannot change to Closed if nominated lifted value is empty for one of the nominations
Closed	Ready for Harbour	The Cargo Status cannot change to Ready for Harbour if the lifting account on nomination is closed
When trying to move a nomination from a cargo that is closed		It is not allowed to move a nomination from a cargo that is closed or approved

### Instantiation / Deletion

From Status	To Status	Validation
Tentative	Ready for Harbour	Instantiate Cargo Activity Timesheet (TO.0001), Cargo Analysis (TO.0006), and BLMR Info (TO.0005)
Ready for Harbour	Tentative	Delete Cargo Activity Timesheet (TO.0001), Cargo Analysis (TO.0006), and BLMR Info (TO.0005)
<All>	Cancelled	Quantities are ignored in balance calculations

## Interface Functions

### Overview

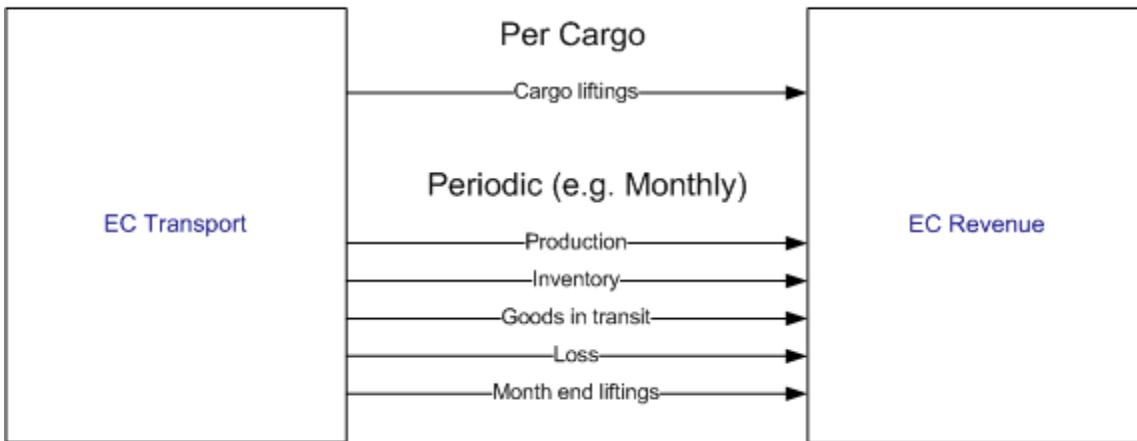
The modules of EC Transport have a close integration with other EC modules, and do wherever applicable rely on the same data sources. However, when information is used in different domains, information is normally replicated from the source domain to the target domain. Such replication is performed by Interface Functions. A set of default Interface Functions are supplied with the EC product. An example of such replication is when cargo related information is to be

used for invoicing purposes. Then the cargo information (source domain) is replicated to EC Revenue (target domain) to ensure the information lives its own life as it enters the commercial domain.

When configuring an integrated EC implementation, the available Integration Functions must be verified to ensure that the default information mapping corresponds with the identified business requirements. The Interface Functions must be adjusted if deviations exists.

The below diagram outlines the default Interface Functions supplied with the EC Transport product:

## EC Transport – Interface Functions



### Cargo Liftings (Parcels)

Cargo Liftings (per Parcel) are replicated to EC Revenue when it is actually lifted on the carrier, i.e. the Bill of Lading value. For CIF (Cost Insurance Freight) cargoes also expected unload and unload values are replicated.

In Product Measurement Setup (CO.2002) the quantities to replicate can be configured. There can be only four different UOMs, but each can have both net and gross. It is recommended that the mapping is the same for Load and Unload lifting events.

In EC Revenue there are tree quantity types:

- LOAD
- EXP\_UNLOAD
- UNLOAD

Load represent the quantities set in BLMR Info (TO.0005) business function. Expected Unload is calculated based on the load value. The unload value is set in Unload Info (TO.0010). Expected Unload and Unload is only replicated for cargoes where incoterm is CIF.

#### Preconditions

- Nominations must be connected to a Contract and the contract must have 'Available in Revenue' checked. If not it will not be replicated to EC Revenue. No error message.
- Nominations must have an Incoterm. Error message from EC Revenue
- At least one measurement item must exist on the product associated to the storage.
- In 'Product Measurement Setup' there must exist a mapping on at least one measurement item and it must be 'Qty 1'. Measurement Item will indicate if this is a 'Net' or 'Gross' value. 'Net' is default.

Some of the values replicated to EC Revenue can be updated in other business functions, e.g. Carrier in Cargo Info, Consignor in Nomination Detail. This is ok as long as the nomination is not used in EC Revenue. **As soon as a contract is processed and the cargo details are picked up, updates on a cargo/nomination are not allowed.**

### Cargo quantities

Cargo Quantities are quantities that are replicated to EC Revenue on a monthly basis.

Typically these quantities are replicated a few days after the month end. The month should be finalised and closed. Some

of the quantities will fail if there are missing data, others will replicate wrong data.

The different quantities that are supported in the EC Transport Interface Functions are:

- PROD – Production
- INV - Inventory
- GIT – Goods in transit
- LOSS - Loss
- MEL – Month end Lifting's / Exported not lifted

Common for all the quantities is that the quantities are aggregated to the same dimension. The dimensions are Product, Company and Profit Centre. The unit and value used is the 'Nominated Unit' set for the product in Product Measurement Setup (CO.2002).

The scheduler is used to do the replication. The following business actions should be used when a scheduled job is created. What quantities to replicate is decided by the individual implementations:

- CargoProdQtyToRevenue
- CargoGitQtyToRevenue
- CargoLossQtyToRevenue
- CargoMELQtyToRevenue
- CargoInvQtyToRevenue

To learn more about how to create a Scheduled job please see the EC Framework Configuration Manual.

## **Production**

This is the total production for the month. The data is replicated from the table STOR\_DAY\_OFFICIAL.

The preconditions are

- All days of the month must have data
- The RECEIPT\_TYPE must be 'OFFICIAL'

## **Inventory**

Inventory is the difference in the opening and the closing balance. The inventory is listed based on the opening and closing balance of Lifting Accounts.

Preconditions

- Lifting Accounts must have profit centre

## **Goods In Transit**

Goods in transit is goods that is loaded onto a carrier, but not unloaded at the end of the month. This only applies to nominations where Incoterm is CIF. Meaning that the ownership of the cargo is not transferred to the buyer yet.

Preconditions

- Nominations must have incoterm set to CIF. Other with unload data will be ignored.
- Nominations must be connected to a Lifting Account that have Profit Centre

## **Loss**

Loss is the difference between loaded quantity and unloaded quantity on CIF nominations.

Preconditions

- Nominations must have incoterm set to CIF. Other with unload data will be ignored.
- Nominations must be connected to a Lifting Account that have Profit Centre

## **Month End liftings**

Month End Lifting or Exported Not Lifted is how much of the lifting quantity is left in the storage (tank) at the end of the

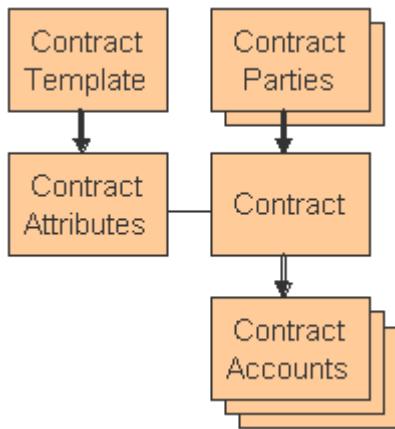
month. This applies to liftings where the actual lifting time crosses a month end border.

#### Preconditions

- Nominations must be connected to a Lifting Account that have Profit Centre

## The EC Contract Concept

The EC Contract Concept is designed to model different contract types, spanning from straightforward to very complex variants. As shown in the figure below, modelling a contract in EC involves:



A further description is provided below.

### Contract

The definition of the core contract values, including contract period, contract year/day offsets, etc.

### Contract Parties

Attaching the contract Vendor and Customer to the contract. The contract parties may be a constellation of multiple companies, associated with the contract through an equity split.

### Contract Template

A contract template is a set of contract attributes relevant for a type of contracts. As an example, a gas sales agreement would typically need a defined set of contract attributes. It would then be appropriate to define a contract template called GSA, containing those attributes.

### Contract Attributes

All contracts must be associated with a Contract Template, defining all additional attributes relevant for this contract. In the example of a gas sales agreement, the contract could be associated with the GSA contract template, meaning the already defined contract attributes are available.

### Contract Accounts

A contract may be set up with associated Contract Accounts. Contract accounts are typically used for contracts where sales allocations (also known as contract calculations or sales calculations) are performed. The accounts for a contract would typically represent those contract obligations relating to a transactional quantity. The transactional quantities can be stored as volume, mass and energy units. Examples of contract accounts are: Monthly Sales Gas Qty, Monthly Off Spec Qty, Monthly Take Or Pay Qty.

When configuring Contracts and Contract Attributes it is possible to define if these should be accessible throughout all EC modules, or if access limitation should apply to one or several modules (EC Transport, EC Sales, etc). All contract attributes are effective dated, meaning it is possible to define contract attributes changing value during the contract life time.

## EC Transport Gas Dispatching

To increase the flexibility in some of the dispatching navigators the following features has been added to the url, thus making it easy for projects to re-use the same screen definitions for capturing alternative / additional data:

Parameter	Description
NAV_MODEL	Optional, will always have a default. Will in most screens default to the most common alternative (CONTRACT / NOMINATION_POINT...) Using this option will result in user not being able to change the model for this specific bf in preferences.
FORCE_NAV_CLASS	Optional. Indicate the navigation level of mandatory field(s) to populate.
CLASS	Optional. Owner of this class must be defined as target as per screen definition.
BF_PROFILE	See CO.1025 – Business Function Profile
TARGET	Constant. Set in screen definition.

Example from the *Daily Input Nomination* BF (pattern: <parameter name>/<value>) :

Default / Simple version:

./com.ec.tran.gd.screens/daily\_input\_nomination/**CLASS/TRNP\_DAY\_NOM\_INPUT/NOMINATION\_CYCLE/false?scree**  
ntemplate=/com.ec.tran.gd.screens/daily\_input\_nomination

With all arguments set / Advanced version:

./com.ec.tran.gd.screens/daily\_input\_nomination/**CLASS/TRNP\_DAY\_NOM\_INPUT/NOMINATION\_CYCLE/false/NAV\_MODEL/TRAN\_OPERATIONAL/BF\_PROFILE/GD.0020/FORCE\_NAV\_CLASS/NOMINATION\_POINT?scree**  
ntemplate=/com.ec.tran.gd.screens/daily\_input\_nomination

## EC Transport Cargo Administration

### Additional Units

This area of the product supports multiple units, by setting up an additional nomination unit in *Product Measurement Setup* and enabling the additional attributes on the relevant screens/classes one can setup e.g. both BBLS and m3 or GJ and kWh etc.

### Context Menu

Cargo Administration supports simple setup of context menus by using the predefined / implemented BusinessAction: com.ec.tran.cp.screens.model.ejb.GenericCargoAction.java in conjunction with the database package: ue\_cargo\_action.execute()

Using this, the projects can setup database code to be executed merely by populating the database tables:  
BF\_COMPONENT, CNTX\_MENU\_ITEM and CNTX\_MENU\_ITEM\_PARAM  
Example:



```

insert into BF_COMPONENT (BF_CODE, COMP_CODE, NAME, URL)
values ('CP.0001', 'nominations', 'nominations', '/nominations');
insert into CNTX_MENU_ITEM (BF_CODE, COMP_CODE, ITEM_CODE, NAME, ACTION_CLASS_NAME,
ACTION_ROW_SCOPE, THRESHOLD_LEVEL, CONFIRM_MESSAGE, REMARK_IND, AUTO_SAVE_IND,
MIN_SELECTED_ROW, MAX_SELECTED_ROW, FUNC_MESSAGE, FUNC_VALIDATION, DIVIDER_IND,
SORT_ORDER, DESCRIPTION)
values ('CP.0001', 'nominations', 'MOVE_P1', 'Move +1 day',
'com.ec.tran.cp.screens.model.ejb.GenericCargoAction', 'selectedRows', 10, null, 'N',
'N', null, null, null, null, 'N', 10, null);
insert into
CNTX_MENU_ITEM_PARAM(BF_CODE,COMP_CODE,ITEM_CODE,PARAMETER_NAME,PARAMETER_VALUE,SORT_ORDER)
values ('CP.0001','nominations','MOVE_P1','CARGO_NO','63',20);
insert into
CNTX_MENU_ITEM_PARAM(BF_CODE,COMP_CODE,ITEM_CODE,PARAMETER_NAME,PARAMETER_VALUE,SORT_ORDER)
values ('CP.0001','nominations','MOVE_P1','OFFSET','1',30);
/*
Remember to update ue_cargo_action.execute() in db.
*/

```

### **System properties**

The following properties are maintained in the ctrl\_property\_meta table

Label	Description	Default value
Allow sub lifting account	Allow nominations on sub lifting accounts	N
CP Forecast - Copy To Original Conf Message	Display a confirmation message before copying forecast to Original	
Capacity Release Details Contact Tab	ON/OFF by Using true/false	true
Capacity Release Details Misc Tab	ON/OFF by Using true/false	true
Capacity Release Details Other Tab	ON/OFF by Using true/false	true
Capacity Release Details Rate Tab	ON/OFF by Using true/false	true
Capacity Release Details Recall/Reput Tab	ON/OFF by Using true/false	true
Capacity Validation auto expand levels	How many levels to auto expand in Capacity Validation in EC Transport - Gas Dispatching	2
Cargo Doc - List All	List all current cargo documents regardless of selected parcel	N
Cargo Doc - Show Tab #2	Show or hide the "Other Document" tab	
Cargo Doc Generate - All Parcels	Generate all documents for current cargo in one click	N
Cargo Doc Generate - Confirmation Message	Display a confirmation message before cargo document generation	
Cargo Doc Generate - Receipt List	Generate cargo documents for specified list of contacts	

Cargo Doc Generate - Scheduled	Generate the cargo documents scheduled/async	N
Cargo Range Type	The cargo range type to be used in popups in EC Transport Cargo Planning screens	CARGO_RANGE_MIDDLE
Contract Balance Check Month Tab	ON/OFF by Using true/false	true
Contract Balance Check Tab	ON/OFF by Using true/false	true
Contract Balance External Scheduling Month Tab	ON/OFF by Using true/false	true
Contract Balance External Scheduling Tab	ON/OFF by Using true/false	true
Contract Balance Internal Scheduling Month Tab	ON/OFF by Using true/false	true
Contract Balance Internal Scheduling Tab	ON/OFF by Using true/false	true
Default system lifting rate to use for hourly liftings	Default system lifting rate to use for hourly liftings	10000
Document Instruction alignment	How to populate document instruction in EC Transport	RECEIVER_ROW
Include all parcels on a cargo in Schedule Lifting	Set to Y to include all parcels on a cargo in Schedule Lifting	Y
Include current date in Storage Level calculation	Should Storage Level calculation include sys date in queries	N
Instantiate Comments in Gas Day Summary	Flag for whether or not to instantiate a new empty comments in Gas Day Summary	Y
Instantiate Contract Seasonality	Flag for whether or not to instantiate Contract Seasonality records	Y
Instantiate Daily Contract Inventory	Flag for whether or not to instantiate Daily Contract Inventory records	Y
Instantiate Monthly Contract Inventory	Flag for whether or not to instantiate Monthly Contract Inventory records	Y
Java class for Output Nom Message	Java class used in Create Output Nom Message BF	com.ec.tran.gd.screens.model.web.SendOutNomMessageDummy
Lifting Account Document Instruction alignment	How to populate lifting account document instruction in EC Transport	RECEIVER_ROW
Location Balance Check Tab	ON/OFF by Using true/false	true
Location External Scheduling Tab	ON/OFF by Using true/false	true
Location Internal Scheduling Tab	ON/OFF by Using true/false	true
Maintain Capacity Bid Contact Tab	ON/OFF by Using true/false	true
Maintain Capacity Bid Contract Capacity Tab	ON/OFF by Using true/false	true
Maintain Capacity Bid Misc Tab	ON/OFF by Using true/false	true
Maintain Capacity Bid Other Tab	ON/OFF by Using true/false	true
Maintain Capacity Bid Rate Tab	ON/OFF by Using true/false	true
Maintain Capacity Release Contact Tab	ON/OFF by Using true/false	true

Maintain Capacity Release Contract Capacity Tab	ON/OFF by Using true/false	true
Maintain Capacity Release Location Events Tab	ON/OFF by Using true/false	true
Maintain Capacity Release Misc Tab	ON/OFF by Using true/false	true
Maintain Capacity Release Other Tab	ON/OFF by Using true/false	true
Maintain Capacity Release Prearranged Tab	ON/OFF by Using true/false	true
Maintain Capacity Release Rate Tab	ON/OFF by Using true/false	true
Maintain Capacity Release Recall/Reput Tab	ON/OFF by Using true/false	true
Nomination Matrix alignment	How to populate nomination matrix in EC Transport - Gas Dispatching	DAYTIME_ROW
Read sub day storage liftings in Storage Level calculation	Should Storage Level calculation read storage liftings from sub day table in queries	N

## EC Transport Dashboard

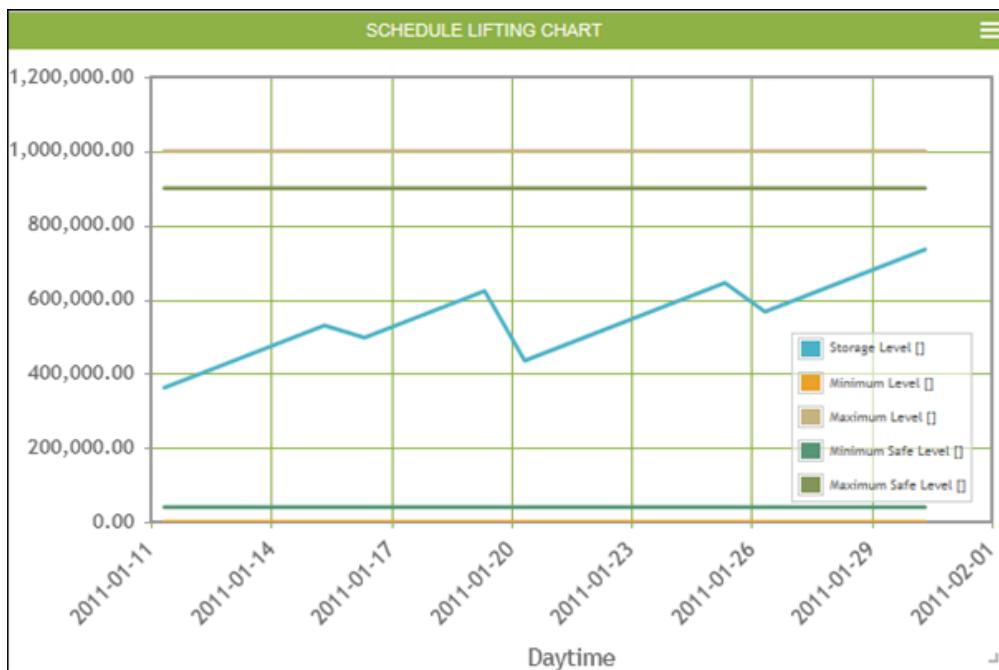
This document describes the EC Transport Dashboard widgets that comes with the standard EC product. EC Transport Dashboard consists of two line widgets and two table widgets. They are predefined, however, it is possible for projects to create or modify the dashboard query in order to support their business needs. The query is stored in the QUERY parameter in CTRL\_DASHBOARD\_PARAM table. To setup these widgets from the dashboard, there is a menu consists of different input parameters on the top right hand corner of the widget.

Transport widgets which are shipped in EC-11.1 are:

- Schedule Lifting Chart
- Lifting Account Entitlement Chart
- Logbook
- Operational Restrictions

### Schedule Lifting Chart:

This dashboard line widget displays the Storage level for a period.



The parameters which can be setup for Schedule Lifting Chart is shown in the screenshot below:

SCHEDULE LIFTING CHART

Parameters	
From Date	2011-01-11 <input type="button" value="..."/>
To Date	2011-01-31 <input type="button" value="..."/>
Storage	TS1 Heavy Crude_Storage <input type="button" value="..."/>
Production Plan	*Planned/Official <input type="button" value="..."/>
Show X axis	<input checked="" type="checkbox"/>
Show Y axis	<input checked="" type="checkbox"/>
Minimum Level Y axis	<input type="checkbox"/>
Maximum Level Y axis	<input type="checkbox"/>
Minimum Safe Level Y axis	<input type="checkbox"/>
Maximum Safe Level Y axis	<input type="checkbox"/>
Show Markers	<input type="checkbox"/>
Legend Position	Inside bottom-right of <input type="button" value="..."/>
Y axis Min	<input type="text" value="0"/>
Y axis Max	<input type="text"/>
Animate	<input type="checkbox"/>
Show Point Labels	<input type="checkbox"/>
<input type="button" value="OK"/> <input type="button" value="CANCEL"/>	

The Schedule Lifting Chart widget can be filtered by From Date, To Date, Storage, Production Plan.

The function of each chart parameters is defined below:

- Show X Axis: Shows the x axis value and its label (Daytime).
- Show Y Axis: Shows the y axis value (Storage Level).
- Minimum Level Y Axis: If this checkbox is selected, it will show the Y Axis of the Minimum Level Storage Level
- Maximum Level Y Axis: If this checkbox is selected, it will show the Y Axis of the Maximum Level Storage Level
- Minimum Safe Level Y Axis: If this checkbox is selected, it will show the Y Axis of the Minimum Safe Level Storage Level
- Maximum Safe Level Y Axis: If this checkbox is selected, it will show the Y Axis of the Maximum Safe Level Storage Level
- Show Markers: Each series will be marked with a circle
- Legend position: The position of the legend in the chart
- Y axis Min: The minimum value of the Y axis to be displayed on the chart
- Y axis Max: The maximum value of the Y axis to be displayed on the chart
- Animate: Will show animation of the line being drawn
- Show Point Labels: Each series will be labeled with the storage level value

#### Lifting Account Entitlement Chart:

This dashboard line widget displays the entitlement per lifting account.



The parameters which can be setup for Lifting Account Entitlement Chart is shown in the screenshot below:

This dialog box allows users to configure various parameters for the chart:

- From Date:** Set to 2011-01-01.
- To Date:** Set to 2011-01-31.
- Lifting Account:** Set to TS1\_CRUDE\_CHEV\_CHE.
- Animate:** Unchecked.
- Show Markers:** Unchecked.
- Show Point Labels:** Checked.
- Show X axis:** Checked.
- Show Y axis:** Checked.
- Y axis Min:** Set to 0.
- Y axis Max:** Set to 140,000.00.
- Legend Position:** Set to Inside top-right of axis.

At the bottom are two buttons: **OK** and **CANCEL**.

The Lifting Account Entitlement Chart widget can be filtered by From Date, To Date and Lifting Account.

The function of each chart parameters is defined below:

- **Animate** – Will show animation of the line being drawn
- **Show X Axis** – shows the x axis and its label (Daytime).
- **Show Y Axis** - Shows the y axis value (Closing Balance).
- **Show Markers** – Each series will be marked with a circle

- Show point labels – Each series will be labeled with the closing balance
- Y axis Min – The minimum value of the Y axis to be displayed on the chart
- Y axis Max – The maximum value of the Y axis to be displayed on the chart
- Legend position – The position of the legend in the chart

**Logbook:**

This dashboard table widget shows the logbook entries entered by TCC user. This logbook widget is read-only, therefore it is not possible to enter logbook entries from the dashboard.

LOGBOOK				
DAYTIME	USER	LOG ENTRY	ALARM	ARCHIVED
2015-01-01 00:00	sysadmin	No need to fax DPR today	<input type="checkbox"/>	<input checked="" type="checkbox"/>

The parameters which can be setup for Logbook is shown in the screenshot below:

**LOGBOOK**

Parameters	
From Date	2015-01-01 <input type="button" value="..."/>
To Date	2015-02-01 <input type="button" value="..."/>
Functional Area	Transport Dispatching <input type="button" value="..."/>
Archived	<input checked="" type="checkbox"/>
OK	CANCEL

This logbook widget can be filtered by From Date, To Date, Functional Area and Archived.

**Operational Restrictions:**

This dashboard table widget is used to view capacity restriction on operational restriction..

RESTRICTION						
DAYTIME	RESTRICTION NAME	CLASS NAME	UOM	DESIGN CAPACITY	RESTRICTED CAPACITY	
2011-01-02	TS3 Norway	Pipeline	D Therms	2,000.00	100.00	
2011-01-02	TS3_BODØ	Delivery Point	D Therms	800.00	200.00	
2011-01-02	TS3_STAVANGER	Delivery Point	D Therms	800.00	500.00	
2011-01-02	TS3_TRD_ENTRY_STREAM	Delivery Stream	D Therms	900.00	100.88	
2011-01-03	TS3_BODØ	Delivery Point	D Therms	800.00	200.66	
2011-01-03	TS3_TRD_ENTRY_STREAM	Delivery Stream	D Therms	900.00	400.00	
2011-01-04	TS3_TRD_ENTRY_STREAM	Delivery Stream	D Therms	900.00	400.00	

The parameters which can be setup for Operational Restriction is shown in the screenshot below:

**RESTRICTION**

Parameters

From Date	2011-01-01
To Date	2011-02-01
Business Unit	TS3 BU1

**OK** **CANCEL**

This operational restriction widget can be filtered by From Date, To Date and Business Unit.

## EC Revenue

This chapter contains the Technical Documentation for EC Revenue:

- EC Revenue System Attributes
- EC Sales - EC Revenue Interface
- EC Revenue Dashboards

## EC Revenue System Attributes

EC Revenue uses a number of System Attributes to control system behavior. This document explains these system attributes.

For a new EC installation all of these should be checked and set according to the required behavior of EC Revenue.

For more details on each System Attribute, please see descriptions below the following overview table:

### **System Attribute 'FCST\_QTY\_ADJ\_SI\_CAT'**

This attribute controls the Stream Item Category for EC Revenue Quantity Forecast 'Prior Year Adjustment Stream Item'.

The Default EC installation value is 'PYA\_ADJ'.

The popup for selecting 'Prior Year Adjustment Stream Item' will only list Stream Items where the Stream Item Category matches this setting

### **System Attribute 'FX\_TRIANGULATE\_CURR\_CODE'**

This attribute controls which currency to use as basis for calculating exchange rates through 'Triangulation'.

The Default EC installation value is 'EUR'

### **System Attribute 'GROUP\_CURRENCY\_CODE'**

This attribute defines the Group currency for the system. Financial transaction monetary values will be converted to this currency if it has been set.

If leaving it blank there will be no Group Currency values calculated by the system.

Default EC installation value is 'USD'.

### **System Attribute 'REP\_FIELD\_GROUP\_TYPE'**

This attribute define the Field Group Type used for reporting.

Please note that there are no canned EC reports using this attribute currently.

### **System Attribute 'USE\_PHYSICAL\_STOCK'**

This attribute controls if the Inventory object should have the option to have 'Physical Stock' configuration.

If set to 'N' all references to Physical Stock are removed from the EC Business Functions.

Default EC installation value is 'Y'.

### **System Attribute 'ACC\_REV\_INTERFACE\_IND'**

This attribute controls whether to create an interface file to the ERP system for Accrual Reversals or not.

Options are Y | N

Default EC installation value is 'Y'

### **System Attribute 'ACC\_REV\_VALIDATION\_LEVEL'**

This attribute indicates the default validation level for the Accrual Reversal document.

Options are OPEN | VALID1 | VALID2 | TRANSFER | BOOKED

Default EC installation value is 'OPEN'

### **System Attribute 'DEFAULT\_REVERSAL\_CODE'**

This attribute holds the default reversal code for interfacing reversals to SAP.

Default EC installation value is '05'.

### **System Attribute 'DOC\_GEN\_LOG\_LEVEL'**

Default log level for document generation in EC Revenue. Valid choices: SUMMARY, INFO or DEBUG.

Default EC installation value is 'INFO'

#### **System Attribute 'DOC\_GEN\_MONTHS'**

If set, the document processing will look for interfaced data back to the number of months, relative to today. Number must be positive. If not set the system will get interfaced data for the last ten years.

Default EC installation value is NULL (blank)

#### **System Attribute 'DEFAULT\_REVERSAL\_DATE'**

This attribute holds the default date of the following booking period to do automatic reversals of Accruals.

It should typically be the same date as the date when the automatic accrual reversals are done in the ERP system

Default EC installation value is '1', i.e. the 1<sup>st</sup> of the month.

#### **System Attribute 'VAT\_REG\_NO\_MANDATORY\_IND'**

This attribute indicates the default validation for VAT Registration Number when setting document level to Valid1.

If this attribute is set to 'Y' the document validation will check that there is a VAT Registration Number set for the vendor/customer

Values are Y | N.

Default EC installation value is 'Y'

#### **System Attribute 'SELECT\_BOOKING\_PERIOD'**

This attribute indicates whether to use selectable booking period or not, Options are Y | N.

Default EC installation value is 'N'

If set to 'Y' then the user can select the booking period to use when booking a Financial Transaction document.

The default booking period will be the 'oldest open booking period', but the user can pick a different booking period that is open.

#### **System Attribute 'DEL\_DOC\_W\_DOCNUM'**

This attribute indicates if it is allowed to delete document with Invoice Number assigned, Options are Y | N.

Default EC installation value is 'Y'

#### **System Attribute 'INV\_4EYE\_VALIDATION'**

This attribute indicates whether to have 5-level / 2-user validation for Inventory or not, Options are Y | N.

If set to 'Y' then the validation hierarchy for the Inventories is: OPEN à VALID1 à VALID2 à TRANSFER à BOOKED where the transition from VALID1 to VALID2 has to be done by a different user than the user that took the Inventory to VALID1 (4-eyes validation)

Default EC installation value is 'N'

#### **System Attribute 'ALWAYS\_GEN\_POSTING'**

This attribute indicates whether the system will always generate posting data, Options are Y | N.

If set to 'Y' the system will generate posting data also for vendors for which there will be no interfacing of postings to the ERP system.

Default EC installation value is 'N'.

#### **System Attribute 'DEFAULT\_BOOKING\_PERIOD'**

This attribute indicates whether the system select default booking period based on document date or the 'oldest open booking period'.

If set to 'BY\_DOC\_DATE' the system will pick the booking period corresponding to the Document Date of the invoice.

Options are BY\_DOC\_DATE | OLDEST\_OPEN.

Default EC installation value is 'OLDEST\_OPEN'

#### **System Attribute 'RESTRICT\_VENDORS'**

This attribute indicates whether to show only Vendors connected to the contract owner company or show all Vendors in the Vendor selection popups

Vendors are connected to a specific Contract Owner Company through the 'Restricted Vendor Setup' Business Function.

Options are Y | N.

Default EC installation value is 'Y'

#### **System Attribute 'RESTRICT\_CUSTOMERS'**

This attribute indicates whether to show only Customers connected to the contract owner company or show all Customers in the Customer selection popups

Customers are connected to a specific Contract Owner Company through the 'Restricted Customer Setup' Business Function.

Options are Y | N.

Default EC installation value is 'Y'.

#### **System Attribute 'CASCADE\_SI\_CHANGE'**

This attribute indicates whether immediate cascade will be ran when changing relevant configuration on stream item (i.e. formulas, calculation methods, split key references etc.).

Options are AUTO | MANUAL | NONE.

Default EC installation value is 'AUTO'

When this attribute is set to 'AUTO' the system will recalculate Stream Item values immediately for all Stream Items that are affected by the change of the Stream Item configuration.

When this attribute is set to 'MANUAL' the system will add a reference in the 'Pending SI Calculation' table to all Stream Items that are affected by the change of the Stream Item configuration. The user can then go to the Business Functions 'Daily SI Pending Calculation' / 'Monthly SI Pending Calculation' / 'Forecast SI Pending Calculation' and manually kick off the recalculation of the Stream Item values, or set them to 'Ignore'.

When this attribute is set to 'NONE' the system will NOT recalculate Stream Item values for any Stream Items that are affected by the change of the Stream Item configuration.

#### **System Attribute 'CASCADE\_SK\_SHARE\_CHANGE'**

This attribute indicates whether immediate cascade will be ran when the Split Shares are changing for a Split Key that is used by a Split Key type of Stream Item.

Options are AUTO | MANUAL | NONE.

Default EC installation value is 'AUTO'

When this attribute is set to 'AUTO' the system will recalculate Stream Item values immediately for all Stream Items that are affected by the change of the Split Key Shares.

When this attribute is set to 'MANUAL' the system will add a reference in the 'Pending SI Calculation' table to all Stream Items that are affected by the change of the Split Key Shares. The user can then go to the Business Functions 'Daily SI Pending Calculation' / 'Monthly SI Pending Calculation' / 'Forecast SI Pending Calculation' and manually kick off the recalculation of the Stream Item values, or set them to 'Ignore'.

When this attribute is set to 'NONE' the system will NOT recalculate Stream Item values for any Stream Items that are affected by the change of the Split Key Shares.

#### **System Attribute 'CASCADE\_CONV\_FACT\_CHANGE'**

This attribute indicates whether immediate cascade will be ran when the Conversion Factors are changing.

Options are AUTO | MANUAL | NONE.

Default EC installation value is 'AUTO'

When this attribute is set to 'AUTO' the system will recalculate Stream Item values immediately for all Stream Items that are using the Conversion Factor being changed.

When this attribute is set to 'MANUAL' the system will add a reference in the 'Pending SI Calculation' table to all Stream Items that are using the Conversion Factor being changed. The user can then go to the Business Functions 'Daily SI Pending Calculation' / 'Monthly SI Pending Calculation' / 'Forecast SI Pending Calculation' and manually kick off the recalculation of the Stream Item values, or set them to 'Ignore'.

When this attribute is set to 'NONE' the system will NOT recalculate Stream Item values for any Stream Items that are affected by the Conversion Factor.

#### **System Attribute 'CASCADE\_BOE\_FACT\_CHANGE'**

This attribute indicates whether immediate cascade will be ran when the BOE Conversion Factors are changing.

Options are AUTO | MANUAL | NONE.

Default EC installation value is 'AUTO'

When this attribute is set to 'AUTO' the system will recalculate Stream Item values immediately for all Stream Items that are using the BOE Conversion Factor being changed.

When this attribute is set to 'MANUAL' the system will add a reference in the 'Pending SI Calculation' table to all Stream Items that are using the BOE Conversion Factor being changed. The user can then go to the Business Functions 'Daily SI Pending Calculation' / 'Monthly SI Pending Calculation' / 'Forecast SI Pending Calculation' and manually kick off the recalculation of the Stream Item values, or set them to 'Ignore'.

When this attribute is set to 'NONE' the system will NOT recalculate Stream Item values for any Stream Items that are affected by the BOE Conversion Factor.

#### **System Attribute 'VAT\_ON\_ACCRUALS'**

This attribute indicates whether the system should calculate VAT for accruals.

Options are Y | N

Default EC installation value is 'Y'

#### **System Attribute 'ALIGN\_REPORTING\_BOOKING'**

This attribute indicates whether Booking Period and Reporting Period should follow the same close/re-open process, i.e. when closing a given Booking Period then also the corresponding Reporting Period will be closed automatically.

Options are Y | N

Default EC installation value is 'N'

#### **System Attribute 'PROCESS\_ONLY\_OPEN\_DOCS'**

This attribute indicates whether validated documents should be protected from changes from document processing.

Options are Y | N

Default EC installation value is 'N'

#### **System Attribute 'JOURNAL\_ENTRY\_THRESHOLD'**

This attribute indicates the maximum number of journal entry records should be displayed.

Default EC installation value is 500.

#### **System Attribute 'JOURNAL\_MAP\_EXT\_TRAC\_IND'**

This attribute indicates whether to output tracing information when executing Journal Mapping and generating Journal Extract Documents.

Default EC installation value is 'Y'.

**System Attribute 'JOURNAL\_MAP\_TRAC\_MODE'**

This attribute indicates the mode in which journal mapping tracing reports should be generated. Options are FULL | WBS\_ACCOUNT | MAPPING\_CODE.

Default EC installation value is 'FULL'.

**System Attribute 'JOUR\_EXT\_PROC\_LOG\_LEVEL'**

This attribute set the default log level for Journal Extract process in EC Revenue. Valid choices: INFO or DEBUG.

Default EC installation value is 'INFO'.

**System Attribute 'JOURNAL\_MAP\_LOG\_LEVEL'**

This attribute set the default log level for Journal Mapping in EC Revenue. Valid choices: INFO or DEBUG.

Default EC installation value is 'INFO'.

**System Attribute 'HOLD\_FINAL\_WHEN\_ACL\_IND'**

This attribute indicates if to prevent the FINAL document interface data from being processed when there is an un-booked ACCRUAL document for the same processing period and document setup is found. Options are Y | N.

Default EC installation value is 'N'

**System Attribute 'ACNT\_LOGIC\_DATE\_METHOD'**

This attribute indicates which date is used to pick up fin\_account\_mapping, fin\_account, fin\_cost\_object\_mapping, fin\_cost\_object versions when generating posting data. Options are DOC\_DATE | TRANS\_DATE.

Use DOC\_DATE when the versions are decided by document date.

Use TRANS\_DATE when the versions are decided by transaction date.

Default EC installation value is TRANS\_DATE.

**System Attribute 'ORIG\_COUNTRY\_AT\_TRANS'**

This attribute indicates whether the Origin Country at the document transactions is used. Valid choices: Y | N.

Default EC installation value is 'Y'

**System Attribute 'DEST\_COUNTRY\_AT\_TRANS'**

This attribute indicates whether the Destination Country at the document transactions is used. Valid choices: Y | N.

Default EC installation value is 'Y'

**System Attribute 'DEF\_USE\_STREAM\_ITEMS\_IND'**

This attribute indicates the default value for "Use Stream Item Config" on a new transaction template. Valid choices: Y | N. Default EC installation value is 'Y'

**System Attribute 'DEF\_USE\_STREAM\_ITEMS\_IND'**

This attribute indicates the default value for "Use Stream Item Config" on a new transaction template. Valid choices: Y | N.

Default EC installation value is 'Y'

**System Attribute 'FIN\_INV\_VAL\_DIR'**

This attribute defines the directory at the JBoss server that will be used for storing the Inventory interface files. See package ECDP\_OutboundInterface for use of this attribute.

Note that this attribute is NOT added by the default EC installation.

**System Attribute 'FIN\_SALES\_DIR'**

This attribute defines the directory at the JBoss server that will be used for storing the Product Sales documents interface files. See package ECDP\_OutboundInterface for use of this attribute.

Note that this attribute is NOT added by the default EC installation.

#### **System Attribute ‘FIN\_PURCHASES\_DIR’**

This attribute defines the directory at the JBoss server that will be used for storing the Product Purchases documents interface files. See package ECDP\_OutboundInterface for use of this attribute.

Note that this attribute is NOT added by the default EC installation.

#### **System Attribute ‘FIN\_TA\_INCOME\_DIR’**

This attribute defines the directory at the JBoss server that will be used for storing the Tariff Income documents interface files. See package ECDP\_OutboundInterface for use of this attribute.

Note that this attribute is NOT added by the default EC installation.

#### **System Attribute ‘FIN\_TA\_COST\_DIR’**

This attribute defines the directory at the JBoss server that will be used for storing the Tariff Cost documents interface files. See package ECDP\_OutboundInterface for use of this attribute.

Note that this attribute is NOT added by the default EC installation.

#### **System Attribute ‘FIN\_JOU\_ENT\_DIR’**

This attribute defines the directory at the JBoss server that will be used for storing the Journal Entry documents interface files. See package ECDP\_OutboundInterface for use of this attribute.

Note that this attribute is NOT added by the default EC installation.

#### **System Attribute ‘FIN\_ERP\_DOC\_DIR’**

This attribute defines the directory at the JBoss server that will be used for storing the ERP documents interface files. See package ECDP\_OutboundInterface for use of this attribute.

Note that this attribute is NOT added by the default EC installation.

#### **System Attribute ‘CUST\_IN\_BOOK\_REPORT’**

This attribute is used for enabling the customized part of the database views used by the Inventory Booking Control Report. See database view RV\_INV\_BOOK\_CTRL REP\_FULL for more details.

Note that this attribute is NOT added by the default EC installation.

#### **System Attribute ‘ERP\_INTERFACE\_CLASS’**

This attribute defines the ERP business function class used in the TransferInventory and TransferSP java files.

Note that this attribute is NOT added by the default EC installation.

#### **System Attribute ‘DEFAULT\_EC\_RPT\_SET\_DATE’**

This is a default date needed for creating the Excel report for the ‘Cost Mapping Tracing Report’ and ‘Summary Tracing Report’.

If not set the system defaults to ‘2010-01-01’

Note that this attribute is NOT added by the default EC installation.

#### **System Attribute ‘REP\_PROFIT\_CENTER\_C\_TYPE’**

This attribute is no longer in use by the system.

Default EC installation value is ‘REPT’.

#### **System Attribute ‘FCST\_QTY\_STREAM\_CAT’**

This attribute controls the Stream Category for selecting EC Revenue Quantity Forecast Stream Items.

NOTE! This attribute is no longer in use by the system.

**System Attribute 'ENABLE\_DOC\_TRACING'**

This attribute is set to 'Y' if you want the Document/Calculation Tracing to activated.

## EC Sales - EC Revenue Interface

### New EcBp\_Replicate\_Sales\_Qty package

This document describes the functionality of the new **EcBp\_Replicate\_Sales\_Qty** package delivered in EC-11.1. The package has been completely reworked and will work directly out-of-the-box when certain requirements are met.

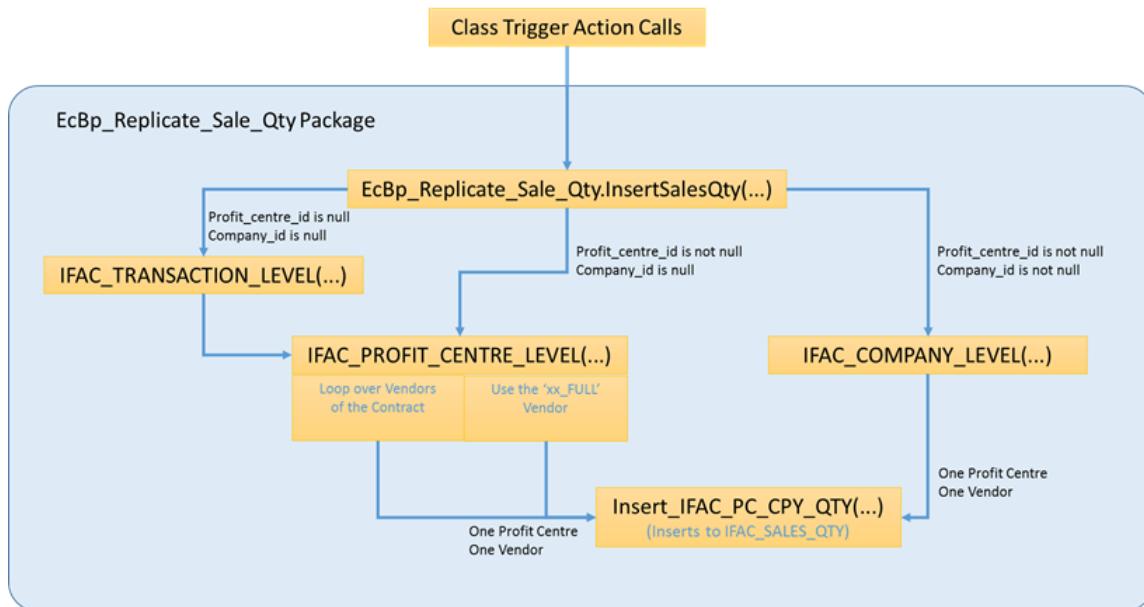
All the important procedures and functions within the new package have User Exit support through the **ue\_Replicate\_Sales\_Qty** package.

The new package can receive Contract Account data from these levels:

- Contract Account Level
  - No Profit Centre set
  - No Company set
  - Classes:
    - DV\_SCTR\_ACC\_MTH\_STATUS
    - DV\_SCTR\_ACC\_YR\_STATUS
- Contract Account / Profit Centre Level
  - Profit Centre set
  - No Company set
  - Classes:
    - DV\_SCTR\_ACC\_MTH\_PC\_STATUS
    - DV\_SCTR\_ACC\_YR\_PC\_STATUS
- Contract Account / Profit Centre Level / Company Level
  - Profit Centre set
  - Company set
  - Classes:
    - DV\_SCTR\_ACC\_MTH\_PC\_CPY
    - DV\_SCTR\_ACC\_YR\_PC\_CPY

Note that none of the DAILY Contract Account Data is supported. Such data should be aggregated to a monthly level and stored at the ..MTH... classes.

The structure of the new EcBp\_Replicate\_Sales\_Qty package is as follows:



Please note that the **IFAC\_PROFIT\_CENTRE\_LEVEL** procedure can interface to the Profit Centre Level or the Company Level in the IFAC table:

- If the number of vendors in the contract = 1 (Single Vendor Contract): Interface to Company Level
- If Contract Attribute 'IFAC\_PC\_USE\_FULL\_VENDOR' = 'Y' | NULL and the number of vendors in the contract > 1: Interface to Profit Centre
- If Contract Attribute 'IFAC\_PC\_USE\_FULL\_VENDOR' = 'N': Interface to Company Level

## **EcBp\_Replicate\_Sale\_Qty.InsertSalesQty Procedure**

This is the entry-point for all Class Trigger Action calls for all the data classes that are configured for interfacing quantity data into the IFAC\_SALES\_QTY table.

The procedure makes some initial checks to ensure that IFAC interfacing should take place:

- Contract is 'Available in EC Revenue' (ec\_contract.revn\_ind = Y)
- Contract Account has been set to do 'Interface to EC Revenue (ec\_contract\_account.interface\_to\_revenue = Y)

The procedure has User Exit support through the UE\_Replicate\_Sale\_Qty.ue\_insertSalesQty(...) procedure.

This procedure evaluates the call and makes subsequent call to the correct procedure:

- If profit\_centre\_id is null and company\_id is null à call IFAC\_TRANSACTION\_LEVEL
- If profit\_centre\_id is not null and company\_id is null à call IFAC\_PROFIT\_CENTRE\_LEVEL
- If profit\_centre\_id is not null and company\_id is not null à call IFAC\_COMPANY\_LEVEL

## **EcBp\_Replicate\_Sale\_Qty.IFAC\_TRANSACTION\_LEVEL Procedure**

This procedure is called when *p\_profit\_centre\_id is null* and *p\_company\_id is null* in the incoming call to the EcBp\_Replicate\_Sale\_Qty.InsertSalesQty procedure.

The quantities sitting in the Contract Account data classes have not been split to Profit Centre nor Company.

Supported data classes are:

- DV\_SCTR\_ACC\_MTH\_STATUS
- DV\_SCTR\_ACC\_YR\_STATUS

These quantities will be interfaced to the IFAC\_SALES\_QTY table if the contract in question is a truly **Single Profit Centre contract** – and then the quantities will be allocated 100% to that Profit Centre.

Error messages will be issued if the contract in question is not a truly Single Profit Centre contract, i.e. if all the Transaction Templates of the contract are referring to ONE and ONLY ONE Profit Centre.

When the Profit Centre has been determined the procedure forwards the call to procedure IFAC\_PROFIT\_CENTRE\_LEVEL with the Profit Centre sat.

The contract may have one or more Vendors defined where each will get their share of the quantities according to their share – see procedure IFAC\_PROFIT\_CENTRE\_LEVEL.

The procedure has User Exit support through the UE\_Replicate\_Sale\_Qty.ue\_IFAC\_TRANSACTION\_LEVEL(...) procedure.

## **EcBp\_Replicate\_Sale\_Qty.IFAC\_PROFIT\_CENTER\_LEVEL Procedure**

This procedure is called when *p\_profit\_centre\_id is not null* and *p\_company\_id is null* in the incoming call to the EcBp\_Replicate\_Sale\_Qty.InsertSalesQty procedure (or from the EcBp\_Replicate\_Sale\_Qty.IFAC\_TRANSACTION\_LEVEL procedure).

The quantities sitting in the Contract Account data classes have been split to Profit Centre but not Company.

Supported data classes are:

- DV\_SCTR\_ACC\_MTH\_PC\_STATUS
- DV\_SCTR\_ACC\_YR\_PC\_STATUS

This procedure can either interface to the Profit Centre Level of the IFAC or the Company Level of the IFAC.

The rules for deciding are:

1. If the number of Vendors in the contract is 1 – Single Vendor Contract:  
The logic will find this Vendor and do the interface to the Company Level in the IFAC table
2. If the number of Vendors in the contract is > 1 – Multi Vendor Contract:
  - a. If Contract Attribute 'IFAC\_PC\_USE\_FULL\_VENDOR' = 'Y':  
Interface to the Profit Centre Level by using the '<xx>\_FULL' vendor where <xx> is the Country Code of the Country of the Contract Owner Company.  
In this case the distribution to the Company Level will be done by the Document Processing logic in EC Revenue.
  - b. If Contract Attribute 'IFAC\_PC\_USE\_FULL\_VENDOR' = NULL (not defined):  
Interface to the Profit Centre Level by using the '<xx>\_FULL' vendor where <xx> is the Country Code of

the Country of the Contract Owner Company.

In this case the distribution to the Company Level will be done by the Document Processing logic in EC Revenue.

- c. If Contract Attribute 'IFAC\_PC\_USE\_FULL\_VENDOR' = 'N':

Interface to the Company Level by finding the individual Vendors of the contract and the corresponding share and loop over these doing IFAC inserts for each Vendor.

Note that using the <xx>\_FULL vendor or not may have some rounding implications:

- Use the <xx>\_FULL Vendor:  
The values ending up at the Profit Centre Level in the EC Revenue document is interfaced directly and then split to vendors.  
This will ensure that the value at the Profit Centre at the EC Sales side is the same as what you will have for the Profit Centre level at the EC Revenue side:

EC Sales / IFAC Data			EC Revenue Data		
	Quantity	Interfaced to IFAC		Profit Center Level	10
Profit Center Level	10	10		Profit Center Level	10
				Split to Vendors	Share Not Rounded
				Vendor 1	0,3335 3,335
				Vendor 2	0,1233 1,233
				Vendor 3	0,2433 2,433
				Vendor 4	0,2999 2,999
				Totals	1 10,000

- Not use <xx>\_FULL Vendor:  
The EC Sales Profit Centre values are split to vendors BEFORE being interfaced. The values ending up at the Profit Centre Level in the EC Revenue document is the sum of the values split to Vendor level. **If there is rounding applied in this process the values split out and added back together may not be the same as the starting point:**

EC Sales / IFAC Data			EC Revenue Data		
	Quantity				
Profit Center Level	10				9,9
Split to Vendors	Share	Not Rounded	Rounded to 1 decimal		
Vendor 1	0,3335	3,335	3,3		3,3
Vendor 2	0,1233	1,233	1,2		1,2
Vendor 3	0,2433	2,433	2,4		2,4
Vendor 4	0,2999	2,999	3,0		3,0
Totals	1	10,000	9,9		9,9

The procedure has User Exit support through the UE\_Replicate\_Sale\_Qty.ue\_IFAC\_PROFIT\_CENTRE\_LEVEL(...) procedure.

#### EcBp\_Replicate\_Sale\_Qty.IFAC\_COMPANY\_LEVEL Procedure

This procedure is called when *p\_profit\_center\_id is not null* and *p\_company\_id is not null* in the incoming call to the EcBp\_Replicate\_Sale\_Qty.InsertSalesQty procedure.

The quantities sitting in the Contract Account data classes have been split to Profit Centre AND Company.

Supported data classes are:

- DV\_SCTR\_ACC\_MTH\_PC\_CPY
- DV\_SCTR\_ACC\_YR\_PC\_CPY

Note that the Company reference coming into this procedure is a COMPANY object – but the IFAC table needs VENDOR objects. Because of this there is logic in this procedure that will try to find the Vendor that corresponds to the Company through the company\_id reference of the Vendor class. Multiple Vendors may be linked to the same company – so the Vendor to use must also be defined as a Vendor in the contract in question.

If there is **only one** vendor found that corresponds to the company\_id of the call then this procedure will call the IFAC\_PC\_CPY\_QTY procedure with this Vendor. Note that the Vendor Share is set to 1 in this case as we assume the

quantities have already been allocated by company/vendor.

If there is no Vendor linked to the Company an error message will be issued.

If there are multiple Vendors in the Contract being linked to the same Company an error message will be issued.

The procedure has User Exit support through the UE\_Replicate\_Sale\_Qty.ue\_IFAC\_COMPANY\_LEVEL(...) procedure.

### **EcBp\_Replicate\_Sale\_Qty.IFAC\_PC\_CPY\_QTY Procedure**

This procedure is responsible for doing the actual inserts into the IFAC\_SALES\_QTY table. It is called with a given Profit Centre, Vendor, and Vendor Share. The inserts to the IFAC table can be done to two levels:

- The so-called 'Company Level', i.e. each insert has a distinct Profit Centre and a 'real', distinct Vendor
- The so-called 'Profit Centre Level', i.e. each insert has a distinct Profit Centre and the vendor used it the <xx>\_FULL vendor  
(where <xx> is the Country Code of the Country of the Contract Owner Company)

### **Rounding Logic**

The procedure supports rounding of the following values based on Contract Attribute settings:

- Contract Attribute 'QTY1\_ROUNDING' set to the number of decimals to round to à QTY1 to the IFAC will be rounded to this number of decimals. If this attribute is NULL no rounding will take place.
- Contract Attribute 'QTY2\_ROUNDING' set to the number of decimals to round to à QTY2 to the IFAC will be rounded to this number of decimals. If this attribute is NULL no rounding will take place.
- Contract Attribute 'QTY3\_ROUNDING' set to the number of decimals to round to à QTY3 to the IFAC will be rounded to this number of decimals. If this attribute is NULL no rounding will take place.
- Contract Attribute 'QTY4\_ROUNDING' set to the number of decimals to round to à QTY4 to the IFAC will be rounded to this number of decimals. If this attribute is NULL no rounding will take place.
- Contract Attribute 'AMOUNT\_ROUNDING' set to the number of decimals to round to à PRICING\_VALUE to the IFAC will be rounded to this number of decimals. If this attribute is NULL no rounding will take place.

Note that it is the value after applying the Vendor Share (p\_party\_share) that will be rounded:

```

IF In_ifac_qty_1 is not null THEN
    --Apply vendor share:
    In_ifac_qty_1 := In_ifac_qty_1*p_party_share;
    --Apply rounding if set:
    IF In_Qty1Rounding is not null THEN
        In_ifac_qty_1 := round(In_ifac_qty_1,In_Qty1Rounding);
    END IF;
END IF;
```

### **Transaction Template logic**

The procedure uses the ecdp\_inbound\_interface.GetIfacSalesQtyTT function to find a Transaction Template that fits the contract account data. There are two 'tries' for the call to the ecdp\_inbound\_interface.GetIfacSalesQtyTT function:

1. With UOM1\_CODE set
2. If no TT found then we try with UOM1\_CODE cleared out

From the Transaction Template found the UOM1\_CODE | UOM2\_CODE | UOM3\_CODE | UOM4\_CODE are extracted.

### **The IFAC Insert**

The following table lists all the data fields populated for the IFAC insert. Most of the fields to be inserted are assigned to a separate variable:

IFAC Data Field	Local Variable	Comments
-----------------	----------------	----------

contract_code	lv2_ifac_contract_code	Found from calling parameter p_object_id: ec_contract.object_code(p_object_id)
description	lv2_ifac_account_name	Account Name is passed on to IFAC.DESCRIPTION
processing_period	ld_ifac_processing_period	Set to 'First of Month' for monthly data. Set to 'First of Year' for yearly data
period_start_date	ld_ifac_period_start_date	Set to 'First of Month' for monthly data. Set to 'First of Year' for yearly data
period_end_date	ld_ifac_period_end_date	Set to 'Last of Month' for monthly data. Set to 'Last of Year' for yearly data
profit_center_code	lv2_ifac_profit_centre_code	Found from calling parameter p_profit_centre_id: ecdp_objects.GetObjCode(p_profit_centre_id)
delivery_point_code	lv2_ifac_del_pnt_code	Found from Contract Account setting. Error if missing.
price_object_code	lv2_ifac_price_object_code	Found from Contract Account setting. Can be NULL
price_concept_code	lv2_ifac_price_concept_code	Found from Price Object if this has been set on the Contract Account. Else found from the Contract Account directly. Either way – can't be NULL.
product_code	lv2_ifac_product_code	Found from Contract Account setting. Error if missing.
vendor_code	lv2_ifac_vendor_code	Found based on calling parameter p_vendor_id
qty1	ln_ifac_qty_1	Vendor share applied. Rounding applied if set.
uom1_code	lv2_TT_uom1_code	The UOM1_CODE of the Transaction Template found
qty2	ln_ifac_qty_2	Vendor share applied. Rounding applied if set.
uom2_code	lv2_TT_uom2_code	The UOM2_CODE of the Transaction Template found
qty3	ln_ifac_qty_3	Vendor share applied. Rounding applied if set.
uom3_code	lv2_TT_uom3_code	The UOM3_CODE of the Transaction Template found
qty4	ln_ifac_qty_4	Vendor share applied. Rounding applied if set.
uom4_code	lv2_TT_uom4_code	The UOM4_CODE of the Transaction Template found
pricing_value	ln_ifac_pricing_value	Vendor share applied. Rounding applied if set.

unit_price	In_ifac_unit_price	Calculated as In_ifac_pricing_value/ In_ifac_qty_1 before applying party_share and rounding
qty_status	lv2_ifac_qty_status	Picked up from Contract Account – if this is NULL then set to 'FINAL'
doc_status	lv2_ifac_doc_status	Picked up from Contract Attribute 'DOC_FINAL_ACCRUAL_SETTING' – if this is NULL then set to 'FINAL'.  Note that this has UE Support!
object_type	lv2_ifac_object_type	Found from calling parameter p_profit_centre_id: ecdp_objects.GetObjClassName(p_ profit_centre_id)
dist_type	lv2_ifac_dist_type	Hardcoded to 'OBJECT' because the IFAC insert is always at Company Level
status	'NEW'	Hardcoded to 'NEW'
contract_account_class	lv2_ifac_account_class_name	Given by the call through parameter p_class_name
calc_run_no	In_ifac_calc_run_no	Found from Contract Account data field 'CALC_RUN_NO'
contract_account	lv2_ifac_contract_account	Given by the call through parameter p_account_code
created_by	p_user	The user doing the insert.

## Quantity Mapping

When quantities from the EC Sales Contract Account data classes are moved into the EC Revenue space it is important that these quantities end up in the right place.

In EC Sales the quantities are stored in these columns

- VOL\_QTY
- MASS\_QTY
- ENERGY\_QTY
- X1\_QTY
- X2\_QTY
- X3\_QTY

The UOMs for these quantities are defined here:

- VOL\_QTY: Contract Attribute 'DEL\_VOL\_UOM'
- MASS\_QTY: Contract Attribute 'DEL\_VOL\_UOM'
- ENERGY\_QTY: Contract Attribute 'DEL\_VOL\_UOM'
- X1\_QTY: Attribute 'X1\_UOM'
- X2\_QTY: Attribute 'X2\_UOM'
- X3\_QTY: Attribute 'X3\_UOM'

The EC Revenue contracts operates with 4 quantity columns:

- QTY\_1
- QTY\_2
- QTY\_3
- QTY\_4

The UOMs for these quantities are defined at the Transaction Templates:

- UOM1\_CODE
- UOM2\_CODE
- UOM3\_CODE
- UOM4\_CODE

The UOMs on the EC Sales side may not correspond directly to the UOMs at the EC Revenue side. Even if there are quantities in various UOMs on the EC Sales side maybe only a subset of these should be interfaced to EC Revenue.  
 (EC Sales Contract Account has SM3, GJ, and MT – but only the SM3 number is to be interfaced.)  
 And there are also cases where a single quantity in the EC Sales side is used as basis for two or more quantities on the EC Revenue side.  
 (MJ quantity in EC Sales interfaced to EC Revenue as GJ and KWH.)

New functionality has been added to the EC Sales Contract Account configuration where it is possible for each Contract Account for a given Contract to map the EC Sales quantities to the QTY1-4 quantities in EC Revenue:

Account Code	Account Name	Start Date	End Date	Interface to EC Revenue	Revenue Mapping QTY1	Revenue Mapping QTY2	Revenue Mapping QTY3	Revenue Mapping QTY4	Quantity Status	Count
NORMAL_BUTANE	Normal Butane	2010-01-01		Yes	Volume Qty	Energy Qty	Mass Qty	Extra 1 Qty		
EXCESS_BUTANE	Excess Butane	2010-01-01		Yes	Volume Qty	Energy Qty	Mass Qty	Extra 1 Qty		
SHORTFALL_BUTANE	Shortfall Butane	2010-01-01		Yes	Volume Qty	Energy Qty	Mass Qty	Extra 1 Qty		
NORMAL_PROPANE	Normal Propane	2010-01-01		Yes	Volume Qty	Energy Qty	Mass Qty	Extra 1 Qty		
EXCESS_PROPANE	Excess Propane	2010-01-01		Yes	Volume Qty	Energy Qty	Mass Qty	Extra 1 Qty		
SHORTFALL_PROPANE	Shortfall Propane	2010-01-01		Yes	Volume Qty	Energy Qty	Mass Qty	Extra 1 Qty		

Event Type      UOM      Mandatory Of

No records found.

The advantage of this new functionality is that it is much easier for the interface logic to pick the right quantities for interfacing to the IFAC table.

The interface logic also supports unit conversion when the UOM in EC Sales is different from the UOM in EC Revenue.

Example:

- EC Sales: DEL\_VOL\_UOM = SM3
- EC Sales VOL\_QTY = 1000 SM3
- EC Revenue: UOM1 = NM3
- Revenue Mapping QTY1 = Volume Qty
- Volume interfaced to IFAC\_SALES\_QTY table = 94,79167 NM3

### Function QtyUOMMapping

This function within the EcBp\_Replicate\_Sales\_Qty is doing the actual mapping between the UOMs. It returns the UOM on the EC Sales side based on the mapping. Example:

- 'Revenue Mapping QTY1' is set to 'Volume Quantity'
- Contract Attribute 'DEL\_VOL\_UOM' = SM3
- The return value form this function is 'SM3' telling that quantity sitting in VOL\_QTY is given in SM3.
- SM3 will be the 'From Unit' when doing the ecdp\_unit.ConvertValue
- Assume the Transaction Template found in EC Revenue has UOM1\_CODE = NM3.  
 This will be the 'To Unit' when doing the ecdp\_unit.ConvertValue

This function handles all the 6 variants of the 'Revenue Mapping':

- Volume Qty – Code 'VOL\_QTY'
- Mass Qty – Code 'MASS\_QTY'
- Energy Qty – Code 'ENERGY\_QTY'
- Extra 1 Qty – Code 'X1\_QTY'
- Extra 2 Qty – Code 'X2\_QTY'
- Extra 3 Qty – Code 'X3\_QTY'

This function is called 4 times from the EcBp\_Replicate\_Sale\_Qty.IFAC\_PC\_CPY\_QTY procedure – one for each of the QTY1 | QTY2 | QTY3 | QTY4.

If there is no mapping for one of these then the function returns NULL.

This function has error handling for the case where a Revenue Mapping has been defined and the corresponding UOM is missing.

Example:

- 'Revenue Mapping QTY3' is set to 'Extra 1 Qty' (Code = X1\_UOM)
- Contract Account attribute 'X1\_UOM' is NULL



- The function will give an error message: '*The Contract Account has been set up to map the X1 Qty to EC Revenue, but there is no UOM defined for quantity X1. Please check the Contract Account config in screen Contract Account.*'

This function has also error handling in case the mapping is unknown - not very likely to happen....but still!

This function has User Exit functionality through **UE\_Replicate\_Sale\_Qty.ue\_QtyUOMMapping(...)**

### Function QtyValueMapping

This function within the EcBp\_Replicate\_Sales\_Qty returns the actual quantity value for the mapping between EC Sales and EC Revenue.

Example:

- 'Revenue Mapping QTY1' is set to 'Volume Quantity'
- The return value from this function is the VOL\_QTY quantity.

This function handles all the 6 variants of the 'Revenue Mapping':

- Volume Qty – Code 'VOL\_QTY'
- Mass Qty – Code 'MASS\_QTY'
- Energy Qty – Code 'ENERGY\_QTY'
- Extra 1 Qty – Code 'X1\_QTY'
- Extra 2 Qty – Code 'X2\_QTY'
- Extra 3 Qty – Code 'X3\_QTY'

This function has error handling for the case where a Revenue Mapping has been defined and the corresponding quantity is missing.

Example:

- 'Revenue Mapping QTY3' is set to 'Extra 1 Qty' (Code = X1\_UOM)
- The X1\_QTY is NULL
- The function will give an error message: *The Contract Account has been set up to map the X1 Qty to EC Revenue, but quantity X1 is null. Please check the Contract Calculation - or the config in screen Contract Account.*'

This function is called 4 times from the EcBp\_Replicate\_Sale\_Qty.IFAC\_PC\_CPY\_QTY procedure – one for each of the QTY1 | QTY2 | QTY3 | QTY4.

If there is no mapping for one of these then the function returns NULL.

This function has also error handling in case the mapping is unknown - not very likely to happen....but still!

This function has User Exit functionality through **UE\_Replicate\_Sale\_Qty.ue\_QtyValueMapping(...)**

### Function GetIfacRecordPriceObject

This function within the EcBp\_Replicate\_Sales\_Qty tries to find a Price Object connected to the contract in the case there is no Price Object defined at the Contract Account. The function will include Price Objects where Quantity Status and/or Price Status is null. If one of these or both are set at the Price Object then these Price Objects are prioritized. This function has UE support through **ue\_replicate\_sale\_qty.ue\_GetIfacRecordPriceObject()** funtion.

### Procedure ApproveCalc

If the Contract has the **calc\_approval\_check** attribute set to 'Y' then the Contract Account data will not be transferred to the IFAC table on the insert action, but rather when the Calculation behind the contract account data has been Approved. When appoving the Calculation the EcBp\_Replicate\_Sales\_Qty.ApproveCalc is executed to do the actual transfer of the Contract Account data into the IFAC table.

### Function UE\_Replicate\_Sale\_Qty.ue\_PriceStatus

Price Status is by default NOT part of an Contract Account setting – and the default handling of the Price Status attribute in the new EcBp\_Replicate\_Sales\_Qty package is to set it to NULL. The default can be overridden by using the **UE\_Replicate\_Sale\_Qty.ue\_PriceStatus** function – for instance by adding a project-specific attribute to the Contract Account holding the Price Status.

## EC Revenue Dashboards

This document describes the EC Revenue Dashboard widgets that comes with the standard EC product. The widgets are designed to be useful as-is, but they can also be used as an example for creating your own dashboards.

Each dashboard widget in EC must have a Query that will provide the data. This Query is stored in the QUERY record in the CTRL\_DASHBOARD\_PARAM table. The QUERY must be in XML format.

There are 3 different ways to define the query:

1. By accessing a data class directly – CLASS syntax
2. By creating a view in the database and then use the SQL syntax for the QUERY parameter with a SQL query that accesses the data in the view
3. By a SQL that gives the data directly

For the EC Revenue Dashboards described in this document we have mostly used variant 2. This variant gives good flexibility such as EC function calls and it is easy to change the where-conditions, etc.

For the Inventory value dashboards a new Class has been created – DASH\_INV\_VALUES – and the widgets access the data through this class

All the views defined are using this naming standard for the view definition: **V\_DASH\_<xxxx>**.

Each individual Dashboard definition is uniquely identified by a WIDGET\_CODE. All the EC Revenue Dashboard widgets are using this naming standard: **REVN\_<yyyy>**

### Financial Transactions

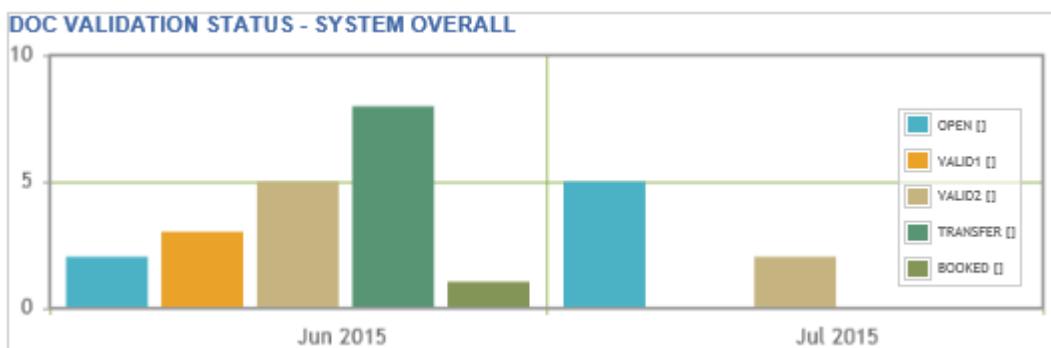
The dashboard widgets for the EC Revenue Financial Transaction business area are divided into two main categories:

- Counting of documents at the various Document Validation Levels:  
Open / Valid 1 / Valid 2 / Transfer / Booked
- Monetary values added up to various levels

For all of these there are a number of variants further described in the following sections.

#### ***Financial Transactions - Document Validation Status Dashboards***

The purpose of these dashboards is to give an overview of the number of documents at the various document validation levels. Typically it is important to make sure that all documents for the current booking period are set to booked before closing the booking period:



The Document Validation Status dashboards will have variants:

- All (showing all with no filtering on Contract Owner Company / Business Unit / Contract Area)
- By Contract Owner Company
- By Contract Area
- By Business Unit
- By Business Unit and Contract Owner Company
- By Contract Area and Contract Owner Company

### Document Date vs Booking Period

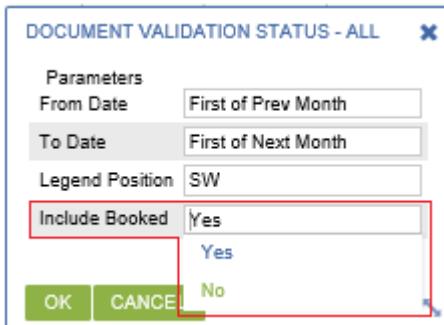
When grouping the documents into a given month this can be done either by Document Date or Booking Period. The result may be different for these two variants because the Document Date may be in a different calendar month compared to the Booking Period.

Documents at level Open / Valid 1 / Valid 2 have no Booking Period set as the Booking Period is set then the document is moved to Transfer. For these documents EC will predict the Booking Period by finding the Booking Period that would have been used if the document was taken to Transfer at the time. This logic also supports the case where system attribute

'DEFAULT\_BOOKING\_PERIOD' = 'BY\_DOC\_DATE', in which case the booking period will be the same as the Document Date. Also note that because of this additional check these dashboards may run a little bit slower than those using the Document Date.

#### Include Booked documents or not

The dashboards have functionality to include or exclude Booked documents in the dashboard:

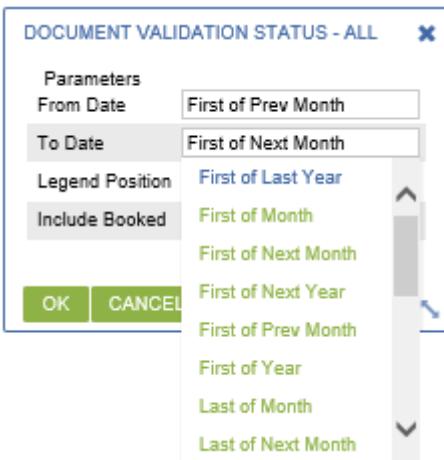


The reasons for this are:

- It is more important to get an overview of the number of documents that are NOT Booked – it is these documents you have to 'look after'
- If there is a big number of Booked documents and only a few documents at the other levels then the dashboard will not be able to show a bar for these due to the resolution of the Y-axis

#### Time Scope for the dashboards

The dashboards have 'From Date' and 'To Date' parameters defining the time scope for the data shown in the dashboard. These are using the pre-defined date methods given by the 'MACRO\_TYPE' EC Code:



By default the setting is:

- From Date = 'First of Prev Month'
- To Date = 'First of Next Month'

giving a 2-month window. For each individual dashboard you can change these settings to give a different time window.

#### Document Validation Status Dashboards

These are the Document Validation Dashboards defined:

Dashboard Name	Dashboard View Name
Document Validation Status - All - By Document Date	V_DASH_DOC_CNT_DD_ALL
Document Validation Status - Contract Owner Company - By Document Date	V_DASH_DOC_CNT_DD_CO
Document Validation Status - Contract Area - By Document Date	V_DASH_DOC_CNT_DD_CA
Document Validation Status - Business Unit - By Document Date	V_DASH_DOC_CNT_DD_BU
Document Validation Status - Business Unit and Contract Owner Company - By Document Date	V_DASH_DOC_CNT_DD_BU_CO
Document Validation Status - Contract Area and Contract Owner Company - By Document Date	V_DASH_DOC_CNT_DD_CA_CO
Document Validation Status - All - By Booking Period	V_DASH_DOC_CNT_BP_ALL
Document Validation Status - Contract Owner Company - By Booking Period	V_DASH_DOC_CNT_BP_CO
Document Validation Status - Contract Area - By Booking Period	V_DASH_DOC_CNT_BP_CA
Document Validation Status - Business Unit - By Booking Period	V_DASH_DOC_CNT_BP_BU
Document Validation Status - Business Unit and Contract Owner Company - By Booking Period	V_DASH_DOC_CNT_BP_BU_CO
Document Validation Status - Contract Area and Contract Owner Company - By Booking Period	V_DASH_DOC_CNT_BP_CA_CO

#### Document Validation Status Dashboard Views

The views used for these dashboard will create 5 records for each calendar month defined by the DAYTIME entries in the SYSTEM\_MTH\_STATUS table – which is the Booking Periods defined in the system. There will be one record for each of the 5 Document Validation Levels

(OPEN/VALID1/VALID2/TRANSFER/BOOKED).

If there are no documents at a given status for a given calendar month it is reported as 0:

SORT_ORDER	IS_DEFAULT	PERIOD_CHAR	PERIOD_DATE	DOC_COUNT	STATUS	INC_BOOKED	DASH_HEADER
50	Y	Jun 2015	01.06.2015	2	OPEN	N	...
10	N	Jun 2015	01.06.2015	3	VALID1	N	...
20	N	Jun 2015	01.06.2015	5	VALID2	N	...
30	N	Jun 2015	01.06.2015	8	TRANSFER	N	...
40	N	Jun 2015	01.06.2015	1	BOOKED	Y	...
50	Y	Jul 2015	01.07.2015	5	OPEN	N	...
10	N	Jul 2015	01.07.2015	0	VALID1	N	...
20	N	Jul 2015	01.07.2015	2	VALID2	N	...
30	N	Jul 2015	01.07.2015	0	TRANSFER	N	...
40	N	Jul 2015	01.07.2015	0	BOOKED	Y	...

The data is sorted in the view such that when retrieving the data into the dashboard no sorting is required.

The QUERY parameter for the dashboard looks like this:

```
<data>
<sql>
select
    x.sort_order as SORT_ORDER,
    x.is_default as IS_DEFAULT,
    x.PERIOD_CHAR as PERIOD_CHAR,
    x.period_date as PERIOD_DATE,
    x.DOC_COUNT as DOC_COUNT,
    x.status as STATUS,
    x.dash_header as DASH_HEADER,
    x.inc_booked as INC_BOOKED
from V_DASH_DOC_CNT_DD_BU x
where x.PERIOD_DATE >= trunc(to_date('$FROMDATE$', 'YYYY-MM-DD"T"hh24:mi:ss'), 'MM')
    and x.PERIOD_DATE < trunc(to_date('$TODATE$', 'YYYY-MM-DD"T"hh24:mi:ss'), 'MM')
    and x.BU_ID = '$BUSINESS_UNIT$'
    and '$INCL_BOOKED$' = 'Y'
union all
select
    x.sort_order as SORT_ORDER,
    x.is_default as IS_DEFAULT,
    x.PERIOD_CHAR as PERIOD_CHAR,
    x.period_date as PERIOD_DATE,
```

```

x.DOC_COUNT as DOC_COUNT,
x.status as STATUS,
x.dash_header as DASH_HEADER,
x.inc_booked as INC_BOOKED

from V_DASH_DOC_CNT_DD_BU x

where x.PERIOD_DATE >= trunc(to_date('$FROMDATE$', 'YYYY-MM-DD"T"hh24:mi:ss'), 'MM')
      and x.PERIOD_DATE < trunc(to_date('$TODATE$', 'YYYY-MM-DD"T"hh24:mi:ss'), 'MM')
      and x.BU_ID = '$BUSINESS_UNIT$'
      and x.inc_booked = 'N'
      and '$INCL_BOOKED$' = 'N'

</sql>

</data>

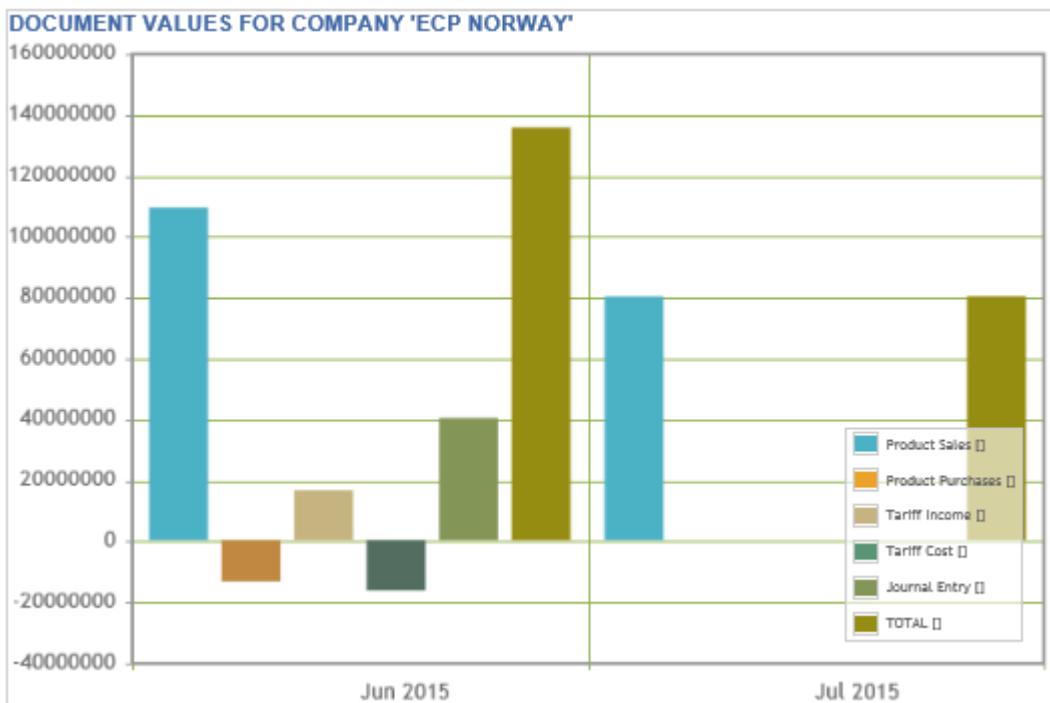
```

The various dashboard variants will have different parameters – in this example Business Unit is a parameter to the dashboard.

### **Financial Transactions – Monetary Value Dashboards**

The purpose of these dashboards is to give a monetary value monthly overview for the various Financial Transaction areas:

- Product Sales (Financial Code SALE)
- Product Purchases (Financial Code PURCHASE)
- Tariff Income (Financial Code TA\_INCOME)
- Tariff Cost (Financial Code TA\_COST)
- Journal Entry (Financial Code JOU\_ENT)



The Monetary Values Dashboards will have these variants:

- By Company – By Document Date and Group Currency

- By Contract Area – By Document Date and Group Currency
- By Business Unit – By Document Date and Group Currency
- By Company and Business Unit – By Document Date and Group Currency
- By Company and Contract Area – By Document Date and Group Currency

For each of the main variants there are dashboard variants as follows:

- By Document Date vs Booking Period
- By Group Currency vs Booking Currency

#### **By Document Date**

The Monetary Value Dashboards 'By Document Date' are showing data based on Document Date which are extracting data from documents at all of the Document Validation Levels (Open / Valid 1 / Valid2 / Transfer / Booked).

#### **By Booking Period**

The Monetary Value Dashboards 'By Booking Period' are showing data based on Booking Period extracting data only from documents at Document Validation Level 'Booked'.

#### **By Group Currency**

All monetary values are converted into the Group Currency if the EC installation has the Group Currency defined (system attribute 'GROUP\_CURRENCY\_CODE') and proper exchange rates are available. These dashboards are extracting date from the 'GROUP\_VALUE' column in the CONT\_LI\_DIST\_COMPANY table.

#### **By Booking Currency**

The monetary values can also be displayed in Booking Currency. As the Booking Currency can vary from contract to contract the user must select the booking currency to use for the dashboard. Note that for this case the dashboard will only show monetary values from documents where the Booking Currency matches the selected Booking Currency. These dashboards are extracting date from the 'BOOKING\_VALUE' column in the CONT\_LI\_DIST\_COMPANY table.

#### **Monetary values – Income vs Expense**

The monetary values for all of the Financial Transaction types are normally stored as positive numbers in EC. In EC Revenue the monetary values are normally stored as positive numbers for all the different types of Financial Transactions, i.e. Sales, Purchases, Tariff Income, Tariff Cost, and Journal Entry. When giving an overview of the monetary values for a given month the **expenses** will be presented as negative numbers. To achieve this all the monetary values for the Product Purchase and Tariff Cost are multiplied by (-1) in the views used by these dashboards. Then by adding all the numbers together we can then get a proper Total representing the Net Value for the month.

#### **Company Dimension**

All of the Monetary Value Dashboards are showing data by Legal Entity. Legal Entities in this context are the Companies in the Company table that have Customers or Vendors connected to them.

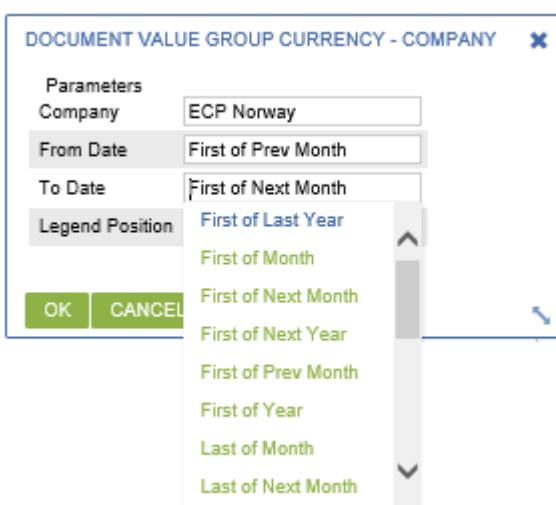
The data for these dashboards are all using the CONT\_LI\_DIST\_COMPANY table as basis. This table has values per Vendor for Financial Codes Sale / Tariff Income / Journal Entry and values per Customer for Financial Codes Purchase / Tariff Cost. Each of these Vendors and Customers have a connection to a Company – which is then the Legal Entity to show the monetary values for.

Each dashboard will have a parameter for setting the Company:



### Time Scope for the Monetary Value Dashboards

The dashboards have 'From Date' and 'To Date' parameters defining the time scope for the data shown in the dashboard. These are using the pre-defined date methods given by the 'MACRO\_TYPE' EC Code:



By default the setting is:

- From Date = 'First of Prev Month'
- To Date = 'First of Next Month'

giving a 2-month window. For each individual dashboard you can change these settings to give a different time window.

### Monetary Value Dashboards – the variants

These are the Monetary Value Dashboards defined for the Financial Transactions area

Dashboard Name	Dashboard View Name
Document Values - Document Date - Group Currency - Company	V_DASH_DOC_VAL_DD_GC
Document Values - Document Date - Group Currency – Company/Business Unit	V_DASH_DOC_VAL_DD_GC_BU
Document Values - Document Date - Group Currency – Company/Contract Area	V_DASH_DOC_VAL_DD_GC_CA
Document Values - Document Date - Booking Currency - Company	V_DASH_DOC_VAL_DD_BC
Document Values - Document Date - Booking Currency – Company/Business Unit	V_DASH_DOC_VAL_DD_BC_BU
Document Values - Document Date - Booking Currency – Company/Contract Area	V_DASH_DOC_VAL_DD_BC_CA
Document Values - Booking Period - Group Currency – Company	V_DASH_DOC_VAL_BP_GC
Document Values - Booking Period - Group Currency – Company/Business Unit	V_DASH_DOC_VAL_BP_GC_BU
Document Values - Booking Period - Group Currency – Company/Contract Area	V_DASH_DOC_VAL_BP_GC_CA
Document Values - Booking Period - Booking Currency - Company	V_DASH_DOC_VAL_BP_BC
Document Values - Booking Period - Booking Currency – Company/Business Unit	V_DASH_DOC_VAL_BP_BC_BU
Document Values - Booking Period - Booking Currency – Company/Contract Area	V_DASH_DOC_VAL_BP_BC_CA

### Monetary Value Dashboard Views

A number of views are created in order to provide the data the way we want it. The views are:

- V\_DASH\_DOC\_VAL\_DD  
This view extracts data from CONT\_LI\_DIST\_COMPANY **based on Document Date for all documents regardless of the Document Validation Level.** It extracts values for both Group Currency and Booking Currency. In this view the monetary values for Purchase and Tariff Cost are multiplied by (-1). The view also extracts the Business Unit IDs, Contract Area IDs, and Company IDs. The following sub-views are all using this view as basis:
  - V\_DASH\_DOC\_VAL\_DD\_GC  
This view will have 5 records for each combination of Month / Company.  
One record for each of the 5 Financial Codes. GROUP\_VALUE is the value extracted.
  - V\_DASH\_DOC\_VAL\_DD\_GC\_BU  
This view will have 5 records for each combination of Month / Company / Business Unit. One record for each of the 5 Financial Codes. GROUP\_VALUE is the value extracted
  - V\_DASH\_DOC\_VAL\_DD\_GC\_CA  
This view will have 5 records for each combination of Month / Company / Contract Area. One record for each of the 5 Financial Codes. GROUP\_VALUE is the value extracted.
  - V\_DASH\_DOC\_VAL\_DD\_BC  
This view will have 5 records for each combination of Month / Booking Currency / Company. One record for each of the 5 Financial Codes. BOOKING\_VALUE is the value extracted.
  - V\_DASH\_DOC\_VAL\_DD\_BC\_BU  
This view will have 5 records for each combination of Month / Booking Currency / Company / Business Unit. One record for each of the 5 Financial Codes. BOOKING\_VALUE is the value extracted
  - V\_DASH\_DOC\_VAL\_DD\_BC\_CA  
This view will have 5 records for each combination of Month / Booking Currency / Company / Contract Area. One record for each of the 5 Financial Codes. BOOKING\_VALUE is the value extracted.
- V\_DASH\_DOC\_VAL\_BP

This view extracts data from CONT\_LI\_DIST\_COMPANY **based on Booking Period for Documents that have been set to Booked**. It extracts values for both Group Currency and Booking Currency. In this view the monetary values for Purchase and Tariff Cost are multiplied by (-1). The view also extracts the Business Unit IDs, Contract Area IDs, and Company IDs. The following sub-views are all using this view as basis:

- V\_DASH\_DOC\_VAL\_BP\_GC  
This view will have 5 records for each combination of Month / Company.  
One record for each of the 5 Financial Codes. GROUP\_VALUE is the value extracted.
- V\_DASH\_DOC\_VAL\_BP\_GC\_BU  
This view will have 5 records for each combination of Month / Company / Business Unit. One record for each of the 5 Financial Codes. GROUP\_VALUE is the value extracted
- V\_DASH\_DOC\_VAL\_BP\_GC\_CA  
This view will have 5 records for each combination of Month / Company / Contract Area. One record for each of the 5 Financial Codes. GROUP\_VALUE is the value extracted.
- V\_DASH\_DOC\_VAL\_BP\_BC  
This view will have 5 records for each combination of Month / Booking Currency / Company. One record for each of the 5 Financial Codes. BOOKING\_VALUE is the value extracted.
- V\_DASH\_DOC\_VAL\_BP\_BC\_BU  
This view will have 5 records for each combination of Month / Booking Currency / Company / Business Unit. One record for each of the 5 Financial Codes. BOOKING\_VALUE is the value extracted
- V\_DASH\_DOC\_VAL\_BP\_BC\_CA  
This view will have 5 records for each combination of Month / Booking Currency / Company / Contract Area. One record for each of the 5 Financial Codes. BOOKING\_VALUE is the value extracted.

The following is an example of the output from these views:

SORT_ORDER	PERIOD_CHAR	PERIOD_DATE	DOC_VALUE	FIN_CODE_TEXT	CO_NAME	GROUP_CURRENCY_CODE	CO_ID
10 Jun 2015	01.06.2015	218411391,5	Product Sales	EC Petroleum Norway Vendor	USD	775F9861F671499B998A9838832422F7	
20 Jun 2015	01.06.2015	0	Product Purchases	EC Petroleum Norway Vendor	USD	775F9861F671499B998A9838832422F7	
40 Jun 2015	01.06.2015	32763087,48	Tariff Income	EC Petroleum Norway Vendor	USD	775F9861F671499B998A9838832422F7	
50 Jun 2015	01.06.2015	0	Tariff Cost	EC Petroleum Norway Vendor	USD	775F9861F671499B998A9838832422F7	
60 Jun 2015	01.06.2015	80304000	Journal Entry	EC Petroleum Norway Vendor	USD	775F9861F671499B998A9838832422F7	
10 Jul 2015	01.07.2015	160079941,4	Product Sales	EC Petroleum Norway Vendor	USD	775F9861F671499B998A9838832422F7	
20 Jul 2015	01.07.2015	0	Product Purchases	EC Petroleum Norway Vendor	USD	775F9861F671499B998A9838832422F7	
40 Jul 2015	01.07.2015	0	Tariff Income	EC Petroleum Norway Vendor	USD	775F9861F671499B998A9838832422F7	
50 Jul 2015	01.07.2015	0	Tariff Cost	EC Petroleum Norway Vendor	USD	775F9861F671499B998A9838832422F7	
60 Jul 2015	01.07.2015	78086,28	Journal Entry	EC Petroleum Norway Vendor	USD	775F9861F671499B998A9838832422F7	

SQL:

```
SELECT * FROM v_dash_doc_val_dd_gc
where co_id = '775F9861F671499B998A9838832422F7' -company 'ECP Norway'
and period_date >= to_date('2015-06-01','YYYY-MM-DD')
and period_date < to_date('2015-08-01','YYYY-MM-DD')
```

For each of the various dashboards there is a QUERY parameter using the corresponding view. The SQL typically looks like this:

```
<data>
<sql>
SELECT
    x.sort_order,
    x.PERIOD_CHAR,
    x.PERIOD_DATE,
    x.DOC_VALUE,
    x.FIN_CODE_TEXT,
    'Document Values by Booking Period for Company "'"
        ||x.co_name
        ||"""
```

```

||"/"
||x.booking_currency_code
||"""
as DASH_HEADER

FROM V_DASH_DOC_VAL_BP_BC x
where x.co_id = '$COMPANY$'
    and x.booking_currency_id = '$BOOKING_CURRENCY$'
    and x.PERIOD_DATE >= trunc(to_date('$FROMDATE$', 'YYYY-MM-DD"T"hh24:mi:ss'), 'MM')
    and x.PERIOD_DATE < trunc(to_date('$TODATE$', 'YYYY-MM-DD"T"hh24:mi:ss'), 'MM')

union all

select
    100 as sort_order,
    max(x.PERIOD_CHAR),
    x.PERIOD_DATE,
    sum(x.DOC_VALUE) as DOC_VALUE,
    'TOTAL' as FIN_CODE_TEXT,
    'Document Values by Booking Period for Company "'"
        ||max(x.co_name)
    ||"""
    ||"/"
    ||x.booking_currency_code
    ||"""
    as DASH_HEADER

FROM V_DASH_DOC_VAL_BP_BC x
where x.co_id = '$COMPANY$'
    and x.booking_currency_id = '$BOOKING_CURRENCY$'
    and x.PERIOD_DATE >= trunc(to_date('$FROMDATE$', 'YYYY-MM-DD"T"hh24:mi:ss'), 'MM')
    and x.PERIOD_DATE < trunc(to_date('$TODATE$', 'YYYY-MM-DD"T"hh24:mi:ss'), 'MM')

group by Period_date, x.co_id, booking_currency_id
order by period_date,sort_order

</sql>

</data>

```

Note that this SQL creates dynamically the DASH\_HEADER column. It also creates a TOTAL record giving the net value of the monetary values for each month.

## Inventory

The dashboard widgets for the EC Revenue Inventory business area are divided into two main categories:

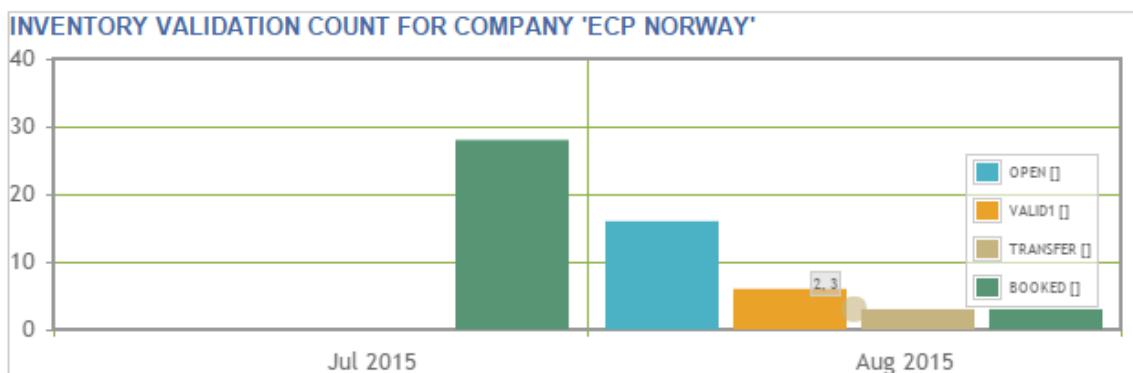
- Counting of processed inventories at the various Inventory Validation Levels:  
Open / Valid 1 / Valid 2 / Transfer / Booked
- Quantity and Monetary values overview



For all of these there are a number of variants further described in the following sections.

### **Inventory Validation Status Dashboards**

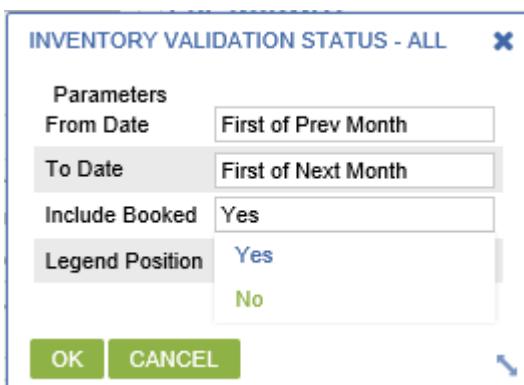
The purpose of these dashboards is to give an overview of the number of inventories at the various inventory validation levels. Typically it is important to make sure that all inventories for the current booking period are set to booked before closing the booking period:



These dashboards use INV\_VALUATION.DAYTIME when grouping them into a calendar month.

#### **Include Booked Inventories or not**

The dashboards have functionality to include or exclude Booked inventories in the dashboard:

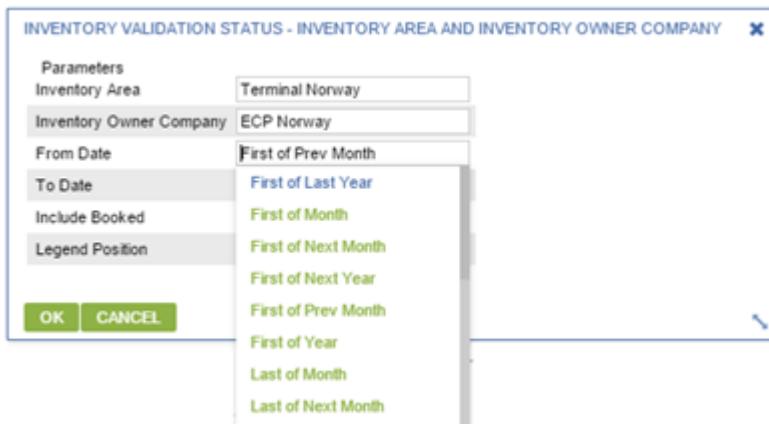


The reasons for this are:

- It is more important to get an overview of the number of inventories that are NOT Booked – it is these inventories you have to 'look after'
- If there is a big number of Booked inventories and only a few inventories at the other levels then the dashboard will not be able to show a bar for these due to the resolution of the Y-axis

#### **Time Scope for the dashboards**

The dashboards have 'From Date' and 'To Date' parameters defining the time scope for the data shown in the dashboard. These are using the pre-defined date methods given by EC Code 'MACRO\_TYPE':



By default the setting is:

- From Date = 'First of Prev Month'
- To Date = 'First of Next Month'

giving a 2-month window. For each individual dashboard you can change these settings to give a different time window.

### Inventory Validation Status Dashboards - Variants

These are the Inventory Validation Status Dashboards defined for the Inventory area

Dashboard Name	Dashboard View Name
Inventory Validation Status - All	V_DASH_INV_CNT_ALL
Inventory Validation Status - Inventory Owner Company	V_DASH_INV_CNT_CO
Inventory Validation Status - Inventory Area	V_DASH_INV_CNT_IA
Inventory Validation Status - Business Unit	V_DASH_INV_CNT_BU
Inventory Validation Status - Inventory Area and Inventory Owner Company	V_DASH_INV_CNT_CO_IA
Inventory Validation Status - Business Unit and Inventory Owner Company	V_DASH_INV_CNT_CO_BU

### Inventory Quantity and Monetary Value Dashboards

A number of dashboards showing various Inventory values have been created. All of these dashboards are using a common view as basis for the data: V\_DASH\_INV\_VALUES. This view extracts data from INV\_DIST\_VALUATION and groups it by Daytime/Inventory/Product/UOM.

On top of this view there is a table class – DASH\_INV\_VALUES. All the various dashboards are retrieving data through this class

To view the data run `select * from tv_dash_inv_values`

### Inventory Closing Quantity LineChart

This dashboard shows the **Inventory Closing Position** for inventories having a given UOM and Product:



The closing position is calculated as follows:

```
sum(nvl(x.closing_ul_position_qty1,0)) +
sum(nvl(x.closing_ol_position_qty1,0)) +
sum(nvl(x.closing_ps_position_qty1,0)) as INV_CLOSING_QTY,
...
FROM INV_DIST_VALUES x
```

There are two variants of this dashboard:

- Variant 1: *Inventory Closing Quantity by Company/Product/UOM*
- Variant 2: *Inventory Closing Quantity by Inventory Area/Company/Product/UOM*

Input parameters are:

- Company
- Product
- UOM
- From Date (using EC Code 'MACRO\_TYPE' – default value is 'First of Year')
- To Date (using EC Code 'MACRO\_TYPE' – default value is 'First of Next Year')
- Inventory Area (for Variant 2)

### Inventory Closing Value LineChart

This dashboard shows the **Inventory Closing Value** by Company for inventories having a given UOM and Product:



The closing position is calculated as follows:

```
sum(nvl(x.ul_group_value,0)) +
sum(nvl(x.ol_group_value,0)) +
sum(nvl(x.ps_group_value,0)) as INV_CLOSING_VALUE,
...
FROM INV_DIST_VALUES x
```

There are two variants of this dashboard:

- Variant 1: *Inventory Closing Value by Company/Product/UOM*
- Variant 2: *Inventory Closing Value by Inventory Area/Company/Product/UOM*

Input parameters are:

- Company
- Product
- UOM
- From Date (using EC Code 'MACRO\_TYPE' – default value is 'First of Year')
- To Date (using EC Code 'MACRO\_TYPE' – default value is 'First of Next Year')
- Inventory Area (for Variant 2)

### Inventory Values Overview (Table)

This dashboard shows

- **Inventory Opening Position Quantity**
- **Inventory YTD Movement Quantity**
- **Inventory Closing Position Quantity**
- **Inventory Closing Position Monetary Value in Group Currency**

by Company for inventories having a given UOM and Product.

There are two variants of this dashboard:

- Variant 1: *Inventory Values (Table) by Company/Product/UOM*
- Variant 2: *Inventory Values (Table) by Inventory Area/Company/Product/UOM*

Input parameters are:

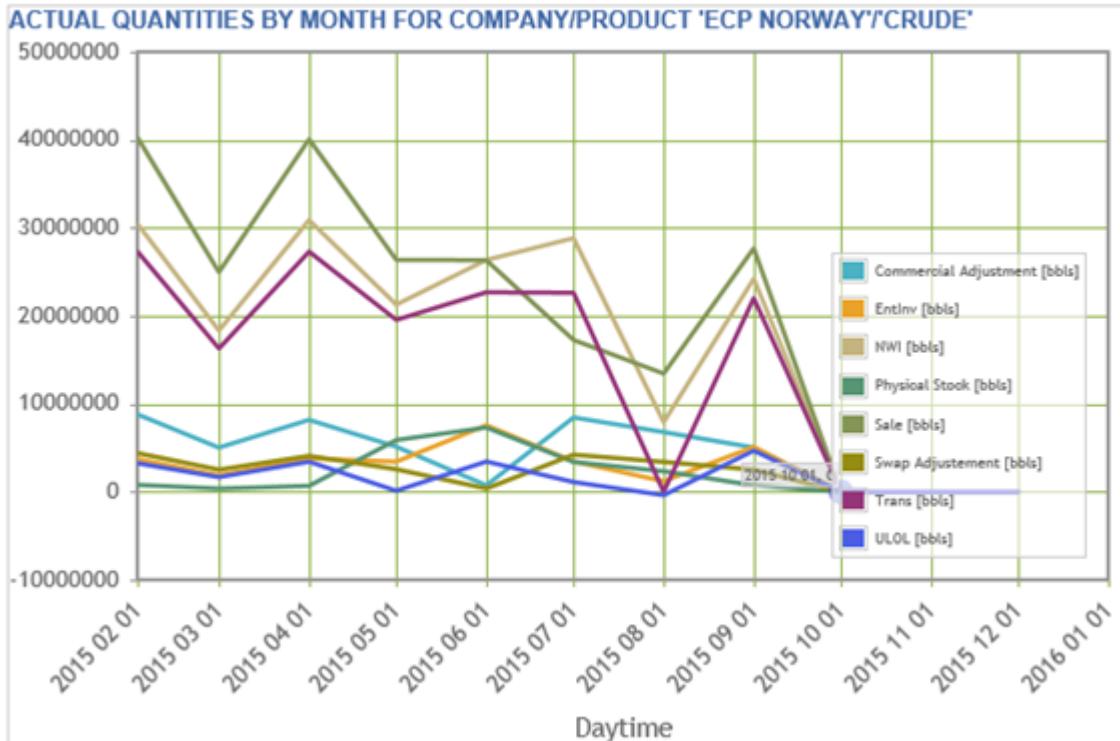
- Company
- Product
- UOM
- From Date (using EC Code 'MACRO\_TYPE' – default value is 'First of Year')
- To Date (using EC Code 'MACRO\_TYPE' – default value is 'First of Next Year')
- Inventory Area (for Variant 2)

INVENTORY VALUES BY COMPANY / PRODUCT / UOM							INVENTORY VALUES BY INVENTORY AREA / COMPANY / PRODUCT / UOM								
PERIOD	INVENTORY CODE	OPENING	MOVEMENT	CLOSING	UOM	CLOSING VALUE	CURRENCY	PERIOD	INVENTORY CODE	OPENING	MOVEMENT	CLOSING	UOM	CLOSING VALUE	CURRENCY
Jan 2009	IN_CASE_1A	-145,000	10,000	-135,000	Sm*	-5758495,2	USD	Jan 2009	IN_CASE_3A	2,150,000	30,000	2,180,000	Sm*	34320873,6	USD
Feb 2009	IN_CASE_1A	-145,000	-10,000	-155,000	Sm*	-6653881,7	USD	Feb 2009	IN_CASE_3A	2,150,000	-20,000	2,130,000	Sm*	33746018,2	USD
Dec 2006	IN_CASE_1A	-145,000	990,000	845,000	Sm*	12511219,4	USD	Dec 2006	IN_CASE_3A	2,150,000	-2,620,000	-770,000	Sm*	-31136120,4	USD
Jan 2010	IN_CASE_1A	845,000	-10,000	835,000	Sm*	12386749,05	USD	Jan 2010	IN_CASE_3A	-770,000	-30,000	-800,000	Sm*	-32153944	USD
Feb 2010	IN_CASE_1A	845,000	-4,410,000	-3,565,000	Sm*	-142777465,7	USD	Feb 2010	IN_CASE_3A	-770,000	650,000	180,000	Sm*	2711256,4	USD
Jan 2009	IN_CASE_1B	-5,000	10,000	5,000	Sm*	78717,8	USD	Jan 2009	IN_CASE_3B	2,750,000	30,000	2,780,000	Sm*	43766988,6	USD
Feb 2009	IN_CASE_1B	-5,000	-10,000	-15,000	Sm*	-43922,1	USD	Feb 2009	IN_CASE_3B	2,750,000	-20,000	2,730,000	Sm*	43254502,2	USD
Dec 2006	IN_CASE_1B	-5,000	90,000	85,000	Sm*	1268884,2	USD	Dec 2006	IN_CASE_3B	2,750,000	-2,620,000	-170,000	Sm*	-8874208,4	USD
Jan 2010	IN_CASE_1B	85,000	-10,000	75,000	Sm*	1112582,25	USD	Jan 2010	IN_CASE_3B	-170,000	-30,000	-200,000	Sm*	-8038486	USD
Feb 2010	IN_CASE_1B	85,000	-450,000	-365,000	Sm*	-14518169,7	USD	Feb 2010	IN_CASE_3B	-170,000	650,000	780,000	Sm*	11748777,72	USD
Jan 2009	IN_CASE_1C	175,000	10,000	185,000	Sm*	2912551,2	USD	Jan 2009	IN_CASE_3C	2,050,000	30,000	2,080,000	Sm*	32746521,6	USD
Feb 2009	IN_CASE_1C	175,000	-10,000	165,000	Sm*	2614283,1	USD	Feb 2009	IN_CASE_3C	2,050,000	-20,000	2,030,000	Sm*	32163604,2	USD
Dec 2006	IN_CASE_1C	175,000	-1,010,000	-935,000	Sm*	-33784494,2	USD	Dec 2006	IN_CASE_3C	2,050,000	-2,920,000	-870,000	Sm*	-35179772,4	USD
Jan 2010	IN_CASE_1C	-835,000	-10,000	-845,000	Sm*	-33962803,35	USD	Jan 2010	IN_CASE_3C	-870,000	-30,000	-900,000	Sm*	-36173187	USD
Feb 2010	IN_CASE_1C	-835,000	4,360,000	3,555,000	Sm*	52549227,6	USD	Feb 2010	IN_CASE_3C	-870,000	650,000	80,000	Sm*	12050022,84	USD
Jan 2009	IN_CASE_2A	15,000	30,000	45,000	Sm*	-502581,6	USD	Jan 2009	IN_CASE_4A	2,470,000	70,000	2,540,000	Sm*	44832700,8	USD
Feb 2009	IN_CASE_2A	15,000	30,000	45,000	Sm*	-776633,7	USD	Feb 2009	IN_CASE_4A	2,470,000	60,000	2,530,000	Sm*	45502474,2	USD

### Quantity

#### Quantities by Company/Stream Item Category/Product/BBL

This dashboard is a Line Chart for actual monthly quantity data with individual series for each Stream Item Category for a given Company and UOM showing the sum of the quantities added together over the various Fields:



A view has been created to provide the data: V\_DASH\_REVN\_QTY:

- The view extracts data from the STIM\_MTH\_ACTUAL table
- The view only include data for **Unique** Stream Items: REPORTING\_CATEGORY = 0
- The view excludes Stream Item values where the FIELD is 'SUM'
- It has 13 columns of quantity data corresponding to the data columns in the STIM\_MTH\_ACTUAL table with column names as follows:
  - BBLS15
  - BBLS
  - MSCF
  - MNM3
  - MSM3
  - MV
  - UST
  - USTV
  - MTV
  - MJ100
  - THERMS
  - KWH
  - BOE
- Columns for COMPANY\_ID / PRODUCT\_ID / STREAM\_ITEM\_CATEGORY\_ID
- Columns for Stream Item Category Code / Stream Item Category Name
- Dashboard header:  
'Actual Quantities by Month for Company/Product <company name> / <Product name>'
- The SUM function has NVL handling for NULL value Stream Items

On top of this view there is a table class: DASH\_REVN\_QTY which is then used for retrieving the data into the dashboard widget.

The dashboard widget retrieves the BBLS column, i.e. showing the data in BBLS.  
This is controlled by CTRL\_DASHBOARD\_PARAM.RowyColName = BBLS

#### Widget Name: Quantities by Company/Stream Item Category/Product/BBLS

Parameters:

- Company
- Product
- From Date

- To Date

QUANTITIES BY COMPANY/STREAM ITEM CATEGORY/PRODUCT/BBLS X

Parameters

Company	ECP Norway
Product	Crude
From Date	First of Year
To Date	First of Next Year

OK CANCEL

## EC Framework

This chapter contains the Technical Documentation for EC Framework

- PKI Technical Documentation
- Advanced File Import - Technical Reference
- EC FRMW Standard Reports Configuration Guide
- EC IS Configuration Guide
- EC IS Staging Table Extension
- EC View Generator and Object Class Model
- How to Setup Customer Defined Password Validation
- How to Setup Timezone
- How to Use Date Macro Parameter
- Row-Level Security
- How to Setup SSL/HTTPS
- Smart Journaling
- How to create, install and configure a Jasper Report
- Messaging Technical Documentation
- How to Encrypt Passwords in EC 11
- Integration Services Technical Documentation
- EC Web Services
- EC Dashboard Configuration Guide

## PKI Technical Documentation

### Signing Process

EC screens can be configured to use signing. This means that when doing insert/update/delete the user will be required to sign the action when pressing the save button. The signing requires a certificate from a PKI provider. These certificates are located in smart cards, which are pocket-sized cards with embedded integrated circuits. When inserted into a card-reader, the chip makes contact with electrical connectors that can read information from the chip. The certificate holds information about a specific user so when a signature is performed it is a binding confirmation that the changes was made by the user owning that certificate.

### Detailed Process Description

#### Signing of Data

When data in specific screens are changed by insert, update or delete, it will require the user to sign the data before it can be saved. This way the changed data will be stored with the signature of the user that changed it.

To sign data the user must first insert the PKI certificate, modify the data and after clicking on the save button the window viewed in Figure 1 appears. All data that is to be signed is wrapped in a text document in EC and will be transferred to this PKI signature page.

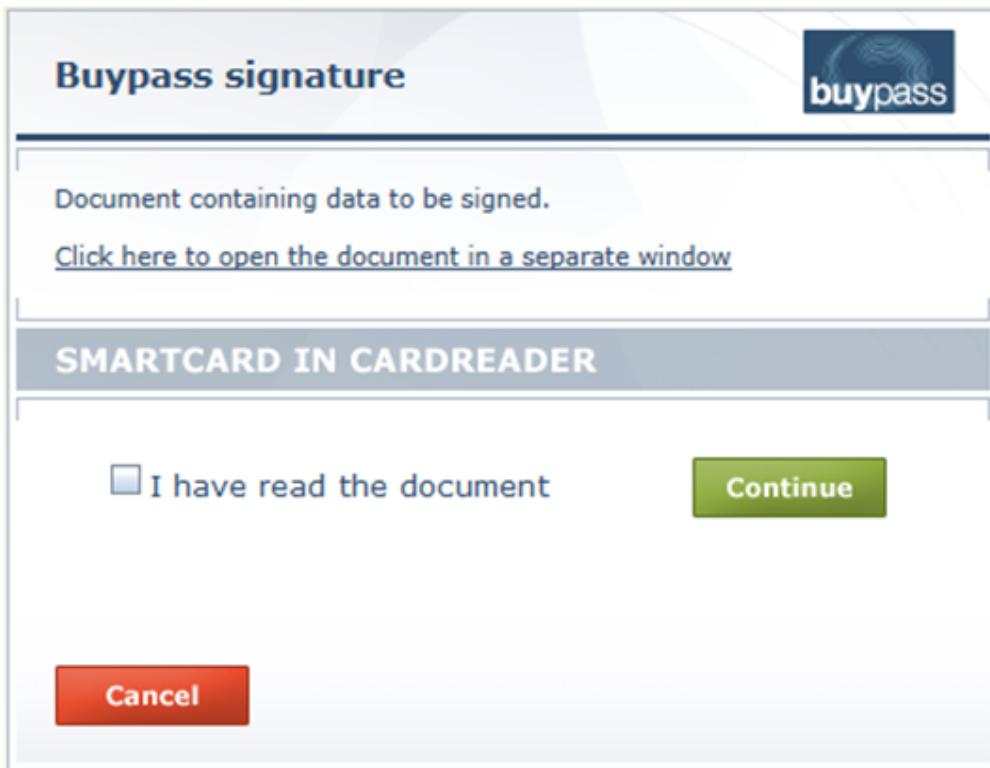


Figure 1 – Accepting the containing data to be signed

The PKI signature page contains a view that shows the user that the PKI certificate is recognized. Below this view, the user can see the text document transferred from EC containing all the data that will be signed and stored.

This information can also be viewed in a separate window by using the "**[Click here to open the document in a separate window](#)**".

To verify that the document is read, the user must check the checkbox and press the **Continue** button to accept the changes.

The screenshot shows a web-based application for digital signatures. At the top, the title "Buypass signature" is displayed next to the "buypass" logo. Below this, a message states "Document containing data to be signed." with a link "Click here to open the document in a separate window". A grey header bar labeled "SMARTCARD IN CARDREADER" spans across the middle. The main content area contains a form for entering a PIN: "Enter PIN (4 digits)" followed by a text input field, a "Confirm" button, and a "Forgot PIN?" link. A red "Cancel" button is located at the bottom left of the form area.

Figure 2 – Enter certificate PIN-code to sign all changes

The window in Figure 2 is the last step in the signing process. The user must enter the certificate PIN-code and press the **Confirm** button. If the PIN-code is accepted, the changed data will now be signed and saved.

This screenshot shows the same "Buypass signature" interface as Figure 2, but the PIN entry dialog is no longer present. Instead, the main content area displays the message "Signature in progress, please wait ...". A circular loading icon with two arrows is centered below the text. A red "Cancel" button is located at the bottom left.

Figure 3 – Wait while signature is in progress page

While processing the signing request, users will see the **Please wait...** message in the PKI page as shown in Figure 3. Please note that no further information will be given to the user while EC executes the save action for signed data after successful interaction with PKI.

### Process Outline

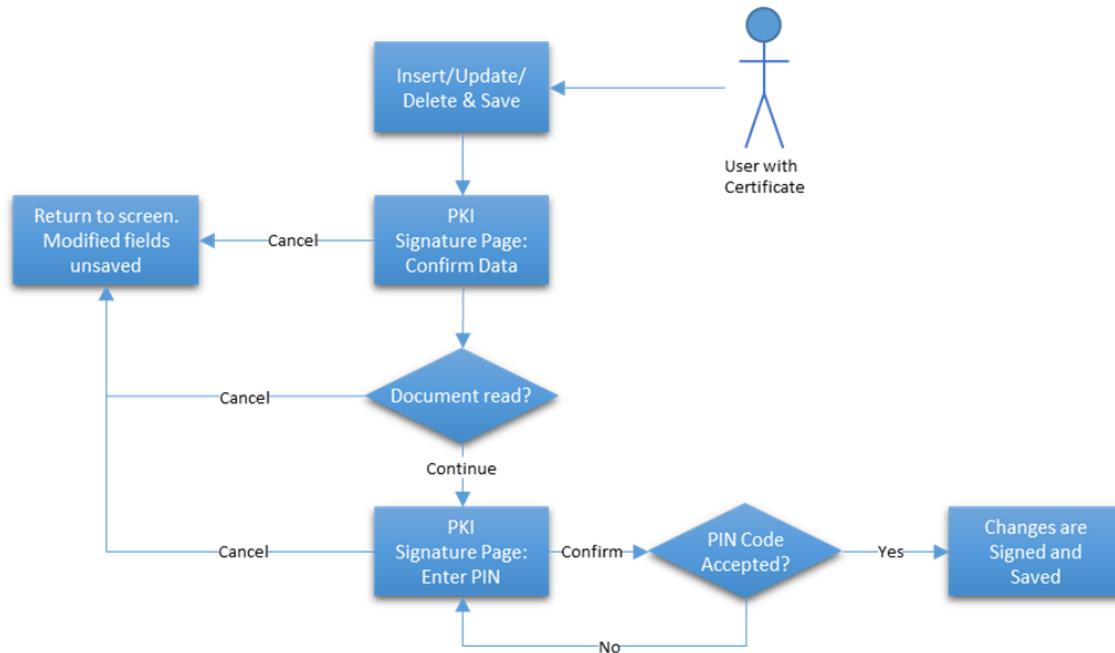


Diagram 2 – PKI Signing Process

Process	Description
Insert/Update/Delete & Save	To start the signing process the user must modify some data in a screen that requires a signature and press <b>Save</b> .
PKI Signature Page: Confirm Data	Page for viewing the modified data and confirming that the data to be saved is correct.
PKI Signature Page: Enter PIN	Page for signing the changes through PKI certificate and PIN code.
Return to screen. Modified fields unsaved	If the signing process is cancelled the windows will close and return the user to the screen he tried to save data in. The modified data are still modified and unsaved.
Changes are Signed and Saved.	All modified data are saved with this user's signature.

Table 2 – PKI Signing Process Description

### Disable PKI

Should there be a problem with the signing process, PKI can be disabled to allow the other processes to work almost as usual. This chapter describes how to enable and disable the possibility for saving data within EC without signing with a PKI certificate.

The enabling and disabling of PKI are done in the **Maintain System Settings** screen. There is a property that indicates that "PKI signing service unavailable". A value of 'true' will Force Save and disable PKI, while 'false' will enable PKI.

When PKI is enabled, all users (both internal and external) with access to the particular business function will have to sign data within the business function each time they try to save or modify the data. For the system to know if the user is internal or external, there will be an added column in **User Maintenance** screen indicating the state with a checkbox.

When PKI is disabled, all internal users will be able to save data within the business function without having to sign with a PKI certificate. However, external users will not be able to save data without signing since they are rejected at the login while PKI is disabled.

### Process Outline

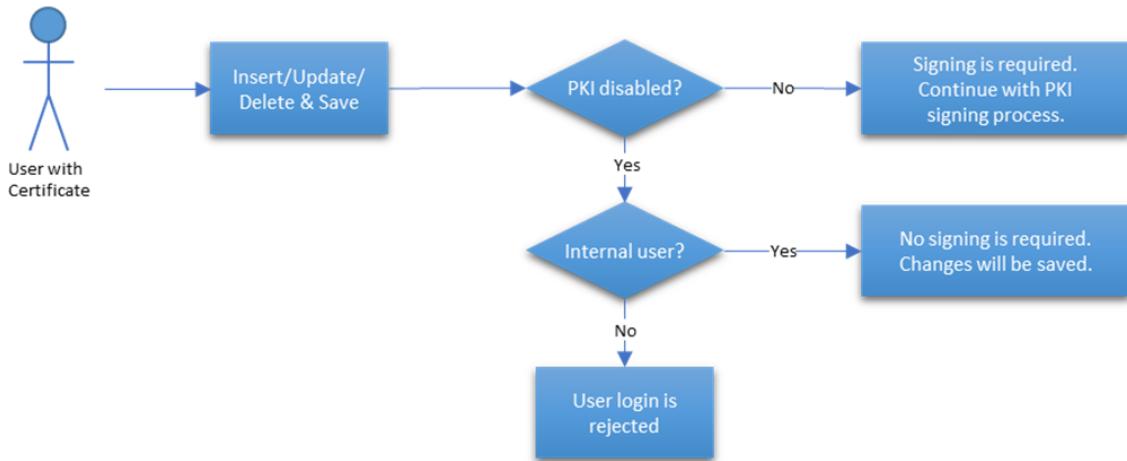


Diagram 3 – Disable PKI Process

Process	Description
Insert/Update/Delete & Save	To start the signing process the user must modify some data in a screen that requires a signature and press <b>Save</b> .
Signing is required. Continue with PKI signing process	If PKI is enabled then both internal and external users must sign before the data can be saved. The user must then do the usual PKI signing.
No signing is required. Changes will be saved.	If the PKI is disabled and the user that did the changes is an internal user, the changes will not be saved without signing.
User login is rejected	If PKI is disabled and the user is external, the user's login will be rejected.

Table 3 – Disable PKI Process Description

### PKI Verification

The PKI verification screen is for verifying that all data in EC is correct with the signed data. In this screen, users can search through all the documents that have been signed. The verification search is done by Period and Business Function.

The **Verify** button will take the selected row and check if the document matches the stored signature. The user will then get a message stating that the verification is a "Success" or "Failure". If it returns a verification failure, it means that the document has been altered and that the signature does not match. After a failure, the Valid column will be updated with N (for NOT VALID). If it is a success, the Verified Date will be updated with the current date.

**SEARCH**

**FAVORITES**

**MENU**

- ✓ **CONFIGURATION**
  - > ACCESS
  - > PREFERENCES
  - > CODES
  - ✓ **SYSTEM**
    - System Attributes
    - **PKI Signature Verification**
    - Status Processes
    - Class Finder
    - Class
    - Class Relations
    - Class Attributes
    - Class Attributes Presentation Config
    - XSD Downloads
    - Group Model
    - Navigation Model
    - Business Function Profiles
    - Initiate Day
    - Integrity Status
  - ✓ **UNITS**
    - Units

**PKI SIGNATURE VERIFICATION**

Business Function	Component ID	Valid	Document Created Date	Verified Date
Measurement Types	/uom_setup	Y	2014-12-11 17:06	2014-12-11 17:06

Document Created

```
=====
Screen: /com.ec.frmw.co.screens/measurement_types
Component: /uom_setup
Action type: modify

Parameters:
  (VIEW_UNIT): N
=====
Screen: /com.ec.frmw.co.screens/measurement_types
Component: /uom_setup
Action type: modify

Parameters:
  (VIEW_UNIT): Y
=====
User: sysadmin
Document was created: 2014-12-11
```

Figure 4 – PKI Verification Business Function in EC

## Advanced File Import - Technical Reference

### Introduction

This section is the technical reference of the Advanced File Import, describing the concepts behind the design, the configuration details and how to perform the execution of file import. The file term in this context means files of different types, like Excel, CSV, Fixed Width, XML etc.

This section has a top-down structure. The first section covers an overview of the Advanced File Import. This is followed by technical details about how to configure the extraction and loading and how this information is organized in the database.

### Overview

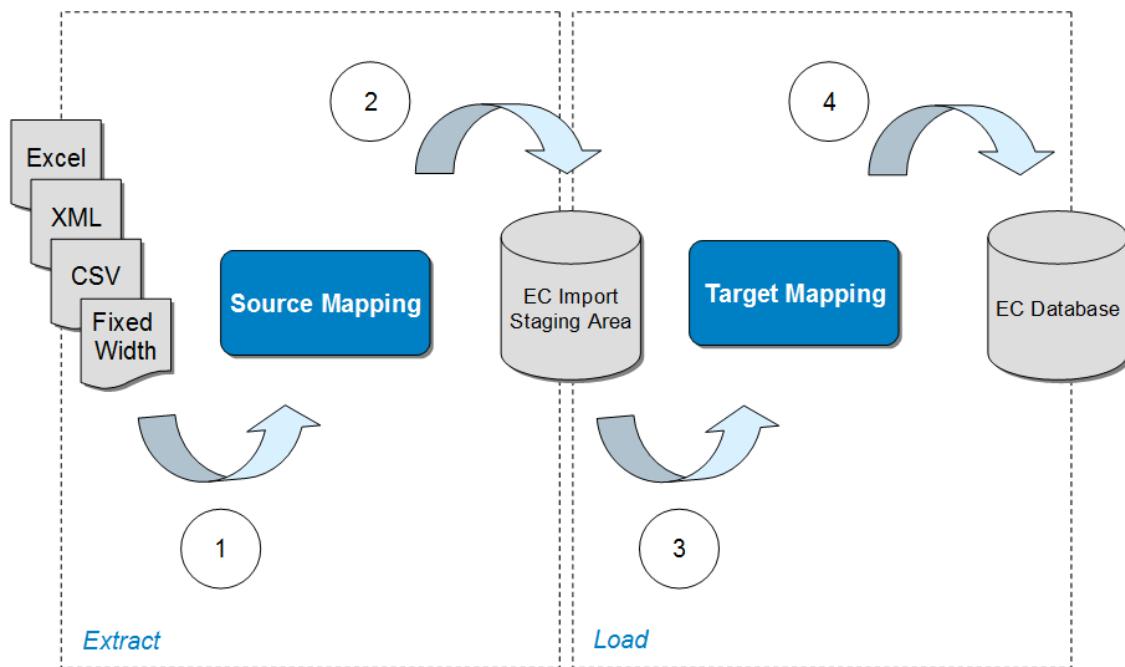
#### Summary

The advanced file import mechanism in EC is a framework which enables the users to easily setup and reconfigure import of advanced files structures. For a typical Excel format it supports multiple sheets, crosstab, tables and forms. In addition the import mechanism supports similar structures for TEXT (CSV and Fixed Width) and XML.

All configuration of the import is done through screens available in EC.

The import mechanism provides a file pickup from any drop folder accessible from the EC application server.

#### Functional Concept



The main concept for the Advanced File Import consists of four steps:

1. Read the source file based on a set of source mappings
2. Convert the read data to staging format and write it to the staging area
3. Read staging data and convert it to the EC class format based on a set of target mappings
4. Store the EC class data

The two first steps can be categorized as extraction and the two last steps as loading of data. In order to setup a such flow, it's necessary to configure the mappings as described below.

## Mapping Configuration

The mappings between a source file and the EC database is split into source mappings and target mappings. The first category represents the extraction and the last category the loading.

### Interface definition

The source mappings or extraction part belongs to an interface, so that it is possible to control the execution of each separate file or file set. The interface consists of a name and options related to transaction types and user exits.

SOURCE INTERFACE													
Code	Name	Functional Area	Type	Transaction Type	Source Type	Manual Staging	EC Validation Level	EC Storage Data Level	Overwrite	Save Files (in Days)	Sort Order	Revision Text Template	
AUDREY	Audrey	ECIS Interface Area	First Insert then Update	Row based transactions	Excel	<input checked="" type="checkbox"/>	Provisional	Provisional	Full	Sort by file name			
BIGIMPORT	BigImport	ECIS Interface Area	First Insert then Update	Row based transactions	Excel	<input checked="" type="checkbox"/>	Provisional	Provisional	Full	Sort by file name			

Field Name	Description
Code	The interface code
Name	The interface name
Functional Area	Ability to choose different EC Functional Area
Type	<p>Ability to choose different processing type on a writing approach (INSERT / UPDATE, INSERT or UPDATE)</p> <ul style="list-style-type: none"> <li>■ Insert: Will only perform insert operation</li> <li>■ Insert then update: Will first perform an insert. If this fails with unique constraint, it will try an update</li> <li>■ Update: Will only try an update operation</li> </ul>
Transaction Type	<p>Ability to choose whether rows imported should be committed row for row or the whole job as one transaction</p> <ul style="list-style-type: none"> <li>■ Row based: If one row fails, it will still try to insert the rest of the rows, and only the failed row will not be written.</li> <li>■ Job based: If one row fails, the whole transaction is rolled back.</li> </ul>
Source Type	Ability to choose different interface source type
Manual Staging Import	Ability to choose manual verification on imported staging data. If this box is ticked, someone will have to manually go into the staging area screen, look at the data and click the import data button before the data can be moved from staging to EC.
EC Validation Level	Tells which level the interface is allowed to overwrite.
EC Storage Data Level	Tells which record status the rows inserted or updated in EC will get when running the interface.
Overwrite	<p>This is for overwriting existing data in EC.</p> <ul style="list-style-type: none"> <li>■ Full: All data will be updated regardless if they are changed or not</li> <li>■ Incremental: Only data which has changed will be updated.</li> </ul>
Save Files (in Days)	Ability to set the time period buffer to keep the imported file(s) in EC
Sort Order	Choose which sort order to read files if there are several files to read.
Revision Text Template	If this is set to a value <value>, the import will update the revision text to <value> + timestamp + counter. The counter is to avoid problems if two values on the same row is updated within the same second.

### Source mapping definition

Each interface consists of one or several source mappings. Every source mapping extracts a value set from the source file. A value set can act as a single key value, a list of keys or a larger data set (the actual values). The source mapping is also given an EC Key reference so that it can be linked to a target mapping, defining how to handle data against EC.

Sort Order	Code	Name	Path Origin	Type	ValueType	Staging Group	ECKey	Key 1	Key 2	Key 3	Key 4	Key 5	Key 6	Key 7	Key 8	Key 9	Key 10	Navigation Method	Column Size
10	DATE	date	Sheet1.A1	KEY_VALUE	DATE														▼
20	WELL	well	Sheet1.A1	KEY_LIST	STRING														▼
30	VALUE	value	Sheet1.A1	DATA	NUMBER		ec1Value	ROWS WELL	DATE										▼
40	VALUE2	value2	Sheet1.A1	DATA	NUMBER		ec2	ROWS WELL	DATE										▼

Field Name	Description
Sort Order	Sort order of the mapping
Code	Source mapping code
Name	Source mapping name
Path Origin	<p>Set where the program should start reading the file. Example:</p> <ul style="list-style-type: none"> <li>▪ Excel: Sheet1.A1</li> <li>▪ CSV: 0.0</li> </ul>
Type	Choose what type of mapping <ul style="list-style-type: none"> <li>▪ KEY_VALUE - a single key value</li> <li>▪ KEY_LIST - a list of key values</li> <li>▪ DATA - data field</li> </ul>
Value Type	Which type of data <ul style="list-style-type: none"> <li>▪ Date</li> <li>▪ String</li> <li>▪ Number</li> </ul>
Staging Group	All data fields with the same staging group will be written in the same update operation, provided that they map to the same row in the database.
ECKey	Unique EC key
Key 1	Source mapping key value 1
Key 2	Source mapping key value 2
Key 3	Source mapping key value 3
Key 4	Source mapping key value 4
Key 5	Source mapping key value 5
Key 6	Source mapping key value 6
Key 7	Source mapping key value 7
Key 8	Source mapping key value 8
Key 9	Source mapping key value 9
Key 10	Source mapping key value 10
Navigation Method	Ability to choose navigation method if reading text file <ul style="list-style-type: none"> <li>▪ CSV - if the mapping reads a csv file</li> <li>▪ Fixed width - if the mapping reads a fixed width file</li> </ul>
Column Size	If using fixed width, this value specifies how long the field is.

#### Source mapping path definition

The atomic extraction commands for a specific source mapping, used to traverse the files, are called Source Paths. Each source mapping has one or several source paths.

## SOURCE MAPPING COMMANDS SOURCE MAPPING VALIDATION RULE

Sort Order	Type	Path	Path Param 1	Path Param 2	Path Param 3
10	UPPER_LEFT	Move(col, row)	2	2	

Field Name	Description
Sort Order	Sort order of the command
Type	Which type of command: <ul style="list-style-type: none"><li>■ UPPER_LEFT - specifies the upper left field of the set</li><li>■ LOWER_RIGHT - specifies the lower right field of the set</li></ul>
Path	What type of operation should be performed <ul style="list-style-type: none"><li>■ Move</li><li>■ FindVertical</li><li>■ FindHorizontal</li></ul>
Path Param 1	Parameter value 1
Path Param 2	Parameter value 2
Path Param 3	Parameter value 3

## SOURCE MAPPING COMMANDS SOURCE MAPPING VALIDATION RULE

Sort Order	Rule Name	Value String	Value Date	Value Number	Value Boolean	Exception Level
No records found.						

Field Name	Description
Sort Order	Sort order for the validation rule
Rule Name	Choose which type of rule to apply
Value String	Value to compare with if String
Value Date	Value to compare with if Date
Value Number	Value to compare with if Number
Value Boolean	Value if a Boolean validation
Exception Level	What output should the schedule give if this validation is breached. Error or warning?

**Target mapping definition**

The target mapping defines how the source data should be written to EC. It defined a link to the source mapping via an EC key, in addition to specifying classes and attributes. It is also possible to specify from unit and to unit in this component. If these are specified the incoming number be converted.



Target Mapping	Class	Attribute	Ec Key	Class Key 1	Class Key 2	Class Key 3	Class Key 4	Class Key 5	Class Key 6	Class Key 7	Class Key 8	Class Key 9	Condition 1	Condition 2	Condition 3	From Unit	To Unit	Constant String	Constant Number	Constant Date
PMIS_Dev_1/1/2015	Accts.Deb.Txn	id		Map 1	Map 2															
PMIS_Dev_1/1/2015	Accts.Deb.Txn	amount		Map 1	Map 2															

Field Name	Description
Class	Which class should the value be written to
Attribute	Which attribute in the class should the value be written to
Ec Key	Relates to the EC Key in the source mapping. Provides a link between the source mapping and the target mapping
Class Key 1	Class key 1 (Choose as many keys as the class has)
Class Key 2	Class key 2 (Choose as many keys as the class has)
Class Key 3	Class key 3 (Choose as many keys as the class has)
Class Key 4	Class key 4 (Choose as many keys as the class has)
Class Key 5	Class key 5 (Choose as many keys as the class has)
Class Key 6	Class key 6 (Choose as many keys as the class has)
Class Key 7	Class key 7 (Choose as many keys as the class has)
Class Key 8	Class key 8 (Choose as many keys as the class has)
Class Key 9	Class key 9 (Choose as many keys as the class has)
Class Key 10	Class key 10 (Choose as many keys as the class has)
Condition 1	Ability to define interface target mapping criteria condition
Condition 2	Ability to define interface target mapping criteria condition
Condition 3	Ability to define interface target mapping criteria condition
From Unit	If unit conversion is needed, choose from unit.
To Unit	If unit conversion is needed, choose to unit.
Constant String	Ability to define a constant string value
Constant number	Ability to define a constant number value
Constant Date	Ability to define a constant date value

### Source Mapping Screen

The screenshot shows the 'Mapping Configuration' interface. On the left, a navigation tree includes sections like 'CONFIGURATION', 'PROCESS AUTOMATION', 'MESSAGING', 'TASK LIST', 'EC INTEGRATION SERVICE', and 'IMPORT'. The main area has tabs for 'SOURCE INTERFACE', 'SOURCE MAPPING', and 'TARGET MAPPING'. Under 'SOURCE INTERFACE', there's a table with columns like Code, Name, Functional Area, Type, Transaction Type, Source Type, etc. Under 'SOURCE MAPPING', there's a table for 'Mapping Commands' and 'Validation Rule'. Under 'TARGET MAPPING', there's a table for 'Target Mappings'.

The source mapping screens allows the user to configure interfaces, source mappings, source paths and target mappings. The EC class information will be provided in the popups to simplify the configuration process.

Further details on how to use the screen please refer to the Advanced File Import – User Guide.

Details about the underlying data structures can be found in this document in the section **Database Details**.

### Target Mapping Screen

The screenshot shows the 'Target Mapping' screen. It features a navigation tree on the left with sections like 'EC PRODUCTION', 'EC SALES', 'EC REVENUE', etc. The main area displays a table of target mappings. The columns include Class, Attribute, Ec Key, and several Class Key columns (1 through 6). A search bar at the top allows filtering by Class, Attribute, or Ec Key.

There exists a separate screen for performing target mappings independently from the interfaces. This might be used when the interfaces share the same mappings. The navigator uses EC class, EC attribute or ECKey as navigator values.

Further details on how to use the screen please refer to the Advanced File Import – User Guide.

Details about the underlying data structures can be found in this document in the section **Database Details**.

### Staging Area Screen with Preview

The screenshot shows the 'DATA STATUS' section of the interface. It displays a table of incoming data with columns: Code, Staging Type, CODE, DATE, and TEMP. The data consists of five rows:

Code	Staging Type	CODE	DATE	TEMP
[TABLE]	TABLE	1 AS1_Well_001	2006-11-01 00:00	10.0
TEMP	TABLE	2 AS1_Well_002	2006-11-01 00:00	20.0
PRESS	TABLE	3 AS1_Well_003	2006-11-01 00:00	30.0
		4 AS1_Well_004	2006-11-01 00:00	
		5 AS1_Well_005	2006-11-01 00:00	50.0

Below the table, there are two status indicators: 'on is OK' and 'tion is OK'.

This screen is used to visualise the incoming data. All data shown here is based on the contents of the staging area. The navigator uses interface, filename and a date as parameter.

Further details on how to use the screen; please refer to the Advanced File Import – User Guide.

Details about the underlying data structures can be found in this document in the section **Database Details**.

### Import History Screen

The screenshot shows the 'IMPORT HISTORY' section of the interface. It displays a table of loaded data with columns: From Date, To Date, Functional Area, Interface, and Filename. The data consists of one row:

From Date:	To Date:	Functional Area:	Interface:	Filename:
		EC	EXCEL_IMPORT	\TA_EXAMPLE.xlsx_0002

Below the table, there is a status indicator: 'Historic Data Load'.

This screen is used to visualise the loaded data. From and to date can be specified. Interface and filename must be specified.

### Execution

Execution is performed by the EC Scheduler, running a business action which is common for all EC Integration Services.

**SCHEDULE**

Name	Description	Functional Area	Status	Pinned To	Next Fire Time	Enabled
AudreyExcelImport		EC	WAITING		2014-12-01 0	<input type="checkbox"/>
BOEConversionFactorSetup	Setup the inter-group conversion factor and queue quantity r	Revenue	EXPIRED			<input type="checkbox"/>
BOLCalculation	Load Calculation	Allocation	EXPIRED	WS002994.g		<input checked="" type="checkbox"/>
BatchImportERPMsg	Batch Import of ERP Messages	Revenue	EXPIRED			<input type="checkbox"/>
BigImport		EC	EXPIRED			<input type="checkbox"/>

(1 of 16) < < 1 2 3 4 5 6 7 8 9 10 > >> 5

**DETAILS BUSINESS ACTION SCHEDULE MONITOR**

**BUSINESS ACTION**

Action	Description	Sequence#	Isolate Transaction
ECISAction	AudreyExcelImport	10	<input type="checkbox"/>

**PARAMETERS**

Name	Value	Sequence#	Name	Name	Value
jobid	AudreyJobID			No records found.	No records found.

**ECIS JOB ACTIONS**

Job Action No	Job Action Class
10	com.ec.ecdm.is.advancedexcel.sourcemappping.jobaction.AdvancedExcelJobAction
20	com.ec.ecdm.is.advancedexcel.staging.jobaction.StagingJobActionTarget
30	com.ec.ecdm.is.advancedexcel.targetmapping.jobaction.TargetMappingJobAction

**ECIS PARAMETERS**

Parameter Name	Parameter Value
FTP_USER_NAME	
FTP_SERVER_NAME	
FTP_PORT	
DROP_FOLDER	
ERROR_FOLDER	
FTP_PASSWORD	
CONFIG_VALIDATION	Y
INTERFACE_CODE	AUDREY
FILE_FILTER	*
FTP_SOURCE_FILE	..

Each job is linked to a job configuration and a specific source interface. In addition a set of folder references must be applied.

## Monitoring

Monitoring is carried out by the EC Scheduler mechanisms. Each interface is connected to a schedule, and the schedule gets a log level. By inspecting the scheduler history screen, the user will get all details about the execution.

**SCHEDULE**

Name	Description	Functional Area	Status	Pinned To	Next Fire Time	Enabled
AudreyExcelImport		EC	WAITING	WS003040.U	2014-12-11 1	<input checked="" type="checkbox"/>
BOEConversionFactorSetup	Setup the inter-group conversion factor and queue quantity r	Revenue	EXPIRED			<input type="checkbox"/>
BOLCalculation	Load Calculation	Allocation	EXPIRED	WS002994.g		<input checked="" type="checkbox"/>
BatchImportERPMsg	Batch Import of ERP Messages	Revenue	EXPIRED			<input type="checkbox"/>
BigImport		EC	EXPIRED			<input type="checkbox"/>

(1 of 16) < < 1 2 3 4 5 6 7 8 9 10 > >> 5

**From Daytime** 2014-12-11 00:00 **To Daytime** 2014-12-12 00:00

**PARAMETERS**

Status	Start Time	Duration	Executed by	Name	Value
WARNING	2014-12-11 10:54:00	00:00:00	WS003040.Un	jobid	AudreyJobID
WARNING	2014-12-11 10:53:00	00:00:00	WS003040.Un		
MISFIRE	2014-12-11 10:52:12	00:00:00	WS003040.Un		
WARNING	2014-12-11 10:51:23	00:00:00	WS003040.Un		

**GET DETAILED LOG**

**BUSINESS ACTION HISTORY**

Name	Description	Start Time	Duration	Step	Status
ECISAction	AudreyExcelImport	2014-12-11 10:54	00:00:00	10	Warning

1 (WARN Data Data  
2 (WARN Data Data  
3 (OK): T. Data

## Data Extraction

In order to perform data extraction against a source file, specific source mappings need to be specified. Each mapping has a set of paths or commands describing how to navigate in the file.

### General Mapping Structure

As mentioned in the Overview section the general mapping structure consists of an interface, source mapping, source path and target mapping structure. This represents different ways to *navigate* in a source file.

In general for all source navigation models we distinguish between three different mapping types:

- *KEY\_VALUE* (extraction of one single value)
- *KEY\_LIST* (extraction of a one dimensional list of values, vertically or horizontally)
- *DATA* (extraction of a complete data set, consisting of one or more dimensions)

In order to find the values above, a set of paths or commands are provided:

- *Move(xDirection, yDirection)* (Move to a specific position in the file)
- *FindHorizontal(text | number | date)* (Move horizontally until a specific expression is found)
- *FindVertical(text | number | date)* (Move vertically until a specific expression is found)

Data types for the find commands are defined:

- *Text: <string | number | date>* can be blank. This can be specified by using the expression "" (ASCII-34 x 2)
- *Number: <number>* is specified by the format #.##, i.e. with a dot (ASCII-14) as the decimal separator if decimals are needed
- *Date: <date>* is specified by ISO format (yyyy-MM-ddTHH:mm:ss)

To hold a range of values, each path / command must be of the following two types:

- *UPPER\_LEFT*
- *LOWER\_RIGHT*

There might be minor differences between how the mappings are used for the different file types, especially when it comes to the navigational principles (commands). This will be shown below.

## Excel Mapping

Extraction of Excel data is based on standard EC libraries. Hence the Excel version compatibility is relying on those. Please check the EC documentation for further details.

### Path Origin

The path origin for an Excel source file is mandatory and is following the *<Sheet>.<Cell>* format. The *Sheet* refers to the naming of a specific sheet in the Excel workbook. The *Cell* refers to a valid cell in the selected sheet. If the sheet changes name rapidly, use *<sheetnumber>.<Cell>* instead of sheetname, but here you need to have *<>* to mark that this is a sheet number. See example.

*Example:*

*Well Test Values.B4*  
*<0>.A1*

### Available Commands

- *Move(<column number>, <row number>)*
- *FindVertical(<search string>, [CELL\_BEFORE])*
- *FindHorizontal(<search string>, [CELL\_BEFORE])*

The move command takes two parameters, one for the column number and one for the row. The column is given as a number of cells from the current position. The rows are given as the number of rows from the current position.

*Example 1)*

1. *Initial position: B4*
2. *Move(3,4)*
3. *Final position: E8*

The FindVertical command takes one mandatory parameter, the search string, and one optional parameter indicating if the cell before the found cell should be returned.

*Example 2)*

1. Initial position: B2
2. FindVertical("", CELL\_BEFORE)
3. Final position: B7

	A	B	C	D
1		Well	Date	WHT
2		W1	06.05.2008	45.2
3		W2	06.05.2008	46.9
4		W3	06.05.2008	44.2
5		W4	06.05.2008	42.7
6		W1	07.05.2008	51.2
7		W2	07.05.2008	53.3
8				

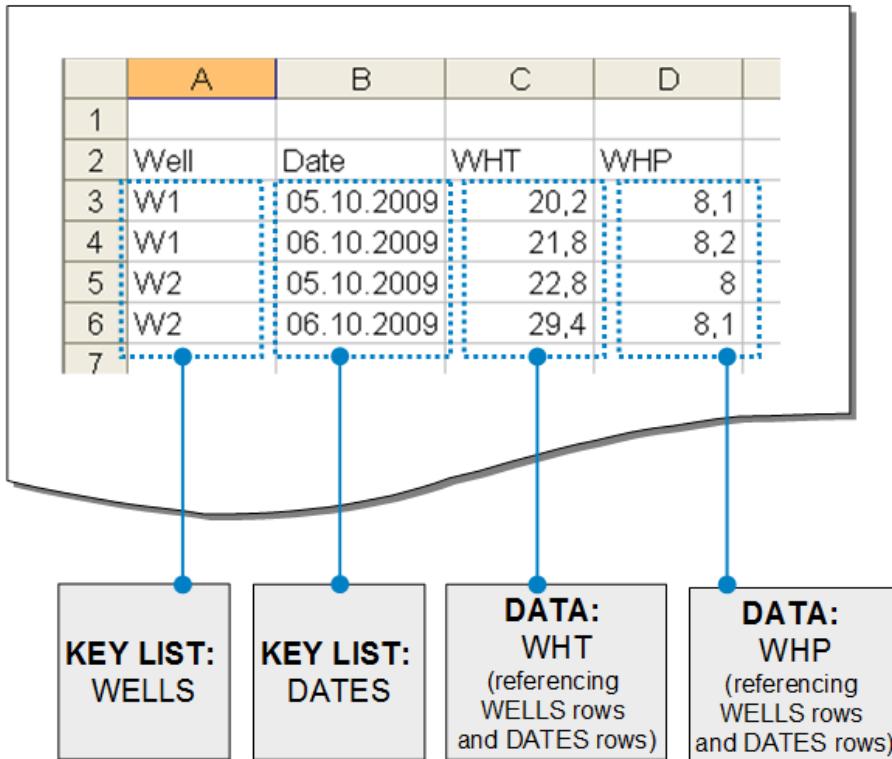
The FindVertical command takes one mandatory parameter, the search string, and one optional parameter indicating if the cell before the found cell should be returned.

*Example 3)*

1. Initial position: C2
2. FindHorizontal("", CELL\_BEFORE)
3. Final position: F2

	A	B	C	D	E	F	
1							
2		Well	02.03.2008	03.03.2008	04.03.2008	05.03.2008	
3		W1		23	24	22	21
4		W2		24	24	24	22
5		W3		26	28	28	24
6		W4		28	28	27	28
7							

**A Table Extraction Example**

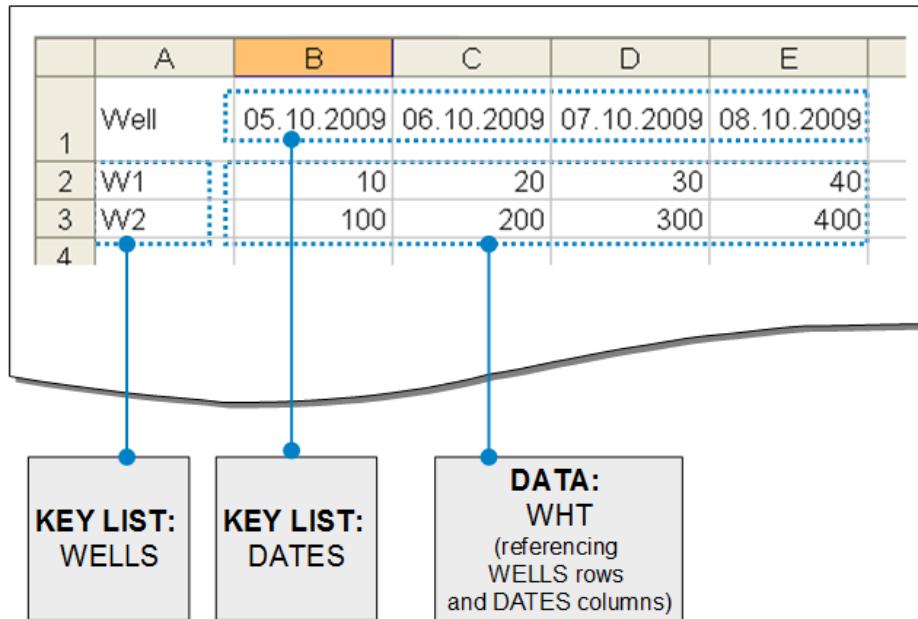


Code	Key 1	Key 2	Value	Staging Type	EC Key
WHT	W1	05.10.2009	20,2	TABLE	Wht
WHP	W1	05.10.2009	8,1	TABLE	Whp
WHT	W1	06.10.2009	21,8	TABLE	Wht
WHP	W1	06.10.2009	8,2	TABLE	Whp
WHT	W2	05.10.2009	22,8	TABLE	Wht
WHP	W2	05.10.2009	8	TABLE	Whp
WHT	W2	06.10.2009	29,4	TABLE	Wht
WHP	W2	06.10.2009	8,1	TABLE	Whp

A typical Excel extraction example consists of two key lists and one or several data sets. The first key list in the example extracts the well codes from the file. The next key list extracts the dates. The two data sets retrieve plain values. The data sets links to the rows from the two key lists. Hence writing the data to the staging area will result in the table as visualized in the drawing above.

When dealing with one dimensional data sets the resulting staging area will normally be of staging type TABLE. If the data set is spanning across multiple dimensions (several columns and rows), a CROSSTABLE will be the result. This is shown in the example below.

#### **A Crosstable Extraction Example**



Code	Key 1	Key 2	Value	Staging Type	Row	Column	EC Key
WHT	W1	05.10.2009	10	CROSSTABLE	1	1	Wht
WHT	W1	06.10.2009	20	CROSSTABLE	1	2	Wht
WHT	W1	07.10.2009	30	CROSSTABLE	1	3	Wht
WHT	W1	08.10.2009	40	CROSSTABLE	1	4	Wht
WHT	W2	05.10.2009	100	CROSSTABLE	2	1	Wht
WHT	W2	06.10.2009	200	CROSSTABLE	2	2	Wht
WHT	W2	07.10.2009	300	CROSSTABLE	2	3	Wht
WHT	W2	08.10.2009	400	CROSSTABLE	2	4	Wht

As shown in the example a multi-dimensional data set is selected. This is indexed by two key lists, one vertical (ROW based) and one horizontal (COLUMN based). The resulting staging area will produce the numbers with the matching keys and in addition refer the corresponding row and column.

### CSV Mapping

The CSV format is based on a column separation specified by a single separator character. This character is by default semi colon (;). This can be customized when setting up the job.

### Path Origin

The path origin for a CSV source file is mandatory and is following the <column>.<line> format. Both the column and line references are indexed from 0.

*Example:*

*Path origin: 0.4*

*Final position: column 1, line 5*

### Available Commands

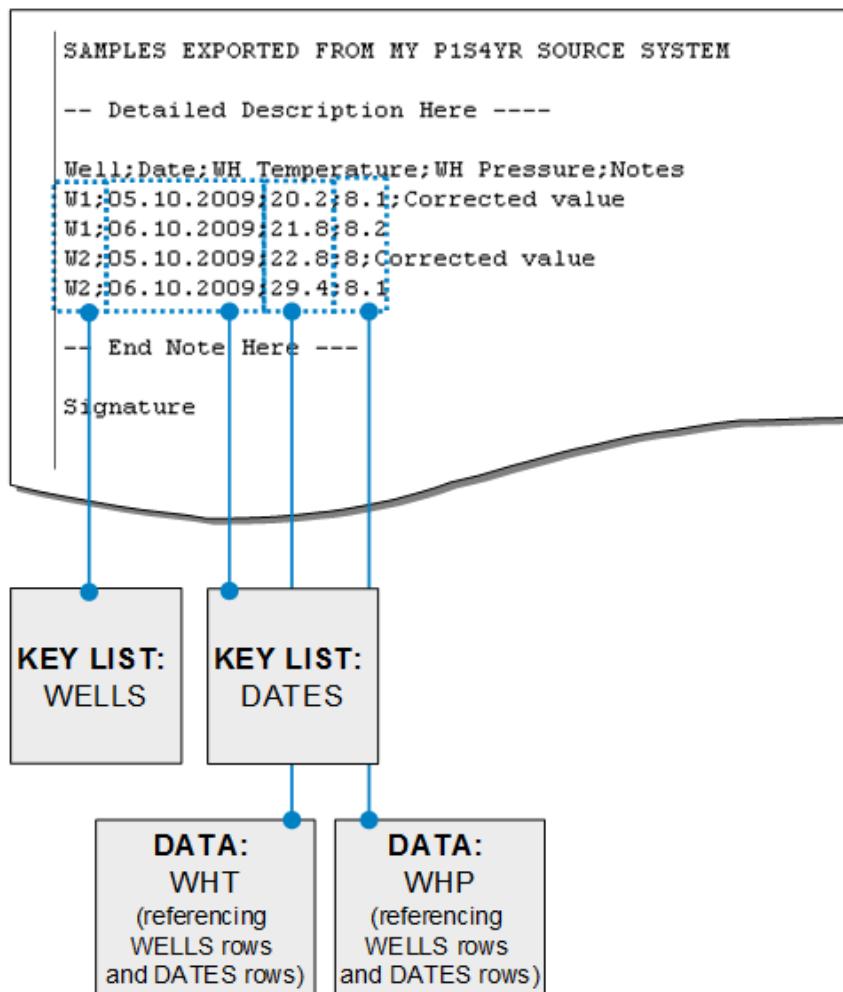
- Move(<column number>, <row number>)
- FindVertical(<search string>, [CELL\_BEFORE])
- FindHorizontal(<search string>, [CELL\_BEFORE])

The move command takes two parameters, one for the column number and one for the row. The column is given as a number of columns from the current position. The rows are given as the number of rows from the current position.

**Example 1)**

1. Initial position: 0.0
2. Move(1,4)
3. Final position: 1.4 (column 2, line 5)

The FindVertical and FindHorizontal command is used in the same way as for Excel. Vertical or Horizontal searches is limited to the current column or line in the file.

**A Table Extraction Example**

Code	Key 1	Key 2	Value	Staging Type	EC Key
WHT	W1	05.10.2009	20,2	TABLE	Wht
WHP	W1	05.10.2009	8,1	TABLE	Whp
WHT	W1	06.10.2009	21,8	TABLE	Wht
WHP	W1	06.10.2009	8,2	TABLE	Whp
WHT	W2	05.10.2009	22,8	TABLE	Wht
WHP	W2	05.10.2009	8	TABLE	Whp
WHT	W2	06.10.2009	29,4	TABLE	Wht
WHP	W2	06.10.2009	8,1	TABLE	Whp

The example above shows how key lists and data can be extracted from a CSV file, not very different from the Excel

approach. It is important that the path origin is set correctly when a file consists of different formatted sections.

## Text Mapping

Text Mapping is treated as *fixed width*, meaning that each column in the file has the same number of characters for each line.

### **Path Origin**

The path origin for a Text source file is mandatory and is following the <column>,<line> format. Both the column and line references are indexed from 0.

*Example:*

*Path origin: 0.4*

*Final position: column 1, line 5*

## Available Commands

- Move(<column number>, <row number>, <column size>)
- FindVertical(<search string>, <column size>)
- FindHorizontal(<search string>, <column size>)

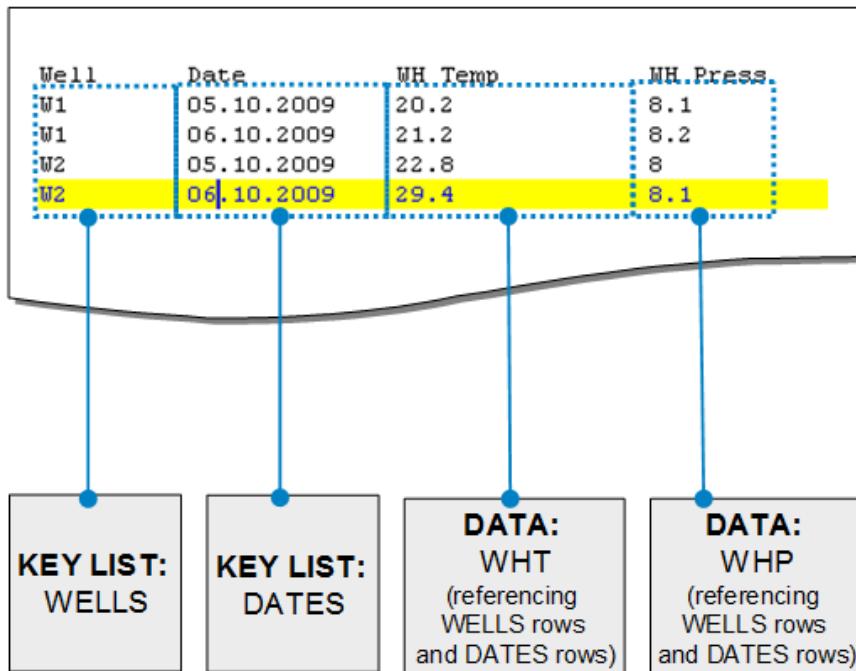
The move command takes three parameters, one for the column number one for the row and a final parameter setting the current column size. The column is given as a number of cells from the current position.

*Example 1)*

1. *Initial position: 0.0*
2. *Move(1,4)*
3. *Final position: 1-11.4 ((column 2 – column 12), line 5)*

The FindVertical and FindHorizontal have the same behaviour as for Excel, except for that it's using an additional parameter, column size / width, to specify each value's range.

## A Table Extraction Example



Code	Key 1	Key 2	Value	Staging Type	EC Key
WHT	W1	05.10.2009	20,2	TABLE	Wht
WHP	W1	05.10.2009	8,1	TABLE	Whp
WHT	W1	06.10.2009	21,8	TABLE	Wht
WHP	W1	06.10.2009	8,2	TABLE	Whp
WHT	W2	05.10.2009	22,8	TABLE	Wht
WHP	W2	05.10.2009	8	TABLE	Whp
WHT	W2	06.10.2009	29,4	TABLE	Wht
WHP	W2	06.10.2009	8,1	TABLE	Whp

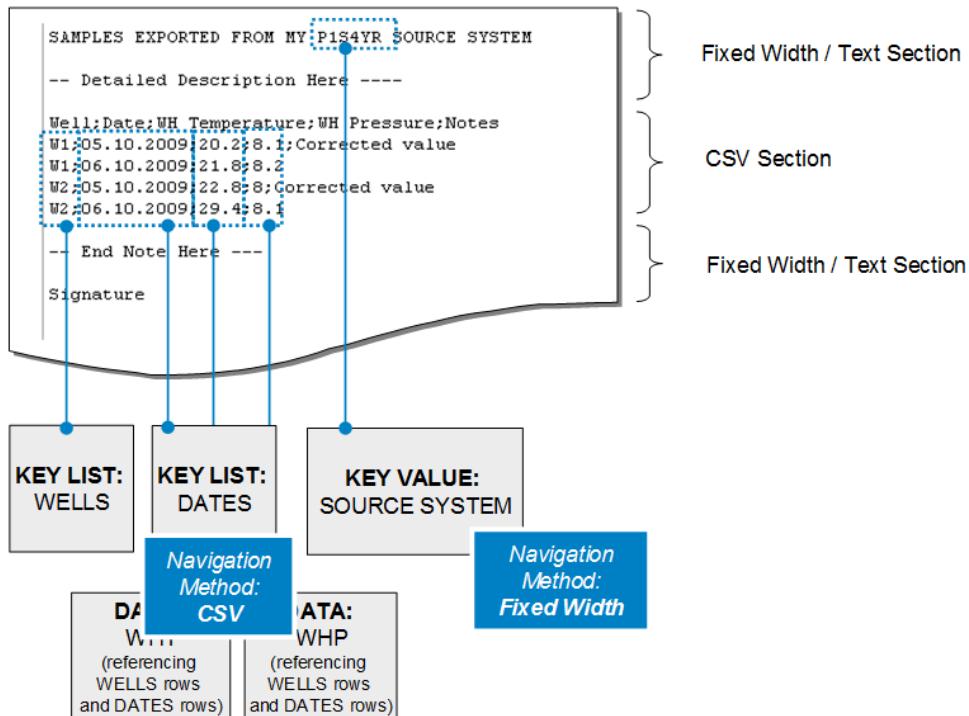
The example shows how each column in the text file has a specific column size / width and that this can be used for consistent navigation between lines. If spanning across a data set with multiple columns, each column is treated with common column size specified by the source command.

### Combined Formats Mapping

In some text cases there's no consistent format throughout the file. It might be a combination of Text and CSV. The interface for this case will typically be specified as text. In addition each source mapping may get the specific navigation method:

1. FIXED WIDTH
2. CSV

Ensure that the source path origin is set correctly for each of the mappings. E.g. The CSV section in the middle of a combined format should be extracted set the path origin to the start of the section and specify the source mapping with navigation method CSV.



### XML Mapping

The XML mapping is basically based on the same concept as for the plain column – line formats. A line in a plain format can correspond to a repeating XML section. A column will correspond to an attribute or tag value in each of the repeating XML sections.

### Path Origin

The source origin for an XML is specified by a single tag. The origin will jump to the first instance of the specified tag. If the outer wrapping for an XML is `<Body></Body>` and the further mappings will be based on the body content, Body is set as path origin.

### Available Commands

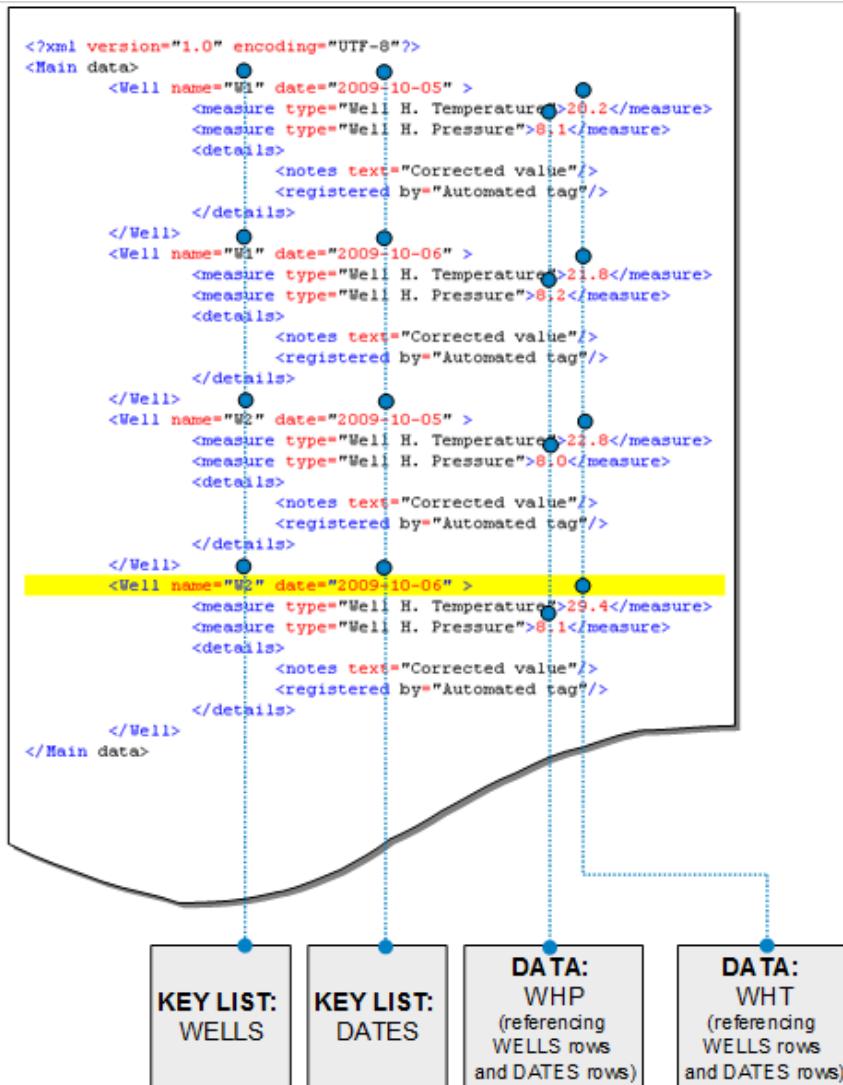
The commands for navigating in an XML file are a little bit different from the other formats. The source mapping is similar, but the paths are specified as an XPath expression. Referring to the user guide, the screen setup is helping the user to select the right tags and attributes, but the finally persisted expression is Xpath.

*Example:*

PATH	TEXT_1
/kFactorCoeffs/vessel/component/ConstantA_Value	
/kFactorCoeffs/account_period/account_month	
/kFactorCoeffs[@type]	
/kFactorCoeffs/vessel[@color]	
/Body/Stream[@Name]	
/Body/Stream/Production[@Type]	Volume
/Body/Stream/Production[@Quantity]	
/Body/Stream[@Gasday]	
/Body/Stream/Production[@Type]	Volume
/Body/Stream/Production[@Quantity]	
/Body/Stream/Properties/Analysis[@Quantity]	
/Body/Stream/Properties/Analysis[@Type]	Composition
/Body/Stream/Gasday	
/Body/Company[@name]	
/Body/Section	

Each of the paths is extracting a node, tag or attribute enabling the navigator to read value(s). The TEXT\_1 is representing a condition to the value extracted.

#### A Table Extraction Example



Code	Key 1	Key 2	Value	Staging Type	EC Key
WHT	W1	05.10.2009	20,2	TABLE	Wht
WHP	W1	05.10.2009	8,1	TABLE	Whp
WHT	W1	06.10.2009	21,8	TABLE	Wht
WHP	W1	06.10.2009	8,2	TABLE	Whp
WHT	W2	05.10.2009	22,8	TABLE	Wht
WHP	W2	05.10.2009	8	TABLE	Whp
WHT	W2	06.10.2009	29,4	TABLE	Wht
WHP	W2	06.10.2009	8,1	TABLE	Whp

## Data Loading

All extracted source data will be loaded into the staging area. The remaining step before it can be further loaded into the EC database is a mapping between staging data and EC classes. This is achieved by applying target mappings. Please refer the Advanced File Import – User Guide in order to see the details on how to setup those mappings.

### Staging area

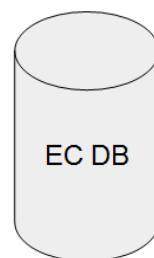
Code	Key 1	Key 2	Value	Staging Type	EC Key
WHT	W1	05.10.2009	20,2	TABLE	Wht
WHP	W1	05.10.2009	8,1	TABLE	Whp
WHT	W1	06.10.2009	21,8	TABLE	Wht
WHP	W1	06.10.2009	8,2	TABLE	Whp
WHT	W2	05.10.2009	22,8	TABLE	Wht
WHP	W2	05.10.2009	8	TABLE	Whp
WHT	W2	06.10.2009	29,4	TABLE	Wht
WHP	W2	06.10.2009	8,1	TABLE	Whp



### Target mapping

```

EC Key: Wht
Class Mapping: PWEL_DAY_STATUS
Attribute Mapping: WH_TEMP
Implicit mappings: KEY_1: OBJECT_CODE
Implicit mappings: KEY_2: DAYTIME
  
```



### Standard Mapping

Each staging row will be identified by an EC key. The target mapping is linking this key to a specific class and attribute. Implicit mapping like the class keys are being performed by internal lookup against the EC database. The class keys are linked in the specified order. If KEY\_1 in the target mapping will be linked against the first class key in the EC class lookup, etc.

### Use of Constants

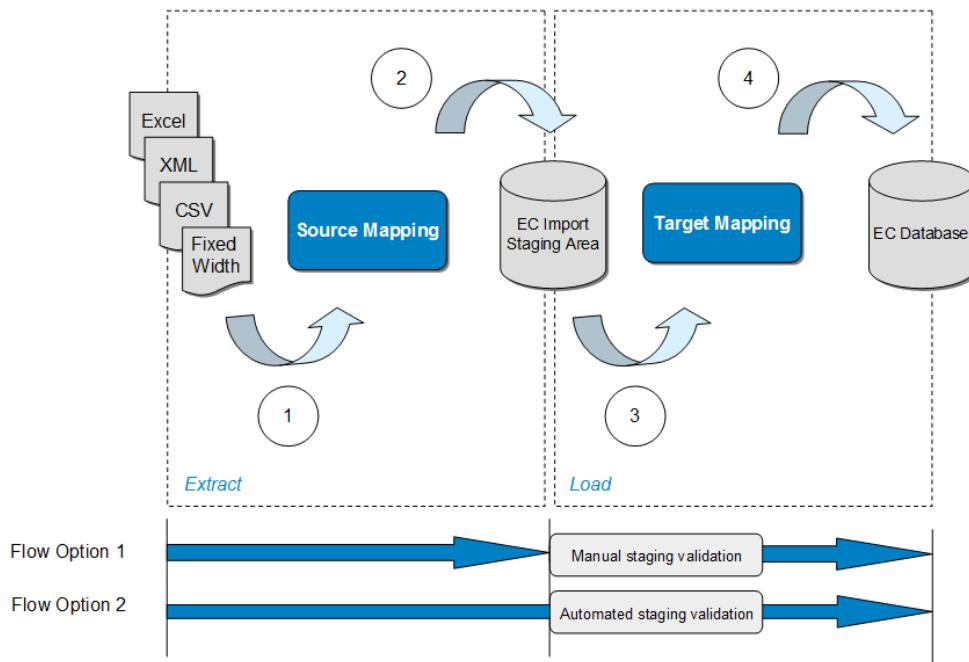
For many classes there might be mandatory attributes which are not available from the source file. This might typically be fixed values, object types etc. This can be resolved by making a duplicate target mapping to a specific key and set the constant value. The constant value is of one of the following types:

- STRING
- DATE
- NUMERIC

### Execution

#### Configuring the Flow

The flow can be configured according to two different approaches, one for manual staging validation and one for automated staging validation.



The first option requires that the flow is split into two different jobs, one for extracting data and loading them into staging and one for loading them into EC. The second flow option can be run in one single job, extracting the data from the source file and loading it into EC thru staging.

Each step in the flow is using EC IS Job Actions. The available Job Actions are:

1. com.ec.ecdm.is.advancedexcel.sourcemap.jobaction.AdvancedExcelJobAction (reading from a source file and producing staging format)
2. com.ec.ecdm.is.advancedexcel.staging.jobaction.StagingJobActionTarget (reading staging format and writing to staging area in database)
3. com.ec.ecdm.is.advancedexcel.staging.jobaction.StagingJobActionSource (reading data from staging area in database one file at the time, which means that the importing of data from staging area to the EC data tables will be done sequentially for each imported file)
  - com.ec.ecdm.is.advancedexcel.staging.jobaction.StagingJobActionSourceChunked Splits and imports the list of staging rows in chunks to avoid excessive memory usage when there is a large number of rows to import.
  - com.ec.ecdm.is.advancedexcel.staging.jobaction.StagingJobActionSourceChunced (deprecated - same as StagingJobActionSource)
4. com.ec.ecdm.is.advancedexcel.targetmapping.jobaction.TargetMappingJobAction

According to flow options in the drawing above the following combinations represents the corresponding setup:

- Flow 1) Job Action 1 -> 2 and Job Action 3 -> 4
- Flow 2) Job Action 1 -> 2 -> 4

Since the Advanced File Import is built on the EC Integration Services, configuration will follow the same scheme as the other EC IS components. It means that the job can either be setup thru an ecisconfig.xml or by applying parameters directly to the business action (the business action screen).

To be able to configure the EC IS through the database must the "**ecis\_db\_config**" business action parameter be set to "Y".

The parameters fulfilling an Advanced File Import configuration is shown below.

NAME	PARAMETER_VALUE	PARAM_TYPE	SUB_TYPE
Jobid	EXCEL_TO_TARGET_REPSOL_AFRODITA	BASIC_TYPE	STRING
ecis_db_config	Y	BASIC_TYPE	STRING
action_1_id	Excel Action	BASIC_TYPE	STRING
action_1_class	com.ec.ecdm.is.advancedexcel.sourcemappling.AdvancedExcelJobAction	BASIC_TYPE	STRING
action_1_FILE_FILTER	NOP2	BASIC_TYPE	STRING
action_1_DROP_FOLDER	C:\projects\Repsol\EC\Import\Afrodita	BASIC_TYPE	STRING
action_1_COMPLETED_FOLDER	C:\projects\Repsol\EC\Import\Afrodita\completed	BASIC_TYPE	STRING
action_1_ERROR_FOLDER	C:\projects\Repsol\EC\Import\Afrodita\errors	BASIC_TYPE	STRING
action_1_INTERFACE_CODE	REPSOL_AFRODITA	BASIC_TYPE	STRING
action_2_id	Staging Action	BASIC_TYPE	STRING
action_2_class	com.ec.ecdm.is.advancedexcel.staging.StagingJobActionTarget	BASIC_TYPE	STRING
action_3_id	Target Action	BASIC_TYPE	STRING
action_3_class	com.ec.ecdm.is.advancedexcel.targetmapping.TargetMappingJobAction	BASIC_TYPE	STRING

## Scheduling the Execution

The screenshot shows the EC Scheduler interface. On the left, there is a navigation menu with sections like Configuration, EC Production, EC Sales, EC Revenue, Reporting, Process Automation, Messaging, Task List, and EC Integration Service (which is expanded to show Schedules, Import, and other sub-options). The main area is titled 'SCHEDULE' and displays a table of scheduled jobs. One job, 'AudreyExcelImport', is highlighted. Below the table, tabs for 'DETAILS', 'BUSINESS ACTION', 'SCHEDULE', and 'MONITOR' are visible. Under 'BUSINESS ACTION', there is a table for 'ECISAction' with a sequence number of 10. It also includes sections for 'PARAMETERS', 'MACRO', and 'FIXED PARAMETERS'. The 'PARAMETERS' section contains a single entry 'jobid' with value 'AudreyJobID'. The 'ECIS JOB ACTIONS' section lists three actions: 10 com.ec.ecdm.is.advancedexcel.sourcemappling.jobaction.AdvancedExcelJobAction, 20 com.ec.ecdm.is.advancedexcel.staging.jobaction.StagingJobActionTarget, and 30 com.ec.ecdm.is.advancedexcel.targetmapping.jobaction.TargetMappingJobAction. The 'ECIS PARAMETERS' section lists various parameters with their values.

Name	Description	Functional Area	Status	Pinned To	Next Fire Time	Enabled
AudreyExcelImport	Setup the inter-group conversion factor and queue quantity	EC	WAITING		2014-12-01 0	<input type="checkbox"/>
BOEConversionFactorSetup	Setup the inter-group conversion factor and queue quantity	Revenue	EXPIRED			<input type="checkbox"/>
BOLCalculation	Load Calculation	Allocation	EXPIRED	WS002994.g		<input checked="" type="checkbox"/>
BatchImportERPMsg	Batch Import of ERP Messages	Revenue	EXPIRED			<input type="checkbox"/>
Bglimport		EC	EXPIRED			<input type="checkbox"/>

Action	Description	Sequence#	Isolate Transaction
ECISAction	AudreyExcelImport	10	<input type="checkbox"/>

PARAMETERS		MACRO		FIXED PARAMETERS	
Name	Value	Sequence#	Name	Name	Value
jobid	AudreyJobID		No records found.		No records found.

ECIS JOB ACTIONS	
Job Action No	Job Action Class
10	com.ec.ecdm.is.advancedexcel.sourcemappling.jobaction.AdvancedExcelJobAction
20	com.ec.ecdm.is.advancedexcel.staging.jobaction.StagingJobActionTarget
30	com.ec.ecdm.is.advancedexcel.targetmapping.jobaction.TargetMappingJobAction

Parameter Name	Parameter Value
FTP_USER_NAME	
FTP_SERVER_NAME	
FTP_PORT	
DROP_FOLDER	
ERROR_FOLDER	
FTP_PASSWORD	
CONFIG_VALIDATION	Y
INTERFACE_CODE	AUDREY
FILE_FILTER	*
FTP_SOURCE_FILE	..

The execution is carried out from the EC Scheduler. The example in the screen above executes a complete flow (or a typical flow option 2 according to the previous section). The folders specified for the business action is:

- Drop folder: The folder where the source file will be picked up.
- Completed folder: The folder where the read and successful files will be moved after the job transaction.
- Error folder: The folder where the failing files will be moved after the job transaction.
- File filter: The first part or complete name of the file(s) to be picked up. No asterix characters are supported.

If FTP file drop service is used, the Drop folder specifies the file path to access the files on FTP remote server and other folder options specified will be ignored. These FTP parameters should be specified:

- FTP Server Name: The name of the remote FTP server which you want to read files from.
- FTP User Name: The username which has access to the FTP remote server.
- FTP Password: The password which matches the user name to access the FTP remote server.
- FTP Port: The port number which the FTP remote server should connect to. If not specified will the default port

be used.

- **FTP Remove File**
  - N (default): To disable validation of removing the file on FTP remote server after dealing with that file.
  - Y: (To enable validation of removing files)

## Monitoring the Execution

The screenshot shows a web-based application interface for monitoring scheduled tasks. On the left, there is a sidebar with a 'FAVORITES' section and a 'MENU' section containing various links like Configuration, EC Production, EC Sales, etc. The main area is titled 'SCHEDULE' and contains a table with columns: Name, Description, Functional Area, Status, Pinned To, Next Fire Time, and Enabled. The table lists several tasks such as 'AudreyExcelImport', 'BOEConversionFactorSetup', 'BOLCalculation', 'BatchImportERPMsg', and 'BigImport'. Below the table is a date range selector from 'From Daytime' to 'To Daytime' with specific times: 2014-12-11 00:00 to 2014-12-12 00:00. A green arrow icon indicates the search or refresh action. To the right of the table is a 'PARAMETERS' section with a 'GET DETAILED LOG' button. This section shows parameters for the selected task, including jobid and AudreyJobID. Below this is a 'BUSINESS ACTION HISTORY' section with a table showing actions: Name, Description, Start Time, Duration, Step, and Status. One entry is shown: 'ECISAction' for 'AudreyExcelImport' at 2014-12-11 10:54, duration 00:00:00, step 10, status Warning. A detailed log pane on the right shows three log entries: 1 (WARN), 2 (WARN), and 3 (OK). The log pane has a scrollable list of log entries.

Name	Description	Functional Area	Status	Pinned To	Next Fire Time	Enabled
AudreyExcelImport		EC	WAITING	WS003040.U	2014-12-11 1	<input checked="" type="checkbox"/>
BOEConversionFactorSetup	Setup the inter-group conversion factor and queue quantity r	Revenue	EXPIRED			<input type="checkbox"/>
BOLCalculation	Load Calculation	Allocation	EXPIRED	WS002994.g		<input checked="" type="checkbox"/>
BatchImportERPMsg	Batch Import of ERP Messages	Revenue	EXPIRED			<input type="checkbox"/>
BigImport		EC	EXPIRED			<input type="checkbox"/>

Status	Start Time	Duration	Executed by	PARAMETERS		GET DETAILED LOG
WARNING	2014-12-11 10:54:00	00:00:00	WS003040.Un	Name	Value	
WARNING	2014-12-11 10:53:00	00:00:00	WS003040.Un	jobid	AudreyJobID	
MISFIRE	2014-12-11 10:52:12	00:00:00	WS003040.Un			
WARNING	2014-12-11 10:51:23	00:00:00	WS003040.Un			

Name	Description	Start Time	Duration	Step	Status
ECISAction	AudreyExcelImport	2014-12-11 10:54	00:00:00	10	Warning

1 (WARN)  
 2 (WARN)  
 3 (OK)

Monitoring the flow is only available after a finished transaction. The EC Scheduler History will provide a status and a summary of the execution. To get more technical details about the run, open the detailed log. The log level is controlled by the scheduler setting in the scheduler screen.

## Error handling

It will be possible to configure the error handling in two different ways.

One way is to handle the whole import job as one atomic transaction. If one or more errors occur the whole data import be rolled back.

The other way is to handle import as several transactions. Errors will then be reported and any successful data contents will be imported into the EC database.

Warnings will be reported and stored in the log and visible through the history screen.

One drop folder must be defined; two other directories must also be defined. One is the "done" folder for successfully loaded files and the other is the "error" folder which stores files which can not be fully loaded. If one or more errors occur the file be stored in the "error" folder.

The different types of transaction handling are defined on each interface. There are three or more important parameters per interface:

<b>Staging Validation</b>	This flag will prevent the data from being directly imported into EC. Manual verification by pressing the import data button in the staging screen must be performed before the data can be imported into EC.
<b>EC Validation Level</b>	Tells which level the interface is allowed to overwrite. The available statuses provided by the default installation of EC is P(rovisional), V(erified) and A(pproved). If the data stored in EC has record status set to 'A' and the EC Validation level is set to 'P' the interface will fail to update the rows within EC.
<b>EC Storage Data Level</b>	Tells which record status the rows inserted or updated in EC will get when running the interface.
<b>Transaction Type</b>	Ability to choose whether rows imported should be committed row for row or the whole job as one transaction. If row based transaction type is chosen the successful rows will be imported while the ones failing will not. If job transaction type is chosen all the data will be rolled back if one failure occurs.
<b>Overwrite</b>	This is for overwriting existing data in EC. These values can be used: <ul style="list-style-type: none"> <li>• Full: Always overwrite existing data in EC.</li> <li>• Incremental: Overwrite only if the new data from the file are different from the existing data in EC.</li> </ul>

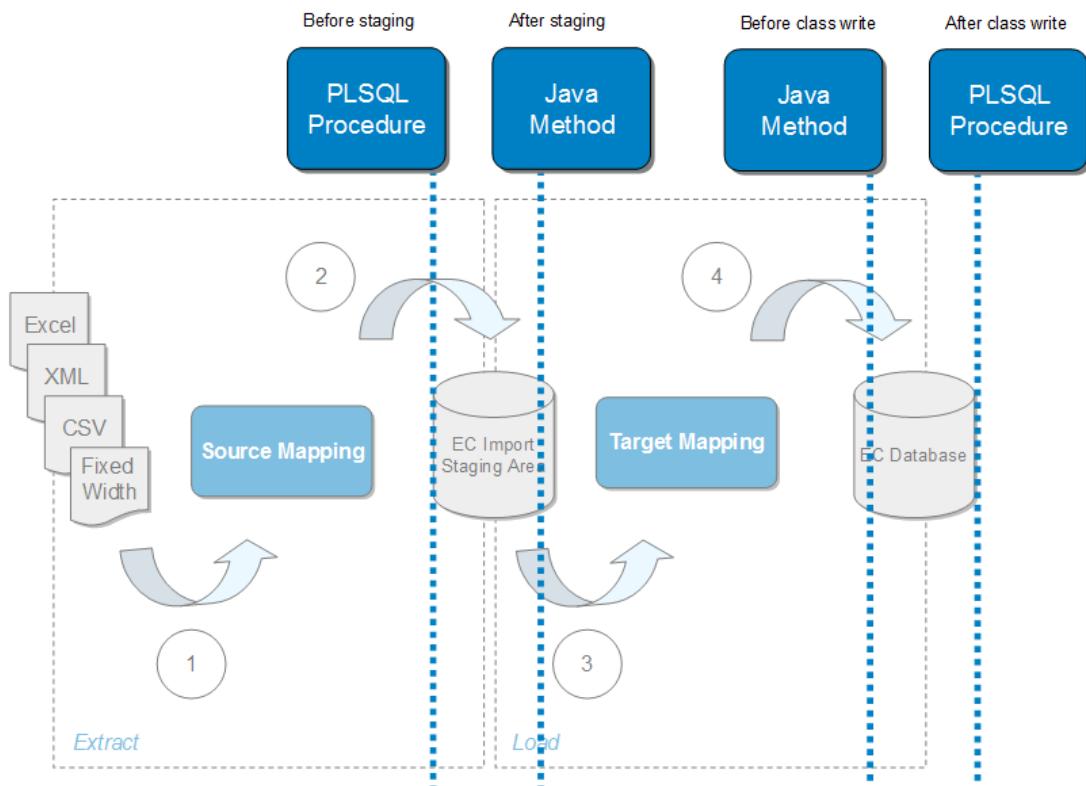
## User exits

User exits in the Advanced File Import context means a customized Java method or a PLSQL procedure which can run before or after specific events. The events supported are:

- Before writing to staging
- After writing to staging
- Before writing to EC
- After writing to EC

The subscription of the user events can be controlled at two levels:

- At interface level (the user exit will be fired for all mappings which belong to the interface).
- At mapping level (the user exit will be fired when a specific mapping is handled).



The user exit subscriptions can be defined at the source interface, source mapping and target mapping.

The user exit signatures must adhere to the following interface:

#### Java

```
/**
 * Perform user exit with input data provided.
 *
 * @param con the connection
 * @param defaultDTOBody the default dto body
 *
 * @throws UserExitException the user exit exception
 */
public void performUserExit(Connection con, DefaultDTOBody
defaultDTOBody) throws UserExitException;

/**
 * Perform user exit.
 *
 * @param con the connection
 * @throws UserExitException the user exit exception
 */
public void performUserExit(Connection con) throws UserExitException;
```

Examples of Java user exits can be found as plsql code in `com.ec.ecdm.is.advancedexcel/acceptancetest/com.ec.ecdm.advancedexcel.util/`

#### PLSQL

```
PLSQLPackage.Procedure(param1, param2..., paramN)
```

...where the input parameters can be of any type, but should correspond to the content of the DTO values, both when it comes to ordering and types.

Examples of PLSQL user exits can be found as plsql code in [com.ec.ecdm.is.advancedexcel/testdata/sql/](#)

## Mail Support

EC Import has a Business Action (BA) for pulling mail from a mail server, fetching the attachment, and writing it to one of three destinations. This can either be a drop folder, the database or passing it along to MHM. The business action that performs this is:

```
com.ec.ecdm.is.advancedexcel.mail.PullMailBusinessAction
```

Typical usage of this BA is to call it before the EC Import BA's are called. Here is a list of the different parameters the BA will use:

Parameter	Description
protocol	The protocol used by the mailserver, either POP3 or IMAP.
Server	The address to the mailserver
port	The mailserver port
user_id	The userid of the mail-user
password	The password of the mail-user
folder	Which mail folder to look for incoming messages in
drop_folder	If messages are to be written to a drop folder by the business action, this folder will be used.
Search	The search-string to look for in the mail subject. Used together with operator
operator	How to search for the search-string in the mail subject. The operator can have the following values:
EQUALS	The subject has to match the string given in the parameter search
STARTS_WITH	The subject has to start with the string given in the parameter search
CONTAINS	The subject has to contain the string given in the parameter search
ENDS_WITH	The subject has to end with the string given in the parameter search
target	Determine where the BA will put the data. Can have the following values:
MHM	Attachments will be saved to EC – MHM, and MHM will need to be configured for further handling of the data.
DATABASE	Attachments will be saved to the database
DROP_FOLDER	Attachments will be saved to the folder given in the parameter drop_folder
interface_code	This is the mapping against EC Import. The data will be saved with this interface_code in the database.
Unzip	Set to y if the attachment is zipped. The BA will then unzip the files to the destination.
Unzip_delete	If you want the zipped file to be deleted after it has been unzipped, set this to y.
ssl_required	If this parameter is set to y, the BA will use SSL to connect to the mail-server

## Database Details

### Source Interface

COLUMN_NAME	DATA_TYPE	NULLABLE
INTERFACE_CODE	VARCHAR2	N
NAME	VARCHAR2	Y
TYPE	VARCHAR2	Y
TRANSACTION_TYPE	VARCHAR2	Y
STAGING_VALIDATION_IND	VARCHAR2	Y
EC_VALID_LEVEL	VARCHAR2	Y
EC_DATA_LEVEL	VARCHAR2	Y
PRE_STAGING_UE_TYPE	VARCHAR2	Y
PRE_STAGING_UE_PATH	VARCHAR2	Y
POST_STAGING_UE_TYPE	VARCHAR2	Y
POST_STAGING_UE_PATH	VARCHAR2	Y
PRE_TARGET_UE_TYPE	VARCHAR2	Y
PRE_TARGET_UE_PATH	VARCHAR2	Y
POST_TARGET_UE_TYPE	VARCHAR2	Y
POST_TARGET_UE_PATH	VARCHAR2	Y
FORMAT	VARCHAR2	Y
SOURCE_TYPE	VARCHAR2	Y
OVERWRITE	VARCHAR2	Y
RECORD_STATUS	VARCHAR2	Y
CREATED_BY	VARCHAR2	N
CREATED_DATE	DATE	N
LAST_UPDATED_BY	VARCHAR2	Y
LAST_UPDATED_DATE	DATE	Y
REV_NO	NUMBER	Y
REV_TEXT	VARCHAR2	Y
APPROVAL_STATE	VARCHAR2	Y
APPROVAL_BY	VARCHAR2	Y
APPROVAL_DATE	DATE	Y
REC_ID	VARCHAR2	Y

### Source Mapping

COLUMN_NAME	DATA_TYPE	NULLABLE
IMP_SOURCE_MAPPING_NO	NUMBER	N
CODE	VARCHAR2	N
INTERFACE_CODE	VARCHAR2	N

SORT_ORDER	NUMBER	Y
NAME	VARCHAR2	Y
NAVIGATION_METHOD	VARCHAR2	Y
PATH_ORIGIN	VARCHAR2	Y
TYPE	VARCHAR2	Y
VALUE_TYPE	VARCHAR2	Y
STAGING_GROUP	VARCHAR2	Y
EC_KEY	VARCHAR2	Y
KEY_1	VARCHAR2	Y
KEY_2	VARCHAR2	Y
KEY_3	VARCHAR2	Y
KEY_4	VARCHAR2	Y
KEY_5	VARCHAR2	Y
KEY_6	VARCHAR2	Y
KEY_7	VARCHAR2	Y
KEY_8	VARCHAR2	Y
KEY_9	VARCHAR2	Y
KEY_10	VARCHAR2	Y
TRANSACTION_TYPE	VARCHAR2	Y
PRE_UETYPE	VARCHAR2	Y
PRE_UEPATH	VARCHAR2	Y
POST_UETYPE	VARCHAR2	Y
POST_UEPATH	VARCHAR2	Y
NAVIGATION_METHOD	VARCHAR2	Y
RECORD_STATUS	VARCHAR2	Y
CREATED_BY	VARCHAR2	N
CREATED_DATE	DATE	N
LAST_UPDATED_BY	VARCHAR2	Y
LAST_UPDATED_DATE	DATE	Y
REV_NO	NUMBER	Y
REV_TEXT	VARCHAR2	Y
APPROVAL_STATE	VARCHAR2	Y
APPROVAL_BY	VARCHAR2	Y
APPROVAL_DATE	DATE	Y
REC_ID	VARCHAR2	Y

Source Path

COLUMN_NAME	DATA_TYPE	NULLABLE
IMP_SOURCE_PATH_NO	NUMBER	N
IMP_SOURCE_MAPPING_NO	NUMBER	N
SORT_ORDER	NUMBER	Y
TYPE	VARCHAR2	Y
PATH	VARCHAR2	Y
PATH_PARAM_1	VARCHAR2	Y
PATH_PARAM_2	VARCHAR2	Y
PATH_PARAM_3	VARCHAR2	Y
RECORD_STATUS	VARCHAR2	Y
CREATED_BY	VARCHAR2	N
CREATED_DATE	DATE	N
LAST_UPDATED_BY	VARCHAR2	Y
LAST_UPDATED_DATE	DATE	Y
REV_NO	NUMBER	Y
REV_TEXT	VARCHAR2	Y
APPROVAL_STATE	VARCHAR2	Y
APPROVAL_BY	VARCHAR2	Y
APPROVAL_DATE	DATE	Y
REC_ID	VARCHAR2	Y

Source XML Path

COLUMN_NAME	DATA_TYPE	NULLABLE
IMP_SOURCE_XML_PATH_NO	NUMBER	N
IMP_SOURCE_MAPPING_NO	NUMBER	N
SORT_ORDER	NUMBER	Y
PATH	VARCHAR2	Y
TEXT_1	VARCHAR2	Y
TEXT_2	VARCHAR2	Y
TEXT_3	VARCHAR2	Y
TEXT_4	VARCHAR2	Y
TEXT_5	VARCHAR2	Y
RECORD_STATUS	VARCHAR2	Y
CREATED_BY	VARCHAR2	N
CREATED_DATE	DATE	N
LAST_UPDATED_BY	VARCHAR2	Y
LAST_UPDATED_DATE	DATE	Y
REV_NO	NUMBER	Y
REV_TEXT	VARCHAR2	Y
APPROVAL_STATE	VARCHAR2	Y
APPROVAL_BY	VARCHAR2	Y
APPROVAL_DATE	DATE	Y
REC_ID	VARCHAR2	Y

### Target Mapping

COLUMN_NAME	DATA_TYPE	NULLABLE
EC_KEY	VARCHAR2	N
CLASS	VARCHAR2	Y
ATTRIBUTE	VARCHAR2	Y
CLASS_KEY_1	VARCHAR2	Y
CLASS_KEY_2	VARCHAR2	Y
CLASS_KEY_3	VARCHAR2	Y
CLASS_KEY_4	VARCHAR2	Y
CLASS_KEY_5	VARCHAR2	Y
CLASS_KEY_6	VARCHAR2	Y
CLASS_KEY_7	VARCHAR2	Y
CLASS_KEY_8	VARCHAR2	Y
CLASS_KEY_9	VARCHAR2	Y
CLASS_KEY_10	VARCHAR2	Y
CONDITION_1	VARCHAR2	Y
CONDITION_2	VARCHAR2	Y
CONDITION_3	VARCHAR2	Y
CONSTANT_STRING_VALUE	VARCHAR2	Y
FROM_UNIT	VARCHAR2	Y
TO_UNIT	VARCHAR2	Y
CONSTANT_NUMBER_VALUE	NUMBER	Y
CONSTANT_DATE_VALUE	DATE	Y
PRE_UETYPE	VARCHAR2	Y
PRE_UEPATH	VARCHAR2	Y
POST_UETYPE	VARCHAR2	Y
POST_UEPATH	VARCHAR2	Y
RECORD_STATUS	VARCHAR2	Y
CREATED_BY	VARCHAR2	N
CREATED_DATE	DATE	N
LAST_UPDATED_BY	VARCHAR2	Y
LAST_UPDATED_DATE	DATE	Y
REV_NO	NUMBER	Y
REV_TEXT	VARCHAR2	Y
APPROVAL_STATE	VARCHAR2	Y
APPROVAL_BY	VARCHAR2	Y
APPROVAL_DATE	DATE	Y
REC_ID	VARCHAR2	Y



# EC FRMW Standard Reports Configuration Guide

## Introduction

There are several 'out of the box' reports available in the system. These are referred to as 'Standard Reports'. EC Framework contains standard reports for general purposes. Similar reports will also be available for the other products in EC. They can be used as templates for customer specific reports, or just out of the box.

The reports are based on JasperSoft, and the Jasper definition files are included in the frmw-report-web.war, found inside ec-app.ear. The files are found under the folder reports\jasper. When using these reports as the starting point for your own customer specific reports, these files should be copied to a separate customer specific war-file. They can be edited directly as a text-file or using a tool like IReport.

The following reports are currently supported:

- Check Rule Log
- User Access By User
- User Access By Role
- EC Role Privileges
- Expired Passwords
- EC Unit Conversion

To use a standard report a report template, report definition and a report needs to be created. The next sections will show how to configure each of these reports.

This section is organized with one chapter per report.

## Check Rule Log Report

The check rule log report shows open check rule validation issues reported in the EC check rule log. Acknowledged issues are not included in this report.

The report uses the following report specific parameters:

<b>FROMDATE</b>	Include log entries from this date.
<b>TODATE</b>	Include log entries to this date (inclusive).
<b>SYSTEM_NAME</b>	Name identifying the company/facility/system/etc of the report. Appears in the report header.

## Report Template

Configure the report template as shown below. Attributes with user defined values are not listed.

<b>System</b>	EC Internal		
<b>Template Name</b>	/com.ec.frmw.report/genericjasperreport		
Parameter Name	Parameter Type	Parameter Sub Type	Mandatory
<b>FROMDATE</b>	Basic Type	DATE	No
<b>TODATE</b>	Basic Type	DATE	No
<b>SYSTEM_NAME</b>	Basic Type	STRING	No
<b>JRXML</b>	Basic Type	STRING	Yes
<b>FORMAT</b>	EC Code Type	REPORT_FORMAT	Yes

## Report Definition

Configure the report definition as shown below. Attributes with user defined values are not listed.

<b>Functional Area</b>	EC
<b>JRXML</b>	/com.ec.frmw.report/reports/jasper/checkrule_log.jrxml
<b>FORMAT</b>	pdf

## User Access by user

The user access by user report shows all users who has privileged EC user access. These users also have Root-role and Web-access roles.

The report uses the following report specific parameters:

<b>DATE</b>	The Report Date.
<b>SYSTEM_NAME</b>	Name identifying the company/facility/system/etc of the report. Appears in the report header.

## Report Template

Configure the report template as shown below. Attributes with user defined values are not listed.

<b>System</b>	EC Internal		
<b>Template Name</b>	/com.ec.frmw.report/genericjasperreport		
Parameter Name	Parameter Type	Parameter Sub Type	Mandatory
<b>DATE</b>	Basic Type	DATE	No
<b>SYSTEM_NAME</b>	Basic Type	STRING	No
<b>JRXML</b>	Basic Type	STRING	Yes
<b>FORMAT</b>	EC Code Type	REPORT_FORMAT	Yes

## Report Definition

Configure the report definition as shown below. Attributes with user defined values are not listed.

<b>Functional Area</b>	EC
<b>JRXML</b>	/com.ec.frmw.report/reports/jasper/User_access_by_use.r.jrxml
<b>FORMAT</b>	pdf

## User Access by role

The user access by role report shows all roles that has privileged EC user access.

The report uses the following report specific parameters:

<b>DATE</b>	The Report Date.
<b>SYSTEM_NAME</b>	Name identifying the company/facility/system/etc of the report. Appears in the report header.

### Report Template

Configure the report template as shown below. Attributes with user defined values are not listed.

<b>System</b>	EC Internal		
<b>Template Name</b>	/com.ec.frmw.report/genericjasperreport		
Parameter Name	Parameter Type	Parameter Sub Type	Mandatory
<b>DATE</b>	Basic Type	DATE	No
<b>SYSTEM_NAME</b>	Basic Type	STRING	No
<b>JRXML</b>	Basic Type	STRING	Yes
<b>FORMAT</b>	EC Code Type	REPORT_FORMAT	Yes

### Report Definition

Configure the report definition as shown below. Attributes with user defined values are not listed.

<b>Functional Area</b>	EC		
<b>JRXML</b>	/com.ec.frmw.report/reports/jasper/User_access_by_role.jrxml		
<b>FORMAT</b>	pdf		

### EC role privileges

The role privileges report shows that which access level that each role has for the EC screens in the tree view.  
The report uses the following report specific parameters:

<b>DATE</b>	The Report Date.
<b>SYSTEM_NAME</b>	Name identifying the company/facility/system/etc of the report. Appears in the report header.

### Report Template

Configure the report template as shown below. Attributes with user defined values are not listed.

<b>System</b>	EC Internal		
<b>Template Name</b>	/com.ec.frmw.report/genericjasperreport		

Parameter Name	Parameter Type	Parameter Sub Type	Mandatory
DATE	Basic Type	DATE	No
SYSTEM_NAME	Basic Type	STRING	No
JRXML	Basic Type	STRING	Yes
FORMAT	EC Code Type	REPORT_FORMAT	Yes

### Report Definition

Configure the report definition as shown below. Attributes with user defined values are not listed.

Functional Area	EC
JRXML	/com.ec.frmw.report/reports/jasper/role_privileges_head.jrxml
FORMAT	pdf

### Expired passwords

The expired passwords report shows users' password expiry dates, and how long overdue the password change is.

The report uses the following report specific parameters:

DATE	The Report Date.
SYSTEM_NAME	Name identifying the company/facility/system/etc of the report. Appears in the report header.

### Report Template

Configure the report template as shown below. Attributes with user defined values are not listed.

System	EC Internal		
Template Name	/com.ec.frmw.report/genericjasperreport		
Parameter Name	Parameter Type	Parameter Sub Type	Mandatory
DATE	Basic Type	DATE	No
SYSTEM_NAME	Basic Type	STRING	No
JRXML	Basic Type	STRING	Yes
FORMAT	EC Code Type	REPORT_FORMAT	Yes

### Report Definition

Configure the report definition as shown below. Attributes with user defined values are not listed.

Functional Area	EC
-----------------	----

<b>JRXML</b>	/com.ec.frmw.report/reports/jasper/expired_passwords.jrxml
<b>FORMAT</b>	pdf

## Ec unit conversion

The EC unit conversion report shows the unit conversion configuration in EC and also, in case unit conversion has been overridden for an object, a subsection under that line which specifies what object had this unit conversion overridden and what the new values are after overriding them.

The report uses the following report specific parameters:

<b>DATE</b>	The Report Date.
<b>SYSTEM_NAME</b>	Name identifying the company/facility/system/etc of the report. Appears in the report header.

## Report Template

Configure the report template as shown below. Attributes with user defined values are not listed.

<b>System</b>	EC Internal		
<b>Template Name</b>	/com.ec.frmw.report/genericjasperreport		
Parameter Name	Parameter Type	Parameter Sub Type	Mandatory
<b>DATE</b>	Basic Type	DATE	No
<b>SYSTEM_NAME</b>	Basic Type	STRING	No
<b>JRXML</b>	Basic Type	STRING	Yes
<b>FORMAT</b>	EC Code Type	REPORT_FORMAT	Yes

## Report Definition

Configure the report definition as shown below. Attributes with user defined values are not listed.

<b>Functional Area</b>	EC
<b>JRXML</b>	/com.ec.frmw.report/reports/jasper/ec_unit_conversion.jrxml
<b>FORMAT</b>	pdf

# EC IS Configuration Guide

## Introduction

The section describes how to configure EC Integration Services for the most common data capture scenarios and should be read in conjunction with the [Integration Services Technical Documentation](#) which describes all features and possible options for the different EC IS components. It focuses on typical configuration scenarios and defines a step by step approach on how to capture the data required.

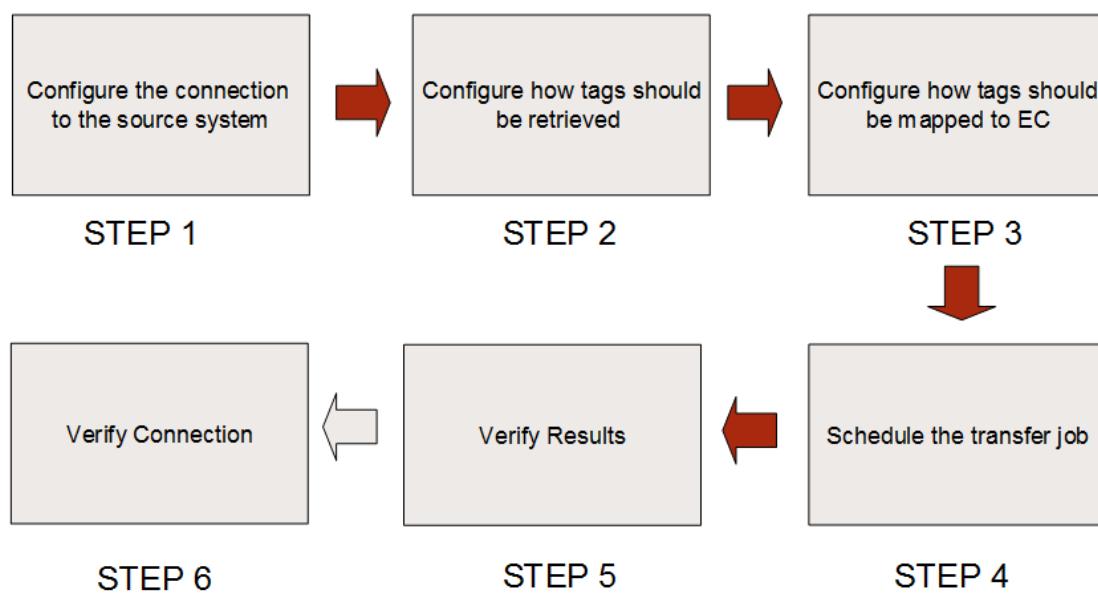
## Configuration Scenarios

Before starting this document, please see Integration Services Technical Documentation and perform the necessary steps.

## How to setup an OPC to EC connection

**Main objective:** Extract historian data from an OPC source and store it in EC classes.

**Flow objects description:** The configuration consists of six steps. First the connection to OPC is set up (Step 1), then which tags should be extracted and how they should be mapped and written to EC is set up (Step 2-3). The job is activated by scheduling it (Step 4). Then, optionally, the job can be verified (Step 5-6).



## **Step 1 – Configure the OPC connection**

- Open the file /<Wildfly>/domain/servers/ec-server/configuration/ecisconfig.xml.
  - Copy the following XML into the document.

```

<config name="OPCConfig">
    <sourceadapter>

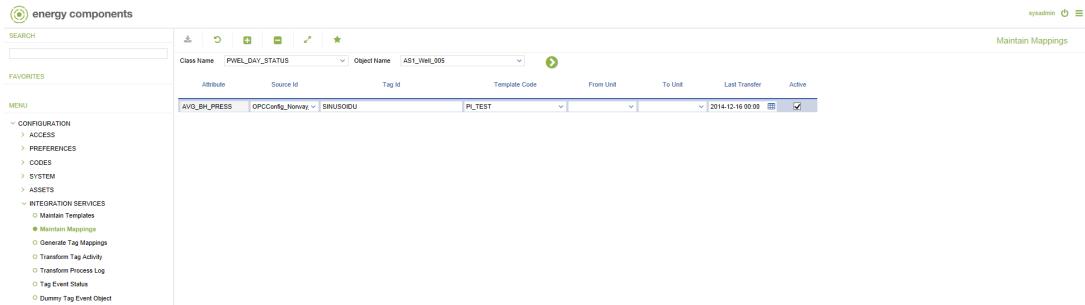
        <class>com.ec.frmw.is.engine.adapter.opc.OPCAdapter</class>
        <sequential>Yes/NO</sequential>
        <parameters>
            <parameter name="AUTH_DOMAIN">MyDomain</parameter>
            <parameter name="USR">opcreader</parameter>
            <parameter name="PWD">password</parameter>
            <parameter name="OPC_HOST">127.0.0.1</parameter>
            <parameter
name="PROG_ID">Some.OPC.Server.1</parameter>
            <parameter name="IID"></parameter>
        </parameters>
    </sourceadapter>
</config>

```

- Substitute the config name (ex. OPCConfig\_Norway\_1). No spaces are allowed in this name. The name is unique and cannot be used for any other config name.
- Set the sequential flag to 'Yes' (indicating that data from OPC will come ordered by date).
- Set the domain where the OPC server is located (AUTH\_DOMAIN parameter).
- Set the user name, password and host IP (USR, PWD and OPC\_HOST parameters).
- Set the program id **OR** the interface ID (PROG\_ID or IID parameter). The value of this one is given by the OPC server.
- Open the screen Configuration -> Codes -> EC Codes - All.
- Select the code type DT\_SOURCE\_ID.
- Add a new code equal to the config name in this step.

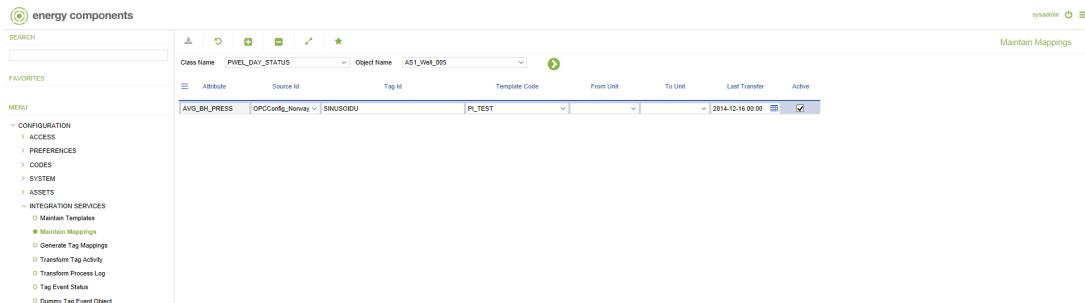
### **Step 2 – Configure the load templates**

- Ensure that you have identified the different source tags needed to do the data capture and how they should be aggregated in order to write to a final EC class.
- Open the screen Configuration -> Integration Service -> Maintain Templates.
- For each combination of aggregation type and resolution (the length of an aggregated period), create a new template and give information about the following details:
  - *Template Code*: A unique code to identify the template.
  - *Description*: Description to identify the template.
  - *Source Function*: AVG, MIN, MAX, MEAN, SUM defining how the source system should aggregate when extracting values for each tag. If no aggregation is needed at the source side use SAMPLE as the function.
  - *Source DataType*: DATE, DECIMAL, STRING defining the source system data type.
  - *Source Interval*: The length of the aggregated time intervals given in seconds.
  - *Source Delay*: The number of seconds between the execution time to the latest available value.
  - *Minimum Samples*: The number of values necessary for doing an aggregation at the target / write side instead of the source side. This parameter may be 1 if the target function is SAMPLE.
  - *Target Interval*: The length of the aggregation time intervals given in seconds. This parameter may be set equal to the source interval if no aggregation is required at the target side.
  - *Target Function*: AVG, MIN, MAX, MEAN, SUM or LATEST defining how to aggregate the extracted values. If no aggregation is needed at the target side use SAMPLE as the function.
  - *Target Date Time*: The name of the date time attribute in the final EC class. This should in most cases be set to "DAYTIME".
  - *Use Summer Time*: Set this if the final EC class needs a daylight saving flag as a part of the primary key.
  - *Overwrite User*: Set this if existing rows in an EC class (updated by non-system users) can be updated by EC IS. How to specify system users is described in the EC IS Technical Reference.
  - *Overwrite Status*: Set this to "Provisional", "Verified" or "Approved" to indicate the highest status of the data EC IS may be allowed to overwrite.
  - *Production Day Start*: The number of offset hours from midnight to the start of the production day.
  - *Shift Time To Period Start*: Define the number of shift times to the period start.



### Step 3 – Configure the tag mappings

- Identify all the tags needed to extract and the class attributes to write data into.
- Open the screen Configuration -> Integration Service -> Maintain Mappings.
- Choose the EC class to provide mappings for. Remember that ECIS needs access to be able to write to this class. Add the class as an object of type class in the Object Maintenance screen, and add the roles you want to be able to update it.
- Choose the specific object to provide mappings for.
- Create one mapping for each class attribute and provide the following information.
  - **Attribute:** The name of class attribute.
  - **Source Id:** A reference to one of the available sources, ex. OPCConfig\_Norway\_1 (ref. Step 1. Will be used for grouping the mappings between different jobs). If the source id cannot be found, please check the technical reference for how to register new sources in EC.
  - **Tag Id:** The name of the tag in the source system providing values for this specific attribute on the selected object. (Note: If the same source tag will be used for many different attributes please postfix the next mappings with a "-COPY", ex. TAG\_A-COPY-1).
  - **Template Code:** A reference to one of the templates inserted in step 2.
  - **From Unit:** Select the unit of measure for the values extracted from the source system.
  - **To Unit:** Select the standard unit of measure for the EC class attribute. If the class has an already defined UOM attribute, the "To Unit" property may not be set.
  - **Last Transfer:** Set the date and time specifying the latest sample time for the attribute. This setting is also used for forcing EC IS to perform a recapture of data (please check the EC IS Technical Reference for further details on this property).
  - **Active:** Turn on the mapping.
  - **Attribute Primary Key <n>:** Specify additional PK columns for the class.
  - **Value of Primary Key <n>:** Specify additional PK columns for the class.



### Step 4 – Schedule a transfer job

- Open the scheduler screen Configuration -> Schedules.
- Create a new schedule.
- Set loglevel and user in the Details tab. Also set retain count if you want EC to do housekeeping and not keep all schedule histories.  
Go to the Business Action tab, and create a business action for the schedule. Choose the ECISDataTransfer action and set the sequence number to 1. This should exist in the list as a default value. If not:
  - Open the Configuration -> Scheduler -> Business Actions screen.
  - Add a new Business Action.
  - Set the name to ECISDataTransfer.
  - Set the action reference to com.ec.frmw.is.engine.source.action.SourceAction.
  - Save Business Action.
- Select the source system (configurationid). This should reflect the configuration chosen in the ecisconfig.xml and used as a part of the tag mappings. In this example the OPCConfig\_Norway\_1 has been used.
- Set an identifier for the job. This is not affecting how the system runs but is rather an internal descriptor for the job.

- Go to the Schedule tab, choose the type of schedule you want to run (Cron, Once etc.), and set the timing details.
- Save the settings.
- Go to the Details tab again, and enable the schedule.

The screenshot shows the 'Schedules' screen in the energy components application. A job named 'PI\_TEST' is selected. The 'BUSINESS ACTION' tab is active, displaying parameters and macro definitions. The 'DETAILS' tab shows the schedule configuration, including the job's name, description, functional area, status, and execution time.

### Step 5 – Verify test run

- Go to the Monitor tab in the scheduler.
- Check the status of the reading job.
- Open the scheduler log for further details.
- Open the status screen Configuration -> Integration Service -> Transform Status Log.
- Check details of the writing.

The screenshot shows the 'Monitor' screen in the energy components application. It displays a log entry for a job named 'PI\_TEST' with a status of 'OK' and a duration of 00:00:06. The log entry includes the start time (2014-12-16 09:33:06) and the executed by field (WL041176.IZ).

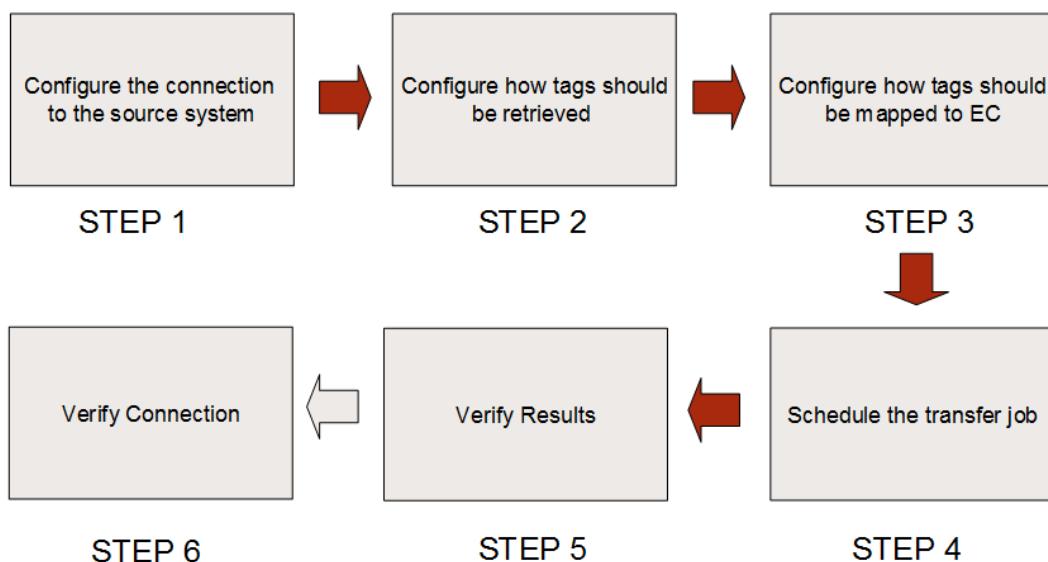
The screenshot shows the 'Transform Process Log' screen in the energy components application. It displays a log message for a service named 'Transform Process Log' with a log level of 'Info' and a daytime of '2014-12-01 07:20'. The log message indicates 'No records found'.

### How to setup a PI to EC connection

**Main objective:** Extract historian data from a PI source and store it in EC classes.

**Flow objects description:** The configuration consists of six steps. First the connection to PI is set up (Step 1), then which tags to extract and how they are mapped and written to EC is configured (Step 2 - 3). The job is activated by scheduling it (Step 4). The job can then be verified with two optional steps (Step 5 - 6).

Be aware that this guide uses the J-interop version of the PI adapter. The recommended adapter to use is the PI JDBC adapter. Please refer to the Integration Services Technical Documentation for the changes needed if the PI JDBC adapter is to be used.



#### **Step 1 – Configure the PI connection**

- Open the file </Wildfly>/domain/servers/ec-server/configuration/ecisconfig.xml.
- Copy the following XML into the document.

```

<config name="PI_SOURCE_1">
    <dtomergeenabled>true</dtomergeenabled>
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.pi.jinterop.PiAdapter</class>
        <sequential>Yes</sequential>
        <parameters>
            <parameter name="authdomain">MyDomain</parameter>
            <parameter name="domainuser">auser</parameter>
            <parameter name="domainpwd">password</parameter>
            <parameter name="dsn">127.0.0.1</parameter>
            <parameter name="username">piadmin</parameter>
            <parameter name="pwd"></parameter>
            <parameter name="timeout">180</parameter>
            <parameter
                name="logfile">c:\temp\logfile2.log</parameter>
        </parameters>
    </sourceadapter>
</config>

```

- Substitute the config name (ex. PIConfig\_Norway\_1). No spaces are allowed in this name. The name is unique and cannot be used for any other config name.

- Set the sequential flag to 'Yes' (indicating that data from PI will come ordered by date).
- Set the domain where the PI server is located (AUTH\_DOMAIN parameter).
- Set the domainuser and the domainpwd that enables ECIS to connect to Windows COM server and get a COM object. This user needs to have DCOM access. See <http://www.j-interop.org/quickstart.html> for more info about how to setup this. There has also been some trouble related to getting access on Windows Server 2008 64 bit. See the Integration Services Technical Documentation section about PI-adapter for more information around this.
- Set the user name, password and host IP (dsn, username and pwd parameters).
- Setup the logfile parameter to point to a log directory. Be aware that there is no housekeeping on this file, so you should remove the logfile parameter while running in production, or setup some project specific housekeeping for this file.
- Open the screen Configuration -> Codes -> EC Codes - All.
- Select the code type DT\_SOURCE\_ID.
- Add a new code equal to the config name in this step.

### **Step 2 - 5**

- Step 2 – 5 can be done by following the description of the step for the OPC scenario. (Step 3 – Configure the tag mappings, Step 4 – Schedule a transfer job, Step 5 – Verify test run).

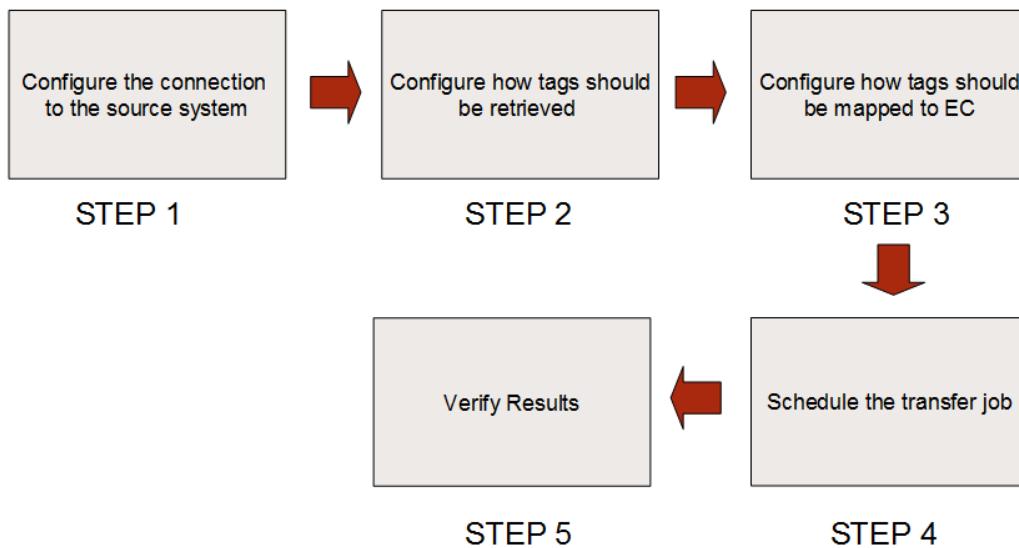
### **Step 6 – Verify connection**

- If the verification in step 5 failed because of connectivity problems to PI, please recheck your settings.

### **How to setup a tag based database to EC connection**

**Main objective:** Extract tag based data from a database source and store it in EC classes.

**Flow objects description:** The configuration consists of five steps. First a connection to the database source is set up (Step 1), then the tags to extract and how they are mapped and written to EC is configured (Step 2 - 3). The job is activated by scheduling it (Step 4). To verify the job an optional step can be taken (Step 5).



### **Step 1 – Configure the tag based database connection**

- Open the file /<Wildfly>/domain/servers/ec-server/configuration/ecisconfig.xml.
- Copy the following XML into the document.

```

<config name="tagBasedJdbcConfig">
  <sourceadapter>

    <class>com.ec.frmw.is.engine.adapter.jdbc.SourceTagJdbcAdapter</c
lass>
    <sequential>Yes</sequential>
    <parameters>
      <parameter
name="DRIVER">oracle.jdbc.driver.OracleDriver</parameter>
      <parameter
name="DB_URL">jdbc:oracle:thin:@host:1521:SID</parameter>
      <parameter name="DB_USER">user</parameter>
      <parameter name="DB_PWD">pwd</parameter>
      <parameter name="SQL_TAG_ID">IDENTIFIER</parameter>
      <parameter
name="SQL_TABLE">db_name.SOMETABLE</parameter>
      <parameter name="SQL_VALUE">SOMEVALUE</parameter>
      <parameter name="SQL_DAYTIME">DAYTIME</parameter>
    </parameters>
  </sourceadapter>
</config>

```

- Substitute the config name (ex. TagConfig\_Norway\_1). No spaces are allowed in this name. The name is unique and cannot be used for any other config name.
- Set the sequential flag to 'Yes' (indicating that data from the database will come ordered by date). This flag assumes that the SQL sent to the database source system is ordering the output by date. If you cannot order it, set the flag to 'No'.
- Set the database driver used (DRIVER parameter, leave the default value if using an Oracle database).
- Set the user name, password and database URL (DB\_USER, DB\_PWD and DB\_URL parameters).
- The other SQL parameters are all optional and used only if you need to set up your own SQLs. To do so please refer to the technical documentation (see: Integration Services Technical Documentation).
- Open the screen Configuration -> Codes -> EC Codes - All.
- Select the code type DT\_SOURCE\_ID.
- Add a new code equal to the config name in this step.

### **Step 2 – Configure the load templates**

- Please refer to the step 2 in the OPC section.
- Exception: Tag based JDBC to EC connection only supports source function SAMPLE and LATEST.

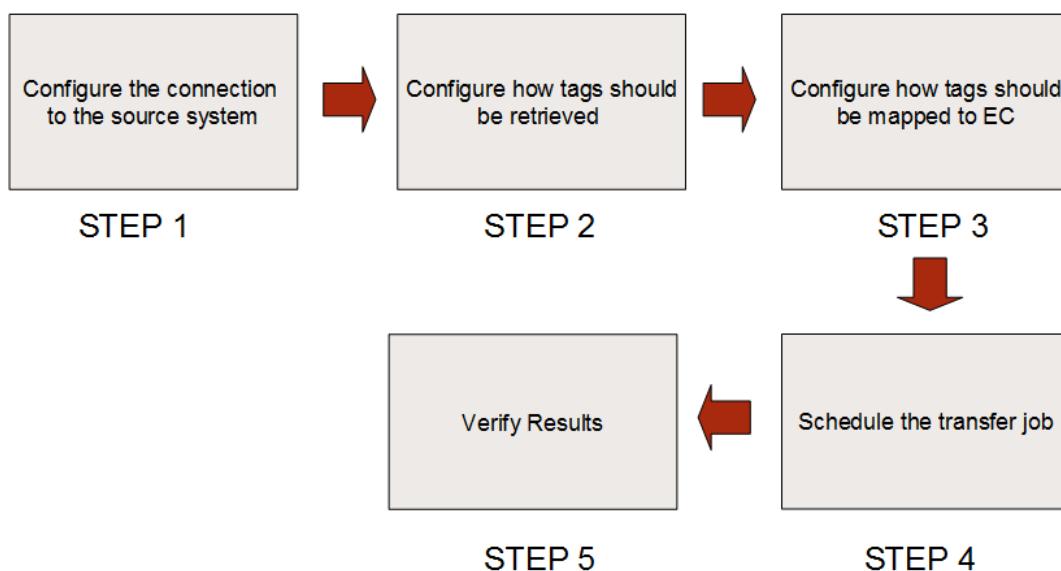
### **Step 3 - 5**

Please refer to step 3 - 5 in the OPC section (Step 3 – Configure the tag mappings, Step 4 – Schedule a transfer job, Step 5 – Verify test run).

### **How to setup a tag based Excel to EC connection**

**Main objective:** Extract tag based data from an Excel file and store it in EC classes.

**Flow objects description:** The configuration consists of five steps. First the connection to a file is set up (Step 1), then which tags to extract and how they should be mapped and written to EC is configured (Step 2 - 3). The job is activated by scheduling it (Step 4). The job can be verified with an optional step (Step 5).



#### **Step 1 – Configure the tag based Excel connection**

- Open the file /<Wildfly>/domain/servers/ec-server/configuration/ecisconfig.xml.
- Copy the following XML into the document.

```

<config name="FILEADAPTER" >
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
            <sequential>Yes</sequential>
            <parameters>
                <parameter name="DropFolder">c:\\temp</parameter>
                <parameter
name="CompletedFolder">c:\\temp\\completed</parameter>
                <parameter
name="ErrorFolder">c:\\temp\\error</parameter>
                <parameter name="BadFolder">c:\\temp\\bad</parameter>
                <parameter name="DateFormat">dd.MM.yyyy
HH:mm:ss</parameter>
                <parameter name="FileFilter"></parameter>
                <parameter name="StartCell">A1</parameter>
                <parameter name="EndCell">E104</parameter>
                <parameter name="QualityColumnIndex">0</parameter>
                <parameter name="TagColumnIndex">1</parameter>
                <parameter name="TimeColumnIndex">2</parameter>
                <parameter name="ValueColumnIndex">3</parameter>
            </parameters>
        </sourceadapter>
    </config>

```

- Substitute the config name (ex. TagConfig\_Norway\_1). No spaces are allowed in this name. The name is unique and cannot be used for any other config name.
- Set the sequential flag to 'Yes' (indicating that data from file will come ordered by date).
- Set the drop folder where the file containing data are.
- Set the error folder where data with errors will be written.
- Set the bad and the completed folder where the file will be moved.

- Set the date format used in the file (The format is based on a composite expression ex. yyyy-MM-dd HH:mm:ss).
  - Year (yyyy or yy)
  - Month (MM or MMM)
  - Day (dd)
  - Hour (HH)
  - Minute (mm)
  - Second (ss)
  - Spacing characters (- . etc)
- Set the file filter that will match the file to read (if empty then all files will be read).
- Specify the cell where the range of data in the Excel sheet starts (StartCell).
- Specify the cell where the range of data in the Excel sheet ends (EndCell).
- Set the column index for the quality, tag, time and value which the index of these columns in the files (if one of these columns is not present in the file just remove it).
- Open the screen Configuration -> Codes -> EC Codes - All.
- Select the code type DT\_SOURCE\_ID.
- Add a new code equal to the config name in this step.

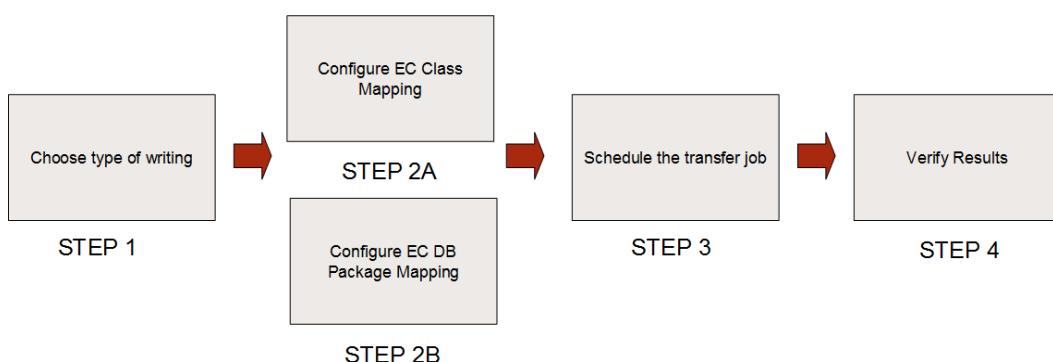
## Step 2 - 5

- Please refer to the step 2 - 5 in the OPC section (Step 3 – Configure the tag mappings, Step 4 – Schedule a transfer job, Step 5 – Verify test run)

## How to setup a general Excel to EC connection

**Main objective:** Extract rows from an Excel file and store them in EC.

**Flow objects description:** The configuration consists of four steps. First a connection to the file is set up along with how it should be written (Step 1 - 2). The job is activated by scheduling it (Step 3). To verify the job an optional step can be taken (Step 4).



### Step 1 – Choose type of writing

- If you are writing directly to an EC class go to step 2A.
- If you are writing to an EC DB package go to step 2B.

### Step 2A – Configure the Excel connection for EC class writing

- Open the file /<Wildfly>/domain/servers/ec-server/configuration/ecisconfig.xml.
- Copy the following XML into the document.

```

<config name="excel_connection">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.RowFileAdapter</class>
        <parameters>
            <parameter
name="DropFolder">c:\\temp\\\\jboss\\\\rowadapter</parameter>
            <parameter>

```

```

name="CompletedFolder">c:\\temp\\jboss\\rowadapter\\completed</parameter>
    <parameter
name="ErrorFolder">c:\\temp\\jboss\\rowadapter\\error</parameter>
    <parameter
name="BadFolder">c:\\temp\\jboss\\rowadapter\\bad</parameter>
    <parameter
name="DateFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
    <parameter name="FileFilter">rowdata</parameter>
    <parameter name="StartCell">A1</parameter>
    <parameter name="EndCell">E104</parameter>
    <parameter
name="TargetDriver">oracle.jdbc.driver.OracleDriver</parameter>
    <parameter name="TargetDbURL">
jdbc:oracle:thin:@<host>:1521:<SID></parameter>
        <parameter name="TargetDbUser">dbuser</parameter>
        <parameter name="TargetDbPassword">pwd</parameter>
        <parameter name="EcClass">PWEL_SAMPLE_1</parameter>
        <parameter name="Col_1" type="list">
            <subparameter name="columnIndex">0</subparameter>
            <subparameter
name="columnType">Date</subparameter>
            <subparameter name=" EcClassAttribute
">DAYTIME</subparameter>
            </parameter>
            <parameter name="Col_2" type="list">
                <subparameter name="columnIndex">1</subparameter>
                <subparameter
name="columnType">Double</subparameter>
                <subparameter name=" EcClassAttribute
">AVG_WH_PRESS</subparameter>
                </parameter>
                <parameter name="Col_3" type="list">
                    <subparameter name="columnIndex">2</subparameter>
                    <subparameter
name="columnType">Double</subparameter>
                    <subparameter name=" EcClassAttribute
">WH_TEMP</subparameter>
                    </parameter>
                    <parameter name="Col_4" type="list">
                        <subparameter name="columnIndex">3</subparameter>
                        <subparameter
name="columnType">String</subparameter>
                        <subparameter name=" EcClassAttribute
">MAX SAND RATE</subparameter>
                        </parameter>
                        <parameter name="Col_5" type="list">
                            <subparameter name="columnIndex">4</subparameter>
                            <subparameter
name="columnType">String</subparameter>
                            <subparameter name=" EcClassAttribute
">OBJECT_ID</subparameter>
                            <subparameter name=" EcAttributeFunction
">UE_ECISFUNCTION.mapObjectID</subparameter>
                            </parameter>
                            <parameter name="Col_6" type="list">
                                <subparameter name="columnIndex">5</subparameter>

```

```
<subparameter  
name="columnType">String</subparameter>  
    <subparameter name=" EcClassAttribute  
">OBJECT_CODE</subparameter>  
        </parameter>  
        <parameter  
name="ErrorFile">c:\\temp\\jboss\\rowadapter\\errors\\errorfile.l  
og</parameter>
```

```

    </parameters>
  </sourceadapter>
</config>

```

- Substitute the config name (ex. ExcelConfig\_Norway\_1). No spaces are allowed in this name. The name is unique and cannot be used for any other config name.
- Set the folder parameters describing where to pick up the files (DropFolder), where to move it after it has been completed / read (CompletedFolder), and where to write error files (ErrorFolder and BadFolder). The file filter (FileFilter) can be left empty (all files will be picked up) or set to a substring of the file name. No asterisks are needed in order to specify a file filter, ex. 'rowdata' will pick up all files containing 'rowdata' as part of the file name.
- Specify the cell where the range of data in the Excel sheet starts (StartCell).
- Specify the cell where the range of data in the Excel sheet ends (EndCell).
- Set the DB connection details (TargetDriver, TargetDbUrl, TargetDBUser and TargetDbPassword).
- Set the name of the EC class which will be used to store the rows (EcClass).
- For each column in the Excel range specify a Column (Col\_x) with the attribute mapping to the EC class (You can also specify a function to fill out the class attribute – please see the "Integration Services Technical Documentation").
- Set the name of the error log file (ErrorFile).
- Open the screen Configuration -> Codes -> EC Codes - All.
- Select the code type DT\_SOURCE\_ID.
- Add a new code equal to the config name in this step.

#### **Step 2B – Configure the Excel connection for EC package writing**

- Open the file /<Wildfly>/domain/servers/ec-server/configuration/ecisconfig.xml.
- Copy the following XML into the document.

```

<config name="excel_connection">
  <sourceadapter>

    <class>com.ec.frmw.is.engine.adapter.file.RowFileAdapter</class>
    <parameters>
      <parameter
        name="DropFolder">c:\\temp\\jboss\\rowadapter</parameter>
      <parameter
        name="CompletedFolder">c:\\temp\\jboss\\rowadapter\\completed</pa
        rameter>
      <parameter
        name="ErrorFolder">c:\\temp\\jboss\\rowadapter\\error</parameter>
      <parameter
        name="BadFolder">c:\\temp\\jboss\\rowadapter\\bad</parameter>
      <parameter
        name="DateFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
        <parameter name="FileFilter">rowdata</parameter>
        <parameter name="StartCell">A1</parameter>
        <parameter name="EndCell">E104</parameter>
        <parameter
          name="DbPackageProcedure">UE_ECISROWTEST.ReceiveTestdat</paramete
          r>
        <parameter
          name="TargetDriver">oracle.jdbc.driver.OracleDriver</parameter>
          <parameter name="TargetDbURL">
            jdbc:oracle:thin:@<host>:1521:<SID></parameter>
          <parameter name="TargetDbUser">dbuser</parameter>
          <parameter name="TargetDbPassword">pwd</parameter>
          <parameter name="Col_1" type="list">
            <subparameter name="columnIndex">0</subparameter>

```

```
        <subparameter  
name="columnType">Date</subparameter>  
        <subparameter  
name="packageIndex">1</subparameter>  
        </parameter>  
        <parameter name="Col_2" type="list">  
            <subparameter name="columnIndex">1</subparameter>  
            <subparameter  
name="columnType">Double</subparameter>  
            <subparameter  
name="packageIndex">2</subparameter>  
            </parameter>  
            <parameter name="Col_3" type="list">  
                <subparameter name="columnIndex">2</subparameter>  
                <subparameter  
name="columnType">Double</subparameter>  
                <subparameter  
name="packageIndex">3</subparameter>  
                </parameter>  
                <parameter name="Col_4" type="list">  
                    <subparameter name="columnIndex">4</subparameter>  
                    <subparameter  
name="columnType">String</subparameter>  
                    <subparameter  
name="packageIndex">4</subparameter>  
                    </parameter>  
                    <parameter  
name="ErrorFile">c:\\temp\\jboss\\rowadapter\\errors\\errorfile.l  
og</parameter>
```

```

</parameters>
</sourceadapter>
</config>

```

- Substitute the config name (ex. ExcelConfig\_Norway\_1). No spaces are allowed in this name. The name is unique and cannot be used for any other config name.
- Set the folder parameters describing where to pick up the files (DropFolder), where to move it after it has been completed / read (CompletedFolder), and where to write error files (ErrorFolder and BadFolder). The file filter (FileFilter) can be left empty (all files will be picked up) or set to a substring of the file name. No asterisks are needed in order to specify a file filter, ex. 'rowdata' will pick up all files containing the 'rowdata' as the part of the file name.
- Specify the cell where the range of data in the Excel sheet starts (StartCell).
- Specify the cell where the range of data in the Excel sheet ends (EndCell).
- Set the name of the DB package and function / procedure which will be used to store the rows (DBPackageProcedure).
- Set the DB connection details (TargetDriver, TargetDbUrl, TargetDbUser and TargetDbPassword).
- For each column you want to extract make a reference to the column index (columnIndex) in the source file and the parameter index (packageIndex) in the package call. The column index starts at 0 and the package index starts at 1.
- Set the name of the error log file (ErrorFile).
- Open the screen Configuration -> Codes -> EC Codes - All.
- Select the code type DT\_SOURCE\_ID.
- Add a new code equal to the config name in this step.

### **Step 3 – Schedule a transfer job**

Schedule a transfer job the same way as shown in the OPC section (Step 4 – Schedule a transfer job).

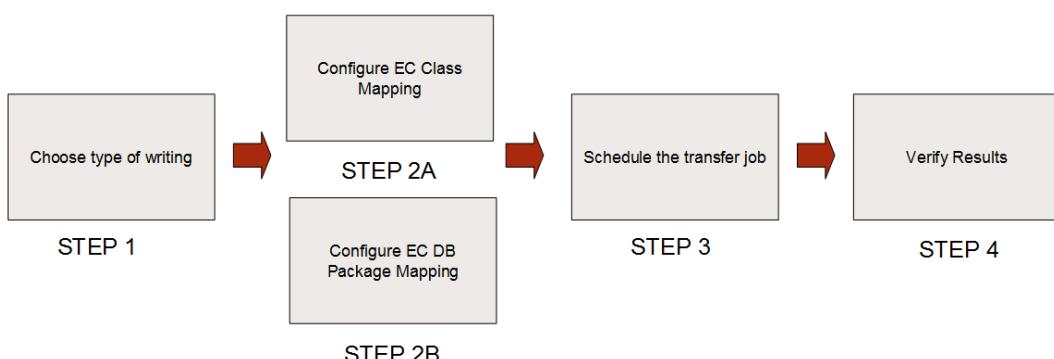
### **Step 4 – Verify test run**

Verify a test run the same way as in the OPC section (Step 5 – Verify test run).

## **How to setup a general database to EC connection**

**Main objective:** Extract rows from a database and store them in EC.

**Flow objects description:** The configuration consists of four steps. First a connection to the database is set up along with how it should be written (Step 1 - 2). The job is activated by scheduling it (Step 3). The job can be verified with an optional step (Step 4).



### **Step 1 – Choose type of writing**

- If you are writing directly to an EC class go to step 2A.
- If you are writing to an EC DB package go to step 2B.

### **Step 2A – Configure the database connection for EC class writing**

- Open the file /<Wildfly>/domain/servers/ec-server/configuration/ecisconfig.xml.
- Copy the following XML into the document.

```

<config name="database_connection">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.jdbc.SourceJdbcAdapterCollection</class>
            <parameters>
                <parameter
name="DRIVER">Oracle.jdbc.driver.OracleDriver</parameter>
                <parameter name="DB_URL">
jdbc:oracle:thin:@<host>:1521:<SID></parameter>
                <parameter name="DB_USER">dbuser</parameter>
                <parameter name="DB_PWD">pwd</parameter>
                <parameter name="SQL "></parameter>
                <parameter name="EcClass">PWEL_SAMPLE_1</parameter>
                <parameter name="Col_1" type="list">
                    <subparameter name=" EcClassAttribute
">DAYTIME</subparameter>
                </parameter>
                <parameter name="Col_2" type="list">
                    <subparameter name=" EcClassAttribute
">AVG_WH_PRESS</subparameter>
                </parameter>
                <parameter name="Col_3" type="list">
                    <subparameter name=" EcClassAttribute
">WH_TEMP</subparameter>
                </parameter>
                <parameter name="Col_4" type="list">
                    <subparameter name=" EcClassAttribute
">MAX SAND RATE</subparameter>
                </parameter>
                <parameter name="Col_5" type="list">
                    <subparameter name=" EcClassAttribute
">OBJECT_ID</subparameter>
                </parameter>
                <subparameter name=" EcAttributeFunction
">UE_ECISFUNCTION.mapObjectID</subparameter>
                </parameter>
                <parameter name="Col_6" type="list">
                    <subparameter name=" EcClassAttribute
">OBJECT_CODE</subparameter>
                </parameter>
                <parameter
name="ErrorFile">c:\\temp\\jboss\\rowadapter\\errors\\errorfile.log</parameter>
            </parameters>
        </sourceadapter>
    </config>

```

- Substitute the config name (ex. DBConfig\_Norway\_1). No spaces are allowed in this name. The name is unique and cannot be used for any other config name.
- Set the DB connection details (DRIVER, DB\_URL, DB\_USER and DB\_PWD).
- Set the name of the EC class which will be used to store the rows (EcClass).
- For each column in the Excel range specify a Column (Col\_x) with the attribute mapping to the EC class (You

can also specify a function to fill out the class attribute – please see the "Integration Services Technical Documentation".

- Open the screen Configuration -> Codes -> EC Codes - All.
- Select the code type DT\_SOURCE\_ID.
- Add a new code equal to the config name in this step.

In some cases you might want to load data into several different EC classes in the same run. In order to configure such a connection properly, see the "Integration Services Technical Documentation".

#### ***Step 2B – Configure the database connection for EC package writing***

- Open the file /<Wildfly>/domain/servers/ec-server/configuration/ecisconfig.xml.
- Copy the following XML into the document.

```

<config name=" database_connection ">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.jdbc.SourceJdbcAdapterCollection </class>
        <parameters>
            <parameter
name="DbPackageProcedure">UE_ECISROWTEST.ReceiveTestdat</parameter>
            <parameter
name="DRIVER">Oracle.jdbc.driver.OracleDriver</parameter>
            <parameter name="DB_URL">
jdbc:oracle:thin:@<host>:1521:<SID></parameter>
            <parameter name="DB_USER">dbuser</parameter>
            <parameter name="DB_PWD">pwd</parameter>
            <parameter name="Col_1" type="list">
                <subparameter name="columnIndex">0</subparameter>
                <subparameter
name="columnType">Date</subparameter>
                <subparameter
name="packageIndex">1</subparameter>
                </parameter>
            <parameter name="Col_2" type="list">
                <subparameter name="columnIndex">1</subparameter>
                <subparameter
name="columnType">Double</subparameter>
                <subparameter
name="packageIndex">2</subparameter>
                </parameter>
            <parameter name="Col_3" type="list">
                <subparameter name="columnIndex">2</subparameter>
                <subparameter
name="columnType">Double</subparameter>
                <subparameter
name="packageIndex">3</subparameter>
                </parameter>
            <parameter name="Col_4" type="list">
                <subparameter name="columnIndex">4</subparameter>
                <subparameter
name="columnType">String</subparameter>
                <subparameter
name="packageIndex">4</subparameter>
                </parameter>
            <parameter
name="ErrorFile">c:\\temp\\jboss\\rowadapter\\errors\\errorfile.log</parameter>
        </parameters>
    </sourceadapter>
</config>

```

- Substitute the config name (ex. DBConfig\_Norway\_1). No spaces are allowed in this name. The name is unique and cannot be used for any other config name.
- Set the name of the DB package and function / procedure which will be used to store the rows (DBPackageProcedure).
- Set the DB connection details (TargetDriver, TargetDbUrl, TargetDbUser and TargetDbPassword).

- For each column you want to extract make a reference to the column index (columnIndex) in the source file and the parameter index (packageIndex) in the package call. The column index starts at 0 and the package index starts at 1.
- Set the name of the error log file (ErrorFile).
- Open the screen Configuration -> Codes -> EC Codes - All.
- Select the code type DT\_SOURCE\_ID.
- Add a new code equal to the config name in this step.

### **Step 3 – Schedule a transfer job**

Schedule a transfer job the same way as in the OPC section (Step 4 – Schedule a transfer job).

### **Step 4 – Verify test run**

Verify a test run the same way as in the OPC section (Step 5 – Verify test run).

## **Troubleshooting**

### **Problem – General issues**

Before reporting any errors there are some common issues that should be checked. Some general error sources are:

- The file that is supposed to be handled is not present at the path where the configuration says it should be.
- The file that is supposed to be handled is open in an editor.
- Date format is wrong.
- Last transfer is wrong.
- Mapping is not activated or mapping is wrong.

# EC IS Staging Table Extension

## Introduction

This section describes the architecture and functionality of a generic staging table concept for EC Integration Services. The solution is intended for loading tag based data where there is a generic tag set that must be seen as a unit when loading data. An example of such data is a set of generic well test tags where one tag identifies the well number, one identifies the start time, while others identify the various measurements during the test.

The challenge in such cases is that when receiving a tag sample on one of the tags, you do not have the necessary information in order to write the data, e.g. in the example above if you receive a pressure sample, you do not know what well it belongs to or what time the well test started. All this is part of the primary key, so you cannot yet write the data.

Another example of a similar case is for loading analysis data for cargos. In this case there may also be a generic set of tags, one tag being the cargo number or another unique identifier.

One basic assumption in the outlined solution is that all samples belonging to one data set (one well test, one cargo, etc.) will be timestamped exactly the same in the source system. The timestamp will be the way to determine which data belongs together.

This document illustrates a generic product solution for loading timestamp grouped tag based data from generic tag sets into EC. It is intended for customer key personnel, TE support personnel, and anyone who needs to understand how this module works.

## General

### Abbreviations

EC	Energy Components
SCADA	Supervisory Control and Data Acquisition. Source system where metering data is collected.
TC	TagCollector

### Definitions

Database job	A scheduled event in a database which runs at specified intervals. Managed by DBMS_JOB in Oracle.
PI	Plant Information system. Commonly used SCADA system.
EC IS	EC Integration Services – the EC Data Transfer Framework.
Trigger update.	A piece of code stored in the database that executes when a certain event occurs, e.g. a row insert or update.
EC tables	Tables within the EC main schema.
Staging tables	Temporary tables in EC schema. These are used to hold temporary data to be loaded into the EC tables.

## Background

The following section describes the data capture of well test data in ExxonMobil. Although the EC IS Staging Table Extension is generic, the EM well test data is the background for implementing the solution.

### ExxonMobil Well Test Data Capture

Data Capture from PI to EC is performed using the EC IS component. This is based on mappings that define how data are read and written, as well as any data aggregation or calculations performed at runtime. In order to write a calculated value, EC IS needs to know the primary key values of the row it should write to. There is already a data class in the system for loading single record well test data to EC based on a primary key of well name and *daytime*. The straightforward way of capturing such data is to have one tag and one EC IS mapping for each metering point on each well. However, in order to reduce the number of tags, ExxonMobil has decided on using a set of generic well test tags that work for all wells, and having well number available as a separate tag. When a well test is complete a set of tag samples will be made available in the PI server. These typically include:

- well number
- start time
- stop time
- duration
- wh\_press
- wh\_temp
- ...

Each sample for a well test will be design have the same unique timestamp. Sample data for one well test may look like

this (6 selected tags):

Timestamp	Well	Start	End	Duration	BHPress	GrsVol
2002-12-31 15:23:12	K111	20021231 043200	20021231 105400	382	123	144

Table 1 - Well test tag format

This format means that EC IS will not know the well number or daytime when writing a value for a well test, and therefore the data cannot be captured directly into the well test tables.

## Solution Overview

The solution to the problem is to first capture the data into temporary staging tables using EC IS. The rows are identified by the timestamp of the samples, as well as an identifier of the type of temporary data (e.g. single well test data or cargo analysis data).

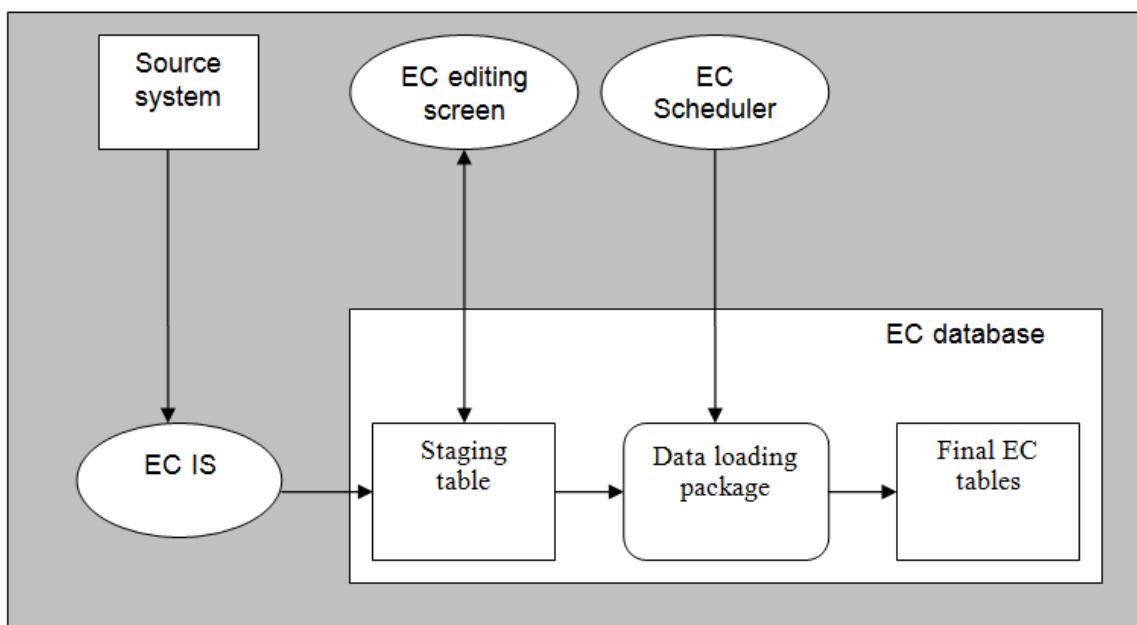


Figure 1 – Product solution overview

The solution outlined above is based on a two step data capture process.

1. From source system (e.g. PI) to EC staging tables using EC Integration Services.
2. From EC staging tables to final EC tables, through a data loading package.

Note that the logic in the data loading package will need to be implemented in each specific case. This is necessary in order to support various tag formats and target tables, making this a generic solution.

An EC screen will be made available for viewing and editing the data in the staging table.

## Database Items

### Item list

The following table lists all items in the database related to the EC IS Staging Table Extension.

Itemtype	Name	Purpose
table	tag_event_status	Temporary staging table to hold data captured from source system.
class	tag_event_status_1	Data class staging type 1.
class	tag_event_status_2	Data class staging type 2.
class	tag_event_status_3	Data class staging type 3.
class	dt_tag_event_status	Table class for configuring tag mappings through Maintain Mappings screen.
package	ecdp_tag_event_status	Main package being executed in stage two of the data capture process. Contains logic to loop over each record that is not already transferred, as well as setting row transfer status correctly. For each record, it will call the user exit package ue_tag_event_status, which will contain logic for transferring the record to final EC tables.
package	ue_tag_event_status	User exit package will need to be implemented for transferring data to the final EC tables. The package will handle one row at a time and will be called by ecdp_tag_event_status.

*Table 2 - Database items*

There is only one table available, and given that it is generic, it will only have generic columns available. The following table lists the table definition for `tag_event_status`.

Name	Type	Nullable	Description
daytime	date	N	Timestamp corresponding to timestamp of tag sample
staging_type	varchar2(32)	N	Staging type
value_1	number	Y	Generic number columns
...	...	...	
value_50	number	Y	
text_1	varchar2(16)	Y	Generic text columns of various lengths
...	...	...	
text_20	varchar2(4000)	Y	
date_1	date	Y	Generic date columns
...	...	...	
date_10	date	Y	
record_status	varchar2(1)	Y	
			EC required columns
created_by	varchar2(30)		
created_date	date		
last_updated_by	varchar2(30)	Y	
last_updated_date	date	Y	
rev_no	number	Y	
rev_text	varchar2(240)	Y	

Table 3 – Table definition – tag\_event\_data

In order to load data to tag\_event\_status, data classes must be created on top of the table. In the product these will only contain the basic required attributes, such as daytime and staging type. Additional attributes for holding primary key values (well codes etc), as well as attributes for the various data items need to be created in each project.

The following table lists the data class TAG\_EVENT\_STATUS\_1 as configured in the product.

Class name TAG_EVENT_STATUS_1	
Attribute name	Column name
daytime	daytime
object_id	<added by view generator> This refers to a dummy object (each data class must map to an object)
object_code	<added by view generator> This refers to a dummy object (each data class must map to an object)
staging_type	staging_type

Table 4 – Product class definition – tag\_event\_status\_1

The following table illustrates how the class might be configured for single well test data.

Class name TAG_EVENT_STATUS_1	
Attribute name	Column name
daytime	daytime
object_id	<added by view generator> This refers to a dummy object (each data class must map to an object)
object_code	<added by view generator> This refers to a dummy object (each data class must map to an object)
staging_type	staging_type
well_code	text_1 (well code)
start_time	text_2 (start of well test from a string tag, format yyyyymmdd hh24miss – used by EM)
end_time	text_3 (end of well test from a string tag, format yyyyymmdd hh24miss – used by EM)
avg_wh_press	value_1
avg_wh_temp	value_2
avg_bh_press	value_3
avg_bh_temp	value_4
avg_gas_rate	value_5
avg_oil_rate	value_6
avg_water_rate	value_7
...	...

Table 5 – Customer modified class definition – tag\_event\_data\_1

## Data Capture Stage 1

Stage 1 of the data capture is moving the data from the source system to the staging table *tag\_event\_status*. There is only one staging table available, which must be able to hold several types of staging data (e.g. single well tests and commingled well tests), and thus part of the primary key of this table will be the "staging type", corresponding to class name on top of tag\_event\_data. Available options will be configured as EC codes. The CODE should correspond to a class name, but CODE\_TEXT, may be changed to a more descriptive text. Unused staging types can be disabled.

In the example below, two different staging types are used, while the third is disabled.

CODE	CODE_TEXT (modified by user)	ENABLED
TAG_EVENT_DATA_1	Single Well Test	X
TAG_EVENT_DATA_2	Commingled Well Test	X
TAG_EVENT_DATA_3	TAG_EVENT_DATA_3	

Table 6 – EC Code type STAGING\_TYPE

## Creating tag mappings

For creating tag mappings, the end user will go to Maintain Mappings screen and select class name TAG\_EVENT\_STATUS. When creating the mapping, the Staging Type must be selected. Based on the staging type, the appropriate attribute list for that staging type will appear.

Staging type	Attribute	Source Id	Tag Id	Template code	etc.
Single Well Test	START_TIME	PI	myTag1	EVENT_DATA_COLLECTION	
Single Well Test	END_TIME	PI	myTag2	EVENT_DATA_COLLECTION	
Single Well Test	AVG_WH_PRESS	PI	myTag3	EVENT_DATA_COLLECTION	
Single Well Test	AVG_WH_TEMP	PI	myTag4	EVENT_DATA_COLLECTION	
Etc					

Table 7 – Example tag mappings – tag\_event\_data\_single\_wt

## Data Capture Stage 2

This section describes the process of transferring data from the temporary staging table to the final EC tables.

### Data loading package

In order to support having a generic staging table concept, the insert logic is moved into a data loading package, the content being implemented specifically for each customer.

Method	Description
ecdp_tag_event_status.transfer	Load all rows not already transferred. Calls ue_tag_event_status.transfer(...) for each record. This package will be fully implemented as part of the product.
ue_tag_event_status.transfer (daytime, staging_type)	Load specified row to final EC tables. The shell of the package is delivered as part of the product, but the logic inside the package is customer specific.

Table 8 - Data loading package

A job will be scheduled in the EC Scheduler to call ecdp\_tag\_event\_status.transfer with regular intervals.

The ue\_tag\_event\_status.transfer procedure will return one of the following values:

Return value	Description
Y	Row was successfully transferred.
N	Row could not be transferred (e.g. due to missing data).
E	An error occurred during validation or transfer of record. An error message is logged in the msg_log column of the record in tag_event_status.

Table 9 - Data loading package

### Data conversion

If data must be converted between staging tables and EC tables, this will be done by the data loading package. The most appropriate solution must be picked in each case.

### Synchronization process

When synchronizing data from tag\_event\_status to the EC tables, the data loading package needs to know which rows to transfer. This is accomplished by the additional column *in\_sync*.

The *in\_sync* column in the tag\_event\_status table is used to keep track of which rows have been updated in the EC

tables. It is also used for labelling rows that caused problems during transfer, e.g. because of invalid date format, or a non-existing well.

The following table lists possible values for the *in\_sync* column.

Value	Description
Y	All data in this row is synchronized in the EC tables.
N	This row is not synchronized with the EC tables. It will be synchronized at next execution of the well test transfer package.
E	An error occurred when attempting to synchronize this row. It will not be retried until new values are written to this row.

Table 10 – *In\_sync* values

The *in\_sync* column is updated in the following cases:

Event	Description	Updated by	New value
Row update	Someone writes to the row, e.g. a new value is captured by EC IS.	Trigger	N
Validation error	The data loading package detects an error, such as invalid date format, or a non-existing well.	ecdp_tag_event_status	E
Transfer error	The data loading package successfully validates data, but fails to transfer a row due to an unknown error.	ecdp_tag_event_status	E
Successful synchronization	The data loading package successfully transfers data from the row into the corresponding row in the EC well test tables.	ecdp_tag_event_status	Y

Table 11 – *In\_sync* update events

### Overwrite conditions

In many cases, it is necessary to protect existing records based on status of the record, e.g. only overwrite provisional data, or only write to data not modified by a user. Such logic will be implemented in *ue\_tag\_event\_status*.

### Setting of record status

If it is necessary to write records with a specified record status other than provisional, this logic will be implemented within the data loading package.

## BF –Tag Event Status

A business function will be developed in order to view and modify data in *tag\_event\_status* table.

Filter navigator will consist of:

- From date
- To date
- Staging type

Presentation of table contents will be based on data class definition for this staging type (for an example see Table 5 – Customer modified class definition – *tag\_event\_data\_1*). The resulting data window would look as follows:



Daytime	Code	Start Time	End Time	Avg Wh Press	Avg Wh Temp	Avg Bh Press	...	In Sync	Log Message
2007-02-03 12:43:02	W01	20070202 100300	20070202 110500	100	200	300		Y	
2007-02-04 12:32:12	W01	20070203 100000	20070203 120300	150	250	350		N	
2007-02-06 19:02:11	W02	20070206 170000	20070206 180000	300	400	500		E	Well code does not exist
Etc.									

# EC View Generator and Object Class Model

## Introduction

This section introduces the Energy Components (EC) Data Services View Generator – what it is, what it does and how it relates to the EC Object Class Model. The fundamentals of the EC Object Class Model are also covered.

If the reader is unfamiliar with the EC Object Class Model, it is recommended that a study is made of Sections 2, 6 and 7 of this document before reading the remaining sections dealing with the View Generator.

## Flexibility

The traditional Entity-Relation design approach to modelling data tends to require that everything about those data is known prior to modelling a concept, and once modelled, it becomes very difficult to change. Experience in the Energy Components domain shows that a more generic and flexible modelling approach enables us to better reflect our business knowledge as we learn more and as the business changes over time. We have seen that the need for data model changes continues to be high. We have experienced structures that we thought would be stable, turn out not to be valid anymore, e.g. that a well is part of a facility, whereas in reality a well could switch facility.

In light of these insights, our approach is to move away from explicit structures to more generic data structures, termed the EC Object Class Model.

## Abstraction

Currently, database tables are exposed directly to the various application parts. This makes it very difficult to build in common functionality to be shared by all parts of the application, e.g. EC provides a mechanism for creating an audit trail (the journal tables) by using database triggers. This is an example of functionality which is "hidden" from the application developer. In the same way, there are many areas where such functionality can be offered from the database, fully transparent to the application. In this way we can simplify application development and ensure consistency, independent of front-end technology.

The approach is to offer, from the database, an access layer above the actual tables. This layer is referred to as EC Data Services. The definition of this access layer follows the object class definition (it is generated from the object class definition). The approach is to have the object classes (and hence the access layer) grouped according to product context, referred to as an EC AppSpace, e.g. EC Production could represent one AppSpace.

The access layer takes the form of a set of views, generated automatically from the object class definitions.

## Generic components

Experience has shown that parts of the EC application can be developed in a generic way so that configuration parameters control system behaviour. Examples of this are the allocation engine, configuration application, visual configurations, formula editors, generic check/validation rules etc. When developing generic components we need to have a more generic way to access data as we have learned that using natural keys (the normal approach in ER / Relational databases), severely prevents us from making good generic code. The approach is to define a generic access layer, the GenAppSpace (which cannot "see" any of the other AppSpaces). As part of this approach, all objects in EC shall have a unique global identifier.

## The EC Data Services Object Class Model

This section sets out the fundamentals of the Object Class Model, in order that these can be referred to in the subsequent description of how the View Generator works.

The principle elements of interest are Classes, Objects and Attributes.

- A Class is a high level definition of something, but does not include any data referring to an actual example of that Class, e.g. a car could be defined as a Class.
- Attributes of a Class are the characteristics or properties of the Class, e.g. make, model, engine size, colour, number of seats and body style could all be attributes of the Car Class.
- Classes, together with their attributes, can be thought of as templates which provide a complete description of an item in isolation (relationships, dependencies etc. are described later) and which can give rise to many completed copies.
- An Object is an instance of a Class, i.e. a specific version of the Class with instance-specific data associated with it. A Saab 95 could be an Object of the Car Class, as could an Audi A6. Each of these instances, or Objects, would have all the corresponding attributes of the Car Class, but hold these for each vehicle, e.g. the Saab may be silver, and the Audi black.
- In an Energy Components implementation, typical Classes would include Wells, Fields, Pipelines, Streams, Tanks, Nodes and Terminals.

Part of the power of the Object Class Model comes from the way in which new classes and attributes can be defined without having to alter the physical structure of the database. This approach requires that, from a user's perspective, data are held in virtual tables. Although the virtual tables are linked to physical tables, this is transparent to the user, but of prime importance to the developer responsible for configuring the classes.

Classes and their attributes are defined in the Object Data Model as metadata held in a series of tables. These tables also hold information on where the data are actually stored, i.e. mapping the virtual to the real. Similarly, there are metadata entries for defining relationships between objects as well as dependencies. The tables used are as follows.

Table	Description
CLASS	What is the class called?
CLASS_ATTRIBUTE	What attributes does it have?
CLASS_ATTR_DB_MAPPING	Where do these attributes come from, e.g. table column (& name) or database function (& package name)?
CLASS_ATTR_PRESENTATION	How should these attributes be displayed, e.g. what units are to be used?
CLASS_DB_MAPPING	Where are the instances of the class, i.e. where are the objects stored, e.g. which table
CLASS_RELATION	Relationships between two object classes can be defined here
CLASS_REL_DB_MAPPING	Where do these relationships come from, e.g. table column (& name) or database function (& package name)?
CLASS_DEPENDENCY	Records which classes implement which interfaces, or which data class that a report class will be reporting on.
CLASS_TRIGGER_ACTION	Class specific code, will be merged into the generated Instead of trigger BEFORE or AFTER Insert, update, delete on the class view

The instances, or objects, of the classes defined above, are stored in separate tables, as are the data associated with objects. Example of tables used for WELL class are as follows.

Table	Description
WELL	Holds the details of each instance(well object) of a well class. (Holds details that are common for all versions of the well object.)
WELL_VERSION	Holds the versionable details of each attribute for each instance of a well class.
PWEL_DAY_STATUS	Holds the data associated with an instance of a well class.

Please see Section 6 for a relationship diagram for the Object Class Model. See also Section 7 for detailed descriptions and specifications of all these tables, including valid entries in columns.

One consequence of this approach to modelling data is that it is very difficult for a developer to relate directly to the database, e.g. writing an SQL query to retrieve information. This is where the advantage of the automatic generation of database views is apparent.

### The Need for a View Generator

As discussed earlier, the Object Class Model requires that there is a set of views provided at a higher level in order that interaction with data is possible. These views are, in effect, virtual tables, hence the need for the views to have

associated functionality that allow operations to be performed on the virtual tables in the same way that inserts, updates and deletes are carried out against 'real' tables. Triggers, also generated by the View Generator, provide this functionality.

All access to underlying data in the database is through these views. The View Generator works by reading object class model information from the metadata tables and creating views and triggers from these data according to a set of rules. The views and triggers automatically include the required mappings from the virtual tables to the actual underlying tables.

### **View Types**

Object views are used to manage Objects and their attributes, i.e. create new instances of Classes defined by metadata, update values, e.g. modify the period of operation of an Object (and thereby 'delete' an Object if necessary by setting its end date equal to its start date) and for selecting Object records from the database.

Data views are used to manage the data associated with Objects and allow for the insertion, modification, deletion and selection of data.

Interface views are built on top of object views and provide a mechanism for connecting two classes for the selection of data from each of these classes from a single, common, view.

Table views are simpler than Object, Data and Interface views because it is usually a 1:1 mapping to the underlying table or view, but with the possibility of defining virtual attributes.

Reporting views are generated foreach object, data, table, interface and report class.

Reporting views can be divided into four different types:

- Reporting views for object and data classes
- Reporting views for table classes
- Reporting views for interface classes
- Reporting views for report classes

### **Report Class and View**

The REPORT class is a new class introduced to support customer specified report views. This document emphasizes on the creating of a report views based on a REPORT class.

Report class will have one Object or Interface class as the owner and linked with many data classes. If more than one data class is included, they will probably share attributes having the same name. The report CLASS will be responsible to handle this issue.

Report views are generated for each type of class: OBJECT, INTERFACE, DATA, TABLE and REPORT. The report builder will generate technically different view for each type of class.

### **Object-Oriented Data Storage**

It is important to state that while many of the terms used in this document may be shared with Object-Oriented approaches to design, the EC Object Class Model is not an attempt to implement an Object-Oriented database. Some object-oriented concepts have been partially implemented, while others are missing, e.g. polymorphism and inheritance.

### **The View Generator**

There are two key aspects to the View Generator. The first is the correct generation of the required views and triggers in accordance with the metadata supplied. The second is the correct behaviour of those views. View functionality is implemented in triggers associated with each view where appropriate.

### **Generating views & triggers**

This section illustrates through description and example how the View Generator turns metadata into views & triggers. Not all Classes have triggers, so the following table lists the permutations of what is generated.

	<b>Generated</b>	
<b>Class</b>	<b>View</b>	<b>Trigger</b>
Object	Yes	Yes
Interface	Yes	Yes
Data	Yes	Yes
Table	Yes	Yes
Report	Yes	No

**Naming of items**

In the following, xxx is the class name

- Object views are called OV\_xxx, e.g. OV\_COMPANY.
- Interface views are called IV\_xxx, e.g IV\_FACILITY
- Data views are called DV\_xxx, e.g. DV\_COMP\_CONT\_YR\_SALES.
- Table views are called TV\_xxx, e.g. TV\_GROUP\_MODEL
- Triggers are called IUD\_xxx, e.g. IUD\_COMPANY.

For reports, the naming shall be like following:

- Report views for Objects or interfaces views are called RV\_xxx e.g. RV\_STREAM
- Report views for Data views are called RV\_xxx e.g. RV\_PWEL\_SUB\_DAY\_STATUS
- Report views for Table views are called RV\_xxx e.g. RV\_STREAM\_SET
- Report views for Report classes are called RV\_xxx e.g. RV\_REPORT\_STREAM

The xxx value for report refers to CLASS.CLASS\_NAME attribute.

**Running the View Generator**

Calling the stored procedure EcDp\_GenClassCode.BuildViewLayer runs the View Generator for all classes defined in Class-table. By default, the generator will write its output to the database and if there are any problems encountered during the generation process, details will be written to the T\_TEMPTEXT table. It is also possible to get the generator to spool the generated code entirely to the T\_TEMPTEXT table instead of the database; calling EcDp\_GenClassCode.BuildViewLayer('SCRIPT') will initiate this.

To build the view for a single class, run the BuildViewLayer with the class name as an argument.

E.g EcDp\_GenClassCode.BuildViewLayer('CREATE','WELL') will generate view and trigger for the well class.

To build both the view and the report view for a single class, run BuildView with the class name as an argument. Eg. EcDp\_GenClassCode.BuildView('WELL') will first generated OV\_Well, if succeeded it generates RV\_WELL.

The entire report layer can be built using stored procedure ECDP\_GenClassCode.BuildReportLayer. By default, the generator will write its output to the database. If there are any problems encountered during the generation process, details will be written to the T\_TEMPTEXT table. The generator may spool the generated code entirely to the T\_TEMPTEXT table by calling the procedure ECDP\_GenClassCode.BuildReportLayer('SCRIPT', null).

The report layer procedure generates report system for all object types if the second argument for the procedure is left null. Specifying object class for it makes the procedure generate report views for the class based on its definition.

When checking T\_TEMPTEXT for generation errors, it is important to limit the selection to messages related to the View Generator, since this table is also used to record other information, e.g. the generation of EC packages. The following code will select the messages of interest from T\_TEMPTEXT.

1. select text from t\_temptext
2. where id in('GENCODEERROR','GENCODEWARING')
3. order by line\_number;

When the view generator tries to compile a view or a trigger, it will first try to compile it with a dummy name starting with XX. This is done to avoid overwriting working views with invalid versions. Only if the XX version compiles successfully will the real View/trigger be generated.

It is therefore a good idea to look for invalid objects starting with XX after a build of the view layer. This is usually the fastest way to detect and correct errors in the class configuration. Note ! Only configuration that is in a good enough

shape to be "created with compilation error" will show up as XX-objects, so looking through t\_tempText is still useful.

In release 8.0, we also extend the BuildViewlayer to recompile objects that dependent on other generated objects, this is not 100% foolproof, it will do max 4 iterations, so if there are dependencies that requires more recompilations, this has to be handled manually.

## Generating an Object View

The following are the "rules" by which metadata are converted into object views. See Section 7.1 for further information about all Object Class types.

The CLASS\_DB\_MAPPING specify where objects are stored/mapped.

No.	Description
1	The view will be called OV_xxx, where xxx is the CLASS_NAME entry in table CLASS
2	The columns of the view will be the non-disabled (DISABLED_IND column) ATTRIBUTE_NAME entries in table CLASS_ATTRIBUTE, will be marked as NOT NULL if IS_MANDATORY is set to 'Y' and will have the data types defined in the DATA_TYPE column
3	The following columns are always added by default and should therefore not be defined in CLASS_ATTRIBUTE: CLASS_NAME, RECORD_STATUS, CREATED_BY, CREATED_DATE, LAST_UPDATED_BY, LAST_UPDATED_DATE, REV_NO, REV_TEXT All object classes must at least have the following attributes defined in CLASS_ATTRIBUTE in order to be able to generate the object view: OBJECT_ID, OBJECT_START_DATE, OBJECT_END_DATE, CODE, DAYTIME, END_DATE and NAME
4	The view columns will be mapped to the physical locations defined in CLASS_ATTR_DB_MAPPING, as either column references or SQL extracts such as function calls
5	CLASS_DB_MAPPING is used to specify where objects data are stored. CLASS_DB_MAPPING.DB_OBJECT_NAME holds the table where the object is stored. CLASS_DB_MAPPING.DB_OBJECT_ATTRIBUTE holds the table where versionable object attributes are stored.
6	Non-disabled relationships defined in CLASS_RELATION are generated as follows. Each relationship gives rise to two lines of code in the view. The first line returns a column called 'xxx_ID', the second a column called 'xxx_CODE' where xxx is whatever is defined in CLASS_RELATION.ROLE_NAME for the TO_CLASS_NAME entry corresponding to the class name of interest, e.g. if the class is COMPANY then the record is read where CLASS_RELATION.TO_CLASS_NAME = COMPANY; in this case the corresponding ROLE_NAME entry is COUNTRY, so the columns defined in the view are COUNTRY_ID AND COUNTRY_CODE. The actual locations of the values returned by these columns are dictated by the entries in CLASS_REL_DB_MAPPING columns DB_MAPPING_TYPE and DB_SQL_SYNTAX. For relationships to be defined correctly there must be entries in both CLASS_RELATION and CLASS_REL_DB_MAPPING. Please see Section 7.6 for further information on working with relationships

It should be noted that even though the View Generator is helpful in generating missing information automatically, it is a matter of good practice that it should not have to, i.e. the developer should seek to always provide a complete definition within the metadata. One reason for this is that other elements of Energy Components implementations also use this information.

### **Object View Example**

The following is an example of an object view generated from metadata. There are two ways to look at the view, by describing it in SQL\*Plus and by examining the source code. Both methods are used to illustrate the points of interest.

The example used is for a class called GENERATOR.

#### Metadata

The following are the metadata used in this example. In order that these can be shown legibly, not all of the columns for each table are shown. Those columns not shown are related to versioning and do not affect the generation of the views from the underlying data. Due to page width limitations, some tables have been transposed.

Table : CLASS

CLASS_NAME	SUPER_CLASS	CLASS_TYPE	APP_SPACE_CODE	TIME_SCOPE_CODE
GENERATOR	null	OBJECT	EC_PROD	VERSIONED

Table : CLASS\_ATTRIBUTE

CLASS_NAME	ATTRIBUTE_NAME	IS_KEY	IS_MANDATORY	CONTEXT_CODE	DATA_TYPE	PRECISION
GENERATOR	OBJECT_ID	Y	Y	EC_PROD	STRING	NULL
GENERATOR	CODE	N	Y	EC_PROD	STRING	NULL
GENERATOR	NAME	N	Y	EC_PROD	STRING	NULL
GENERATOR	OBJECT_START_DATE	N	Y	EC_PROD	DATE	NULL
GENERATOR	OBJECT_END_DATE	N	N	EC_PROD	DATE	NULL
GENERATOR	DAYTIME	N	Y	EC_PROD	DATE	NULL
GENERATOR	END_DATE	N	N	EC_PROD	DATE	NULL
GENERATOR	SCREEN_STATUS	N	N	EC_PROD	STRING	NULL

Table : CLASS\_ATTR\_DB\_MAPPING

CLASS_NAME	ATTRIBUTE_NAME	DB_MAPPING_TYPE	DB_SQL_SYNTAX
GENERATOR	OBJECT_ID	COLUMN	OBJECT_ID
GENERATOR	CODE	COLUMN	OBJECT_CODE
GENERATOR	NAME	ATTRIBUTE	NAME
GENERATOR	OBJECT_START_DATE	COLUMN	START_DATE
GENERATOR	OBJECT_END_DATE	COLUMN	END_DATE
GENERATOR	DAYTIME	ATTRIBUTE	DAYTIME
GENERATOR	END_DATE	ATTRIBUTE	END_DATE
GENERATOR	DESCRIPTION	COLUMN	DESCRIPTION
GENERATOR	SCREEN_STATUS	ATTRIBUTE	EQPM_STATUS_SCR

Table : CLASS\_DB\_MAPPING(Transposed)

CLASS_NAME	GENERATOR
DB_OBJECT_TYPE	TABLE
DB_OBJECT_OWNER	ECKERNEL_81
DB_OBJECT_NAME	EQUIPMENT
DB_OBJECT_ATTRIBUTE	EQPM_VERSION
DB_WHERE_CONDITION	o.CLASS_NAME='GENERATOR'

Table : CLASS\_RELATION (Transposed)

FROM_CLASS_NAME	FCTY_CLASS_1
TO_CLASS_NAME	GENERATOR
ROLE_NAME	OP_FCTY_1
IS_MANDATORY	N
IS_BIDIRECTIONAL	N
CONTEXT_CODE	EC_PROD
MULTIPLICITY	1:N
GROUP_TYPE	operational
REPORT_ONLY_IND	N
IS_DISABLED	N

Generator has an operational group connection to facility class 1. If CLASS\_RELATION.GROUP\_TYPE contains a value it indicates that the relation is a group relation.

Table : CLASS\_REL\_DB\_MAPPING

FROM_CLASS_NAME	TO_CLASS_NAME	ROLE_NAME	DB_MAPPING_TYPE	DB_SQL_SYNTAX
FCTY_CLASS_1	GENERATOR	OP_FCTY_1	ATTRIBUTE	OP_FCTY_CLASS_1_ID

#### Description

The following is the output of a DESC command applied to OV\_GENERATOR

```

SQL> descr ov_generator;
Name Type Nullable Default Comments
-----
1 CLASS_NAME CHAR(9) Y
2 OBJECT_ID VARCHAR2(32)
3 CODE VARCHAR2(32)
4 NAME VARCHAR2(240) Y
5 OBJECT_START_DATE DATE Y
6 OBJECT_END_DATE DATE Y
7 DAYTIME DATE
8 END_DATE DATE Y
9 SCREEN_STATUS VARCHAR2(1) Y
10 DESCRIPTION VARCHAR2(240) Y
11 OP_AREA_CODE VARCHAR2(32) Y
12 OP_AREA_ID VARCHAR2(32) Y
13 OP_FCTY_1_CODE VARCHAR2(32) Y
14 OP_FCTY_1_ID VARCHAR2(32) Y
15 OP_PRODUCTIONUNIT_CODE VARCHAR2(32) Y
16 OP_PRODUCTIONUNIT_ID VARCHAR2(32) Y
17 RECORD_STATUS VARCHAR2(1) Y
18 CREATED_BY VARCHAR2(30)
19 CREATED_DATE DATE
20 LAST_UPDATED_BY VARCHAR2(30) Y
21 LAST_UPDATED_DATE DATE Y
22 REV_NO VARCHAR2(81) Y
23 REV_TEXT VARCHAR2(240) Y

```

### Points of interest

The NOT NULL characteristic of columns shown as a result of describing a view comes from the underlying table structure. Even if CLASS\_NAME has "soft" constraints ensuring that it cannot be NULL, this is not reflected when the view is described in Oracle. To find mandatory columns use the utility EcDesc, described in Section 4.1.

All levels in the group hierarchy are being added as columns in the generated view.

The GENERATOR has the following operational group connection:

Production Unit\Area\Fcty\_Class\_1\Generator

The columns shown in line 18 – 23 are therefore included. Note that only the group level on parent level is editable thus production unit and area cannot be changed using the ov\_generator view.

The following refer to the definitions in CLASS\_ATTRIBUTE.

- There are many more columns than attributes defined because the generator automatically adds any standard columns not included in the class definition.
- The data types in the columns are as per those in the table.

### Source code

The following is the source code generated by the View Generator from the example metadata.

```

1 CREATE OR REPLACE VIEW OV_GENERATOR AS
2 SELECT
3 -- Generated by EcDp_GenClassCode
4 'GENERATOR' AS CLASS_NAME
5 ,o.OBJECT_ID AS OBJECT_ID
6 ,o.OBJECT_CODE AS CODE
7 ,oa.NAME AS NAME
8 ,o.START_DATE AS OBJECT_START_DATE
9 ,o.END_DATE AS OBJECT_END_DATE
10 ,oa.DAYTIME AS DAYTIME
11 ,oa.END_DATE AS END_DATE
12 ,oa.EQPM_STATUS_SCR AS SCREEN_STATUS
13 ,o.DESCRIPTION AS DESCRIPTION
14 ,oa.OP_AREA_CODE AS OP_AREA_CODE
15 ,oa.OP_AREA_ID AS OP_AREA_ID
16 ,oa.OP_FCTY_CLASS_1_CODE AS OP_FCTY_1_CODE
17 ,oa.OP_FCTY_CLASS_1_ID AS OP_FCTY_1_ID
18 ,oa.OP_PU_CODE AS OP_PRODUCTIONUNIT_CODE
19 ,oa.OP_PU_ID AS OP_PRODUCTIONUNIT_ID
20 ,oa.record_status AS RECORD_STATUS
21 ,oa.created_by AS CREATED_BY
22 ,oa.created_date AS CREATED_DATE
23
,decode(sign(nvl(o.last_updated_date,o.created_date)-nvl(oa.last_updated_date,oa.created_date)),1,oa.last_updated_by,oa.last_updated_by) AS LAST_UPDATED_BY
24
,decode(sign(nvl(o.last_updated_date,o.created_date)-nvl(oa.last_updated_date,oa.created_date)),1,oa.last_updated_date,oa.last_updated_date) AS LAST_UPDATED_DATE
25 ,o.rev_no||'.'||oa.rev_no AS REV_NO
26
,decode(sign(nvl(o.last_updated_date,o.created_date)-nvl(oa.last_updated_date,oa.created_date)),1,o.rev_text,oa.rev_text) AS REV_TEXT
27 FROM EQPM_VERSION oa, EQUIPMENT o
28 WHERE oa.object_id = o.object_id
29 AND o.CLASS_NAME='GENERATOR'

```

#### Points of interest

Line(s)	Comment
4	Standard class name column added automatically
5-13	Columns defined as attributes in CLASS_ATTRIBUTE Database locations for these columns as per CLASS_ATTR_DB_MAPPING
16,17	Columns defined from the relationship described in CLASS_RELATION with GROUP_TYPE = 'operational'
14,15 18,19	"Read Only" columns added automatically since GENERATOR is operational connected to FCTY_CLASS_1. These columns are defined in CLASS_RELATION as part of the operational group connection.
20-26	Standard columns added automatically. Line 23,24,26 contain logic to determine whether the column should be read from Equipment or Equipment_Version. The table last updated is being used.
29	Where-clause defined in CLASS_DB_ATTRIBUTE.DB_WHERE_CONDITION.

### Generating an Interface View

The following are the "rules" by which metadata are converted into interface views. Since an interface is built upon other Object classes, it is not defined in quite the same way as them, e.g. entries are not required in the following tables: CLASS\_ATTR\_DB\_MAPPING, CLASS\_DB\_MAPPING, CLASS\_RELATION & CLASS\_REL\_DB\_MAPPING. Instead, the Interface is defined in tables CLASS and CLASS\_ATTRIBUTE, with the objects the interface is built on referenced in CLASS\_DEPENDENCY.

See Section 7.1 for further information about Interface Class types and Section 7.8 for further information on working with dependencies.

No.	Description
1	The view will be called IV_xxx, where xxx is the CLASS_NAME entry in table CLASS
2	One or more Object Classes must exist to provide the underlying information for the Interface
3	The attribute names in all classes associated with the Interface must be the same, otherwise they will not be matched, i.e. if the interface has an attribute called EXAMPLE_TEXT, all of the underlying Classes must have an attribute of the same name. Note, however, that while the attribute names must be the same, their locations need not be, i.e. the EXAMPLE_TEXT attribute may be mapped to WELL_VERSION.TEXT_1 in the first of the underlying Objects, but it may be mapped to WELL_VERSION.TEXT_11 for the second Object.
4	Only non-private attributes in the Interface view and in each underlying Object will be included as columns in the view

### Interface View Example

The following is an example of an interface view generated from metadata. There are two ways to look at the view, by describing it in SQL\*Plus and by examining the source code. Both methods are used to illustrate the points of interest. The example used is for a class called EQUIPMENT.

## Metadata

The following are the metadata used in this example. In order that these can be shown legibly, not all of the columns for each table are shown. Those columns not shown are related to versioning and do not affect the generation of the views from the underlying data. Due to page width limitations, some tables have been transposed.

Table : CLASS

CLASS_NAME	SUPER_CLASS	CLASS_TYPE	APP_SPACE_CODE	TIME_SCOPE_CODE	DESCRIPTION
EQUIPMENT	null	INTERFACE	EC_PROD	VERSIONED	null

Table : CLASS\_ATTRIBUTE

CLASS_NAME	ATTRIBUTE_NAME	IS_KEY	IS_MANDATOR_Y	DATA_TYPE	CONTEXT_CODE
EQUIPMENT	OBJECT_ID	Y	Y	STRING	EC_PROD
EQUIPMENT	CODE	N	Y	STRING	EC_PROD
EQUIPMENT	NAME	N	Y	STRING	EC_PROD
EQUIPMENT	OBJECT_STAR_T_DATE	N	Y	DATE	EC_PROD
EQUIPMENT	OBJECT_END_DATE	N	N	DATE	EC_PROD
EQUIPMENT	DAYTIME	N	Y	DATE	EC_PROD
EQUIPMENT	END_DATE	N	N	DATE	EC_PROD
EQUIPMENT	DESCRIPTION	N	N	STRING	EC_PROD
EQUIPMENT	SCREEN_STATUS	N	N	STRING	EC_PROD

Table : CLASS\_DEPENDENCY

PARENT_CLASS	CHILD_CLASS	DEPENDENCY_TYPE
EQUIPMENT	GENERATOR	IMPLEMENTS
EQUIPMENT	PUMP	IMPLEMENTS

## Description

The following is the output of a DESC command applied to IV\_EQUIPMENT.

```

SQL>descr iv_equipment;
Navn Null? Type
-----
1 CLASS_NAME VARCHAR2(15)
2 CODE VARCHAR2(100)
3 DAYTIME DATE
4 DESCRIPTION VARCHAR2(2000)
5 END_DATE DATE
6 NAME VARCHAR2(100)
7 OBJECT_END_DATE DATE
8 OBJECT_ID VARCHAR2(32)
9 OBJECT_START_DATE DATE
10 SCREEN_STATUS VARCHAR2(100)
11 RECORD_STATUS VARCHAR2(1)
12 CREATED_BY VARCHAR2(30)
13 CREATED_DATE DATE
14 LAST_UPDATED_BY VARCHAR2(30)
15 LAST_UPDATED_DATE DATE
16 REV_NO NUMBER
17 REV_TEXT VARCHAR2(240)

```

#### Points of interest

The NOT NULL characteristic of columns shown as a result of describing a view comes from the underlying table structure. Even if CLASS\_NAME has "soft" constraints ensuring that it cannot be NULL, this is not reflected when the view is described in Oracle. To find mandatory columns use the utility EcDesc, described in Section 4.1.

- Each of the underlying Objects has more attributes than are required by this view, but they are ignored in favor of those defined for the Interface in CLASS\_ATTRIBUTE.

For completeness, the descriptions of the underlying Objects are shown below.

```

SQL> descr ov_generator;
Name Type Nullable Default Comments
-----
1 CLASS_NAME CHAR(9) Y
2 OBJECT_ID VARCHAR2(32)
3 CODE VARCHAR2(32)
4 NAME VARCHAR2(240) Y
5 OBJECT_START_DATE DATE Y
6 OBJECT_END_DATE DATE Y
7 DAYTIME DATE
8 END_DATE DATE Y
9 SCREEN_STATUS VARCHAR2(1) Y
10 DESCRIPTION VARCHAR2(240) Y
11 OP_AREA_CODE VARCHAR2(32) Y
12 OP_AREA_ID VARCHAR2(32) Y
13 OP_FCTY_1_CODE VARCHAR2(32) Y
14 OP_FCTY_1_ID VARCHAR2(32) Y
15 OP_PRODUCTIONUNIT_CODE VARCHAR2(32) Y
16 OP_PRODUCTIONUNIT_ID VARCHAR2(32) Y
17 RECORD_STATUS VARCHAR2(1) Y
18 CREATED_BY VARCHAR2(30)
19 CREATED_DATE DATE
20 LAST_UPDATED_BY VARCHAR2(30) Y
21 LAST_UPDATED_DATE DATE Y
22 REV_NO VARCHAR2(81) Y
23 REV_TEXT VARCHAR2(240) Y

SQL> descr ov_pump;
Name Type Nullable Default Comments
-----
1 CLASS_NAME CHAR(4) Y
2 OBJECT_ID VARCHAR2(32)
3 CODE VARCHAR2(32)
4 NAME VARCHAR2(240) Y
5 OBJECT_START_DATE DATE Y
6 OBJECT_END_DATE DATE Y
7 DAYTIME DATE
8 END_DATE DATE Y
9 SCREEN_STATUS VARCHAR2(1) Y
10 DESCRIPTION VARCHAR2(240) Y
11 OP_AREA_CODE VARCHAR2(32) Y
12 OP_AREA_ID VARCHAR2(32) Y
13 OP_FCTY_1_CODE VARCHAR2(32) Y
14 OP_FCTY_1_ID VARCHAR2(32) Y
15 OP_PRODUCTIONUNIT_CODE VARCHAR2(32) Y
16 OP_PRODUCTIONUNIT_ID VARCHAR2(32) Y
17 DEF_FCTY_1_ID VARCHAR2(32) Y
18 DEF_FCTY_1_CODE VARCHAR2(4000) Y
19 RECORD_STATUS VARCHAR2(1) Y
20 CREATED_BY VARCHAR2(30)
21 CREATED_DATE DATE
22 LAST_UPDATED_BY VARCHAR2(30) Y
23 LAST_UPDATED_DATE DATE Y
24 REV_NO VARCHAR2(81) Y
25 REV_TEXT VARCHAR2(240) Y

```

**Source code**

The following is the source code generated by the View Generator from the example metadata. Line numbers have been added for reference.

```
1 CREATE OR REPLACE VIEW IV_EQUIPMENT
2 (class_name, code, daytime, description, end_date, name,
object_end_date, object_id, object_start_date, screen_status,
record_status, created_by, created_date, last_updated_by,
last_updated_date, rev_no, rev_text)
3 AS
4 - Generated by EcDp_GenClassCode
5 SELECT 'GENERATOR'
6 , OBJECT_ID
7 , CODE
8 , NAME
9 , OBJECT_START_DATE
10 , OBJECT_END_DATE
11 , DAYTIME
12 , END_DATE
13 , DESCRIPTION
14 , SCREEN_STATUS
15 , RECORD_STATUS
16 , CREATED_BY
17 , CREATED_DATE
18 , LAST_UPDATED_BY
19 , LAST_UPDATED_DATE
20 , REV_NO
21 , REV_TEXT
22 FROM OV_GENERATOR
23 UNION ALL
24 SELECT 'PUMP'
25 , OBJECT_ID
26 , CODE
27 , NAME
28 , OBJECT_START_DATE
29 , OBJECT_END_DATE
30 , DAYTIME
31 , END_DATE
32 , DESCRIPTION
33 , SCREEN_STATUS
34 , RECORD_STATUS
35 , CREATED_BY
36 , CREATED_DATE
37 , LAST_UPDATED_BY
38 , LAST_UPDATED_DATE
39 , REV_NO
40 , REV_TEXT
41 FROM OV_PUMP
```

**Points of interest**

<b>Line(s)</b>	<b>Comment</b>
6 - 14	Columns defined as attributes in CLASS_ATTRIBUTE, but which match the names of the attributes defined for the EQUIPMENT Class
15 – 21	Standard columns added automatically, but where the column names match those of the underlying Class
22	Underlying Class being referenced by the Interface view
25 – 33	Columns defined as attributes in CLASS_ATTRIBUTE, but which match the names of the attributes defined for the EQUIPMENT Class
34 – 40	Standard columns added automatically, but where the column names match those of the underlying Class
41	Underlying Class being referenced by the Interface view

### Generating a Data View

The following are the "rules" by which metadata are converted into Data views. See Section 7.1 for further information about Data Class types.

No.	Description
1	The view will be called DV_xxx, where xxx is the CLASS_NAME entry in table CLASS
2	Data Classes are usually be 'owned' by an Object Class or an Interface class, this is defined as an implicit relation in CLASS OWNER CLASS NAME. From release 8.0 it is allowed to have data classes without an owner object class.
3	The columns of the view will be the non-disabled (DISABLED_IND column) ATTRIBUTE_NAME entries in table CLASS_ATTRIBUTE, will be marked as NOT NULL if IS_MANDATORY is set to 'Y' and will have the data types defined in the DATA_TYPE column.
4	If there is only a CLASS_ATTRIBUTE.CLASS_NAME entry and the class is defined in CLASS_DB_MAPPING, then by default, the view will be generated with the following columns, even if they have not been explicitly defined: CLASS_NAME, OBJECT_ID, OBJECT_CODE, RECORD_STATUS, CREATED_BY, CREATED_DATE, LAST_UPDATED_BY, LAST_UPDATED_DATE, REV_NO, REV_TEXT
5	CLASS_DB_MAPPING is used to specify where attributes' data are stored, i.e. a table or a view. The combination of the columns DB_OBJECT_TYPE and DB_OBJECT_NAME define the storage. This forms the FROM part of the view query
6	The view's columns will then be mapped to the physical locations defined in CLASS_ATTR_DB_MAPPING, as either column references or function calls
7	Non-disabled relationships between data classes and object classes defined in CLASS_RELATION are generated as follows. Each relationship gives rise to two lines of code in the view. The first line returns a column called 'xxx_ID', the second a column called 'xxx_CODE' where xxx is whatever is defined in CLASS_RELATION.ROLE_NAME for the TO_CLASS_NAME entry corresponding to the class name of interest, e.g. if the data class is COMP_CONT_YR_SALES then the record is read where CLASS_RELATION.TO_CLASS_NAME = COMP_CONT_YR_SALES; in this case the corresponding ROLE_NAME entry is INDEX, so the columns defined in the view are INDEX_ID AND INDEX_CODE. The actual locations of the values returned by these columns are dictated by the entries in CLASS_REL_DB_MAPPING columns DB_MAPPING_TYPE and DB_SQL_SYNTAX. For relationships to be defined correctly there must be entries in both CLASS_RELATION and CLASS_REL_DB_MAPPING. Please see Section 7.6 for further information on working with relationships
8	From release 8.0 it is allowed to have data classes without daytime as a part of the primary key, if the class has relations to other classes, it needs a date in a code from object_id lookup. This has normally been daytime, but for classes without daytime it will use EcDp_object.getObjStartDate, to get the first object code from the referred object.

It should be noted that even though the View Generator is helpful in generating missing information automatically, it is a matter of good practice that it should not have to, i.e. the developer should seek to always provide a complete definition within the metadata. One reason for this is that other elements of Energy Components implementations also use this information.

### **Data View Example**

The following is an example of a Data view generated from metadata. There are two ways to look at the view, by describing it in SQL\*Plus and by examining the source code. Both methods are used to illustrate the points of interest. The example used is for a data class called PWEL\_DAY\_STATUS\_OW which is owned by object class WELL.

#### **Metadata**

The following are the metadata used in this example. In order that these can be shown legibly, not all of the columns for each table are shown. Due to page width limitations, some tables have been transposed.

**Table : CLASS** (Transposed)

PWEL_DAY_STATUS_OW
null
DATA
EC_PROD
DAY
WELL

**Table : CLASS\_ATTRIBUTE** (Transposed)

CLASS_NAME	PWEL_DAY_STATUS_OW	Ditto	ditto	ditto
ATTRIBUTE_NAME	OBJECT_ID	DAYTIME	AVG_BH_PRESS	AVG_BH_TEMP
IS_KEY	Y	Y	N	N
IS_MANDATORY	Y	Y	N	N
CONTEXT_CODE	EC_PROD	EC_PROD	EC_PROD	EC_PROD
DATA_TYPE	STRING	DATE	NUMBER	NUMBER
IDISABLED_IND	N	N	N	N

**Table : CLASS\_ATTR\_DB\_MAPPING**

CLASS_NAME	ATTRIBUTE_NAME	DB_MAPPING_TYPE	DB_SQL_SYNTAX
PWEL_DAY_STATUS_OW	OBJECT_ID	COLUMN	OBJECT_ID
PWEL_DAY_STATUS_OW	DAYTIME	COLUMN	DAYTIME
PWEL_DAY_STATUS_OW	AVG_BH_TEMP	COLUMN	AVG_BH_TEMP_ID
PWEL_DAY_STATUS_OW	AVG_BH_PRESS	COLUMN	AVG_BH_PRESS

**Table : CLASS\_DB\_MAPPING**

CLASS_NAME	DB_OBJECT_TYPE	DB_OBJECT_OWNER	DB_OBJECT_NAME
PWEL_DAY_STATUS_OW	TABLE	ECKERNEL_81	PWEL_DAY_STATUS

### Description

The following is the output of a DESC command applied to DV\_PWEL\_DAY\_STATUS\_OW.

```
SQL> desc DV_PWEL_DAY_STATUS_OW;
Name Type Nullable Default Comments
-----
1 CLASS_NAME CHAR(18) Y
2 OBJECT_ID VARCHAR2(32)
3 OBJECT_CODE VARCHAR2(4000) Y
4 DAYTIME DATE
5 AVG_BH_PRESS NUMBER Y
6 AVG_BH_TEMP NUMBER Y
7 RECORD_STATUS VARCHAR2(1) Y
8 CREATED_BY VARCHAR2(30)
9 CREATED_DATE DATE
10 LAST_UPDATED_BY VARCHAR2(30) Y
11 LAST_UPDATED_DATE DATE Y
12 REV_NO NUMBER Y
13 REV_TEXT VARCHAR2(240) Y
```

### Points of interest

The NOT NULL characteristic of columns shown as a result of describing a view comes from the underlying table structure. Even if CLASS\_NAME has "soft" constraints ensuring that it cannot be NULL, this is not reflected when the view is described. To find mandatory columns use the utility EcDesc, described in Section 4.1.

The main owner of a data class is not represented in CLASS\_RELATION but in the column OWNER\_CLASS\_NAME in table CLASS. It will always be represented by OBJECT\_ID and OBJECT\_CODE in the Data class view.

The following refer to the definitions in CLASS\_ATTRIBUTE.

- There are many more columns than attributes defined because the generator automatically adds any standard columns not listed.
- The data types in the columns are as per those in the table.

### Source code

The following is the source code generated by the View Generator from the example metadata. Line numbers have been added for reference.

```

1 CREATE OR REPLACE VIEW DV_PWEL_DAY_STATUS_OW AS
2 SELECT
3 -- Generated by EcDp_GenClassCode
4 'PWEL_DAY_STATUS_OW' AS CLASS_NAME
5 , object_id AS OBJECT_ID
6 , EC_WELL.object_code(object_id) AS OBJECT_CODE
7 , DAYTIME AS DAYTIME
8 , AVG_BH_PRESS AS AVG_BH_PRESS
9 , AVG_BH_TEMP AS AVG_BH_TEMP
10 , RECORD_STATUS AS RECORD_STATUS
11 , CREATED_BY AS CREATED_BY
12 , CREATED_DATE AS CREATED_DATE
13 , LAST_UPDATED_BY AS LAST_UPDATED_BY
14 , LAST_UPDATED_DATE AS LAST_UPDATED_DATE
15 , REV_NO AS REV_NO
16 , REV_TEXT AS REV_TEXT
17 FROM ECKERNEL_82.PWEL_DAY_STATUS
18 WHERE
19 EXISTS (
20 SELECT 1 FROM well_version o
21 WHERE o.OBJECT_ID = PWEL_DAY_STATUS.object_id
22 AND o.well_type = 'OB' and pwel_day_status.daytime >= o.daytime
23 AND pwel_day_status.daytime < Nvl(o.end_date,
pwel_day_status.daytime + 1))

```

#### Points of interest

Line(s)	Comment
4 & 6	Standard columns added automatically if not defined. Note that the location of the data is as defined in CLASS_ATTR_DB_MAPPING, i.e. OBJECT_ID is simply a column, whereas OBJECT_CODE is a call to a database function
7 - 9	Columns defined as attributes in CLASS_ATTRIBUTE Database locations for these columns as per CLASS_ATTR_DB_MAPPING.
10 - 16	Standard columns added automatically
17	The data location, as per CLASS_DB_MAPPING.DB_OBJECT_NAME
19 - 23	Where condition defined in CLASS_DB_MAPPING.DB WHERE CONDITION

#### Generating a Report View

While it is sufficient to generate report view for other types of classes, it is necessary to build a report class before generating a report view for particular group of interest. It could be done by other mean than straight to the database, but it is beyond the scope of this document to elaborate on that.

To build report view layer for a class, it is sufficient to call view generator function specific to building report view, EcDp\_GenClassCode.BuildReportLayer('SCRIPT', '<CLASS\_NAME>').

These are the rules by which object classes (OBJECT, INTERFACE, DATA, TABLE) are converted into a set of report

views. See section 7.1 for further information about all Object Class types.

No.	Description
1	The view will be called RV_xxx, where xxx is the CLASS_NAME entry in table CLASS
2	The report views depend on the class meta data. It is imperative that the referred views are valid prior to generation of the report views. This is mainly applicable for report views of data classes and report classes.
3	If the class type is OBJECT, the generated view consists of the version-table and object table(e.g WELL_VERSION and WELL) joined with SYSTEM_DAYS
4	If the class type is DATA, the generated view will usually consists of class attributes from the owner in addition to the data class attributes. This requires that the data class have a date attribute that is relevant to use in join with the owner class. Additional columns showing the unit conversion are also included in the view. Note that to get a join with system day the class must have TIME_SCOPE_CODE = 'EVENT_CONTINUES', and have an END_DATE attribute.
5	If the class type is TABLE, the generated view is only the copy of the table view (TV_xxx).
6	If the class type is REPORT, the generated view consists of the union of all linked report data class views.

It is worth to note that for report class several linked data classes are joined together to generate the report view. Therefore several issues arise. These data classes share attributes having same name. For instance, AVG\_TEMP can be an attribute for both STRM\_DAY\_STREAM\_GAS and STRM\_DAY\_STREAM\_OIL. The report view generated will list the temperature for both data classes.

The report class and all data classes that it reports on must have equal time scope code.

If a data class has a lower time scope code than the report class, the data class must be aggregated up to the same time scope as the report class.

CLASS	TIME_SCOPE_CODE
Class1	DAY
Class2	MTH
Class3	MTH
Class4	YEAR

Class1 has a lower time scope code than class2. (Class1 must be aggregated up to month.)

Class3 has a lower time scope code than class4. (Class3 must be aggregated up to year.)

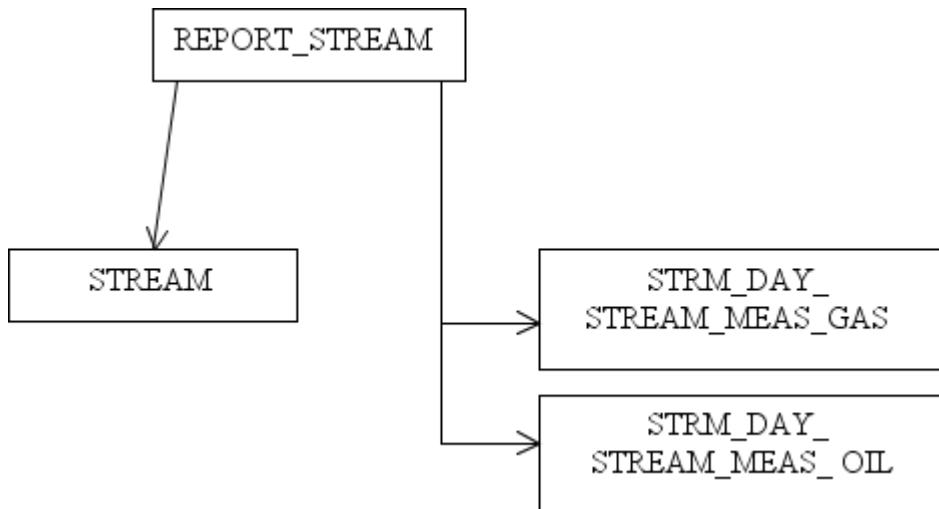
However, as of current release only classes with equal TIME\_SCOPE\_CODE will be supported.

### **Report View Example**

The table below lists columns that need special attention when generating a view for a report class.

TABLE	COLUMN	DESCRIPTION
CLASS	CLASS_TYPE	CLASS_TYPE must be set to REPORT.
	TIME_SCOPE_CODE	TIME_SCOPE_CODE is used to determine the time scope for the generated view.
	OWNER_CLASS_NAME	OWNER_CLASS_NAME must be set equal an OBJECT class name.
CLASS_DEPENDENCY	PARENT_CLASS	PARENT_CLASS must be set equal a REPORT class name.
	CHILD_CLASS	CHILD_CLASS must be set equal a DATA class name. NOTE: The owner of PARENT_CLASS and CHILD_CLASS must be the same.
	DEPENDENCY_TYPE	DEPENDENCY_TYPE must be set to 'REPORT_MEMBER'

The report class is used as base for the generated report view. It contains no attributes, and will only be holding references to other data classes.



The following is an example of a report view generated from metadata.

A report class, REPORT\_STREAM is defined. The report class owned by STREAM and it reports on data classes STRM\_DAY\_STREAM\_MEAS\_GAS and STRM\_DAY\_STREAM\_MEAS\_OIL. The time scope code is set to DAY.

### Types of Report Views

#### Report View: RV\_<REPORT>

Execute EcDp\_genclasscode.BuildReportLayer('CREATE','REPORT\_STREAM'), and the following view will be generated:

```

1 CREATE OR REPLACE VIEW RV_REPORT_STREAM
2 (class_name, owner_class_name, data_class_name, object_id, code,
  
```

```
name,
3 object_start_date, object_end_date, production_day, daytime,
stream_category,
4 stream_type, alloc_period, alloc_fixed, stream_phase,
gross_volume_method,
5 stream_meter_freq, net_volume_method, alloc_code, isvirtual,
bsw_vol_method,
6 grs_mass_method, net_mass_method, bsw_wt_method, std_density_method,
7 water_vol_method, diagram_layout_info, resv_blockFormation_id,
aga_ref_analysis,
8 from_node_id, from_node_code, to_node_id, to_node_code,
op_fcty_1_id,
9 op_fcty_1_code, specific_gravity_method, stream_name, sort_order,
grs_vol_gas_sm3,
10 grs_vol_gas_mscf, grs_vol_oil_sm3, grs_vol_oil_bbls, bs_w,
net_vol_oil_sm3,
11 net_vol_oil_bbls, record_status, created_by, created_date,
last_updated_by,
12 last_updated_date, rev_no, rev_text)
13 AS
14 ( -- Generated by EcDp_GenClassCode
15 SELECT
16 'REPORT_STREAM',
17 r.owner_class_name,
18 r.data_class_name,
19 r.object_id,
20 r.code,
21 r.name,
22 r.object_start_date,
23 r.object_end_date,
24 r.production_day,
25 r.daytime daytime,
26 r.stream_category,
27 r.stream_type,
28 r.alloc_period,
29 r.alloc_fixed,
30 r.stream_phase,
31 r.gross_volume_method,
32 r.stream_meter_freq,
33 r.net_volume_method,
34 r.alloc_code,
35 r.isvirtual,
36 r.bsw_vol_method,
37 r.grs_mass_method,
38 r.net_mass_method,
39 r.bsw_wt_method,
40 r.std_density_method,
41 r.water_vol_method,
42 r.diagram_layout_info,
43 r.resv_blockFormation_id,
44 r.agr_ref_analysis,
45 r.from_node_id,
46 r.from_node_code,
47 r.to_node_id,
48 r.to_node_code,
49 r.op_fcty_1_id,
50 r.op_fcty_1_code,
```

```
51 r.specific_gravity_method,
52 r.stream_name,
53 r.sort_order,
54 r.grs_vol_gas_sm3,
55 r.grs_vol_gas_mscf,
56 NULL,
57 NULL,
58 NULL,
59 NULL,
60 NULL,
61 r.record_status,
62 r.created_by,
63 r.created_date,
64 r.last_updated_by,
65 r.last_updated_date,
66 r.rev_no,
67 r.rev_text
68 FROM rv_strm_day_stream_meas_gas r
69 UNION ALL
70 SELECT
71 'REPORT_STREAM',
72 r.owner_class_name,
73 r.data_class_name,
74 r.object_id,
75 r.code,
76 r.name,
77 r.object_start_date,
78 r.object_end_date,
79 r.production_day,
80 r.daytime daytime,
81 r.stream_category,
82 r.stream_type,
83 r.alloc_period,
84 r.alloc_fixed,
85 r.stream_phase,
86 r.gross_volume_method,
87 r.stream_meter_freq,
88 r.net_volume_method,
89 r.alloc_code,
90 r.isvirtual,
91 r.bsw_vol_method,
92 r.grs_mass_method,
93 r.net_mass_method,
94 r.bsw_wt_method,
95 r.std_density_method,
96 r.water_vol_method,
97 r.diagram_layout_info,
98 r.resv_block_formation_id,
99 r.agr_ref_analysis,
100 r.from_node_id,
101 r.from_node_code,
102 r.to_node_id,
103 r.to_node_code,
104 r.op_fcty_1_id,
105 r.op_fcty_1_code,
106 r.specific_gravity_method,
107 r.stream_name,
```

```
108 r.sort_order,  
109 NULL,  
110 NULL,  
111 r.grs_vol_oil_sm3,  
112 r.grs_vol_oil_bbls,  
113 r.bs_w,  
114 r.net_vol_oil_sm3,  
115 r.net_vol_oil_bbls,  
116 r.record_status,  
117 r.created_by,  
118 r.created_date,  
119 r.last_updated_by,  
120 r.last_updated_date,
```

```

121 r.rev_no,
122 r.rev_text
123 FROM rv_strm_day_stream_meas_oil r)

```

**Points of interest**

Line(s)	Comment
01 - 12	Column definition. List a "union" of all columns defined for owner and all child classes.
56 - 60	Columns not applicable for the rv_strm_day_stream_meas_gas
109 – 110	Columns not applicable for the rv_strm_day_stream_meas_oil

**Report View: RV\_<OBJECT>**

The following view will be created when executing EcDp\_genclasscode.BuildReportLayer('CREATE','STREAM');

```

1 CREATE OR REPLACE VIEW RV_STREAM AS
2 SELECT
3   -- Generated by EcDp_GenClassCode
4   'STREAM' AS CLASS_NAME
5   ,o.object_id AS OBJECT_ID
6   ,o.object_code AS CODE
7   ,oa.name AS NAME
8   ,o.start_date AS OBJECT_START_DATE
9   ,o.end_date AS OBJECT_END_DATE
10  ,s.daytime AS PRODUCTION_DAY
11  ,oa.daytime AS DAYTIME
12  ,oa.end_date AS END_DATE
13  ,oa.stream_category AS STREAM_CATEGORY
14  ,oa.stream_type AS STREAM_TYPE
15  ,oa.alloc_period AS ALLOC_PERIOD
16  ,oa.alloc_fixed AS ALLOC_FIXED
17  ,oa.stream_phase AS STREAM_PHASE
18  ,oa.grs_vol_method AS GROSS_VOLUME_METHOD
19  ,oa.strm_meter_freq AS STREAM_METER_FREQ
20  ,oa.net_vol_method AS NET_VOLUME_METHOD
21  ,to_char('N') AS ISVIRTUAL
22  ,oa.bsw_vol_method AS BSW_VOL_METHOD
23  ,oa.grs_mass_method AS GRS_MASS_METHOD
24  ,oa.net_mass_method AS NET_MASS_METHOD
25  ,oa.bsw_wt_method AS BSW_WT_METHOD
26  ,oa.std_dens_method AS STD_DENSITY_METHOD
27  ,oa.water_vol_method AS WATER_VOL_METHOD
28  ,oa.diagram_layout_info AS DIAGRAM_LAYOUT_INFO
29  ,oa.agr_ref_analysis_id AS AGA_REF_ANALYSIS
30  ,oa.specific_gravity_method AS SPECIFIC_GRAVITY_METHOD
31  ,to_char('N') AS ALLOC_FIXED_GOR
32  ,to_char('N') AS ALLOC_FIXED_WC
33  ,oa.op_pu_id AS OP_PRODUCTIONUNIT_ID

```

```
34 ,oa.op_area_id AS OP_AREA_ID
35 ,oa.op_pu_code AS OP_PRODUCTIONUNIT_CODE
36 ,oa.op_area_code AS OP_AREA_CODE
37 ,oa.from_node_id AS FROM_NODE_ID
38 ,EcDp_Objects.GetObjCode(oa.FROM_NODE_ID) AS FROM_NODE_CODE
39 ,oa.to_node_id AS TO_NODE_ID
40 ,EcDp_Objects.GetObjCode(oa.TO_NODE_ID) AS TO_NODE_CODE
41 ,oa.op_fcty_class_1_id AS OP_FCTY_1_ID
42 ,oa.op_fcty_class_1_code AS OP_FCTY_1_CODE
43 ,oa.record_status AS RECORD_STATUS
44 ,oa.created_by AS CREATED_BY
45 ,oa.created_date AS CREATED_DATE
46
,decode(sign(nvl(o.last_updated_date,o.created_date)-nvl(oa.last_updated_date,oa.created_date)),1,o.last_updated_by,oa.last_updated_by) AS LAST_UPDATED_BY
47
,decode(sign(nvl(o.last_updated_date,o.created_date)-nvl(oa.last_updated_date,oa.created_date)),1,o.last_updated_date,oa.last_updated_date) AS LAST_UPDATED_DATE
48 ,o.rev_no||'.'||oa.rev_no AS REV_NO
49
,decode(sign(nvl(o.last_updated_date,o.created_date)-nvl(oa.last_updated_date,oa.created_date)),1,o.rev_text,oa.rev_text) AS REV_TEXT
50 FROM STRM_VERSION oa, STREAM o, system_days s
51 WHERE oa.object_id = o.object_id
```

```

52 AND s.daytime >= oa.daytime
53 AND (s.daytime < oa.end_date OR oa.end_date IS NULL)
54 AND (s.daytime < o.start_date OR o.end_date IS NULL)

```

### Points of interest

Line(s)	Comment
10	system_days.daytime is always renamed to production_day
04 – 09 11 – 49	Object view columns
21,22	Virtual attributes in ov-view are still virtual attributes in rv-views.
52-54	STRM_VERSION and STREAM are being joined with SYSTEM_DAYS.

### Report View: RV\_<DATA>

The following view will be created when executing EcDp\_genclasscode.BuildReportLayer('CREATE', 'PWEL\_SUB\_DAY\_STATUS'):

```

1 CREATE OR REPLACE VIEW RV_PWEL_SUB_DAY_STATUS AS
2 SELECT
3   -- Generated by EcDp_GenClassCode
4   'PWEL_SUB_DAY_STATUS' AS DATA_CLASS_NAME,
5   o.class_name AS OWNER_CLASS_NAME,
6   o.production_day AS PRODUCTION_DAY,
7   o.OBJECT_ID AS OBJECT_ID,
8   o.CODE AS CODE,
9   o.NAME AS NAME,
10  o.OBJECT_START_DATE AS OBJECT_START_DATE,
11  o.OBJECT_END_DATE AS OBJECT_END_DATE,
12  o.NODE_CLASS_NAME AS NODE_CLASS_NAME,
13  o.WELL_CLASS AS WELL_CLASS,
14  o.WELL_TYPE AS WELL_TYPE,
15  o.PROD_METHOD AS PROD_METHOD,
16  o.GAS_LIFT_METHOD AS GAS_LIFT_METHOD,
17  o.PT_RESDATA_CLASS AS PT_RESDATA_CLASS,
18  o.CALC_METHOD AS CALC_METHOD,
19  o.POTENTIAL_METHOD AS POTENTIAL_METHOD,
20  o.WELL_TEST_METHOD AS WELL_TEST_METHOD,
21  o.DESCRIPTION AS DESCRIPTION,
22  o.ALLOC_FIXED_GOR AS ALLOC_FIXED_GOR,
23  o.ALLOC_FIXED_WC AS ALLOC_FIXED_WC,
24  o.CALC_RULE_CODE AS CALC_RULE_CODE,
25  o.CALC_SEQ_NO AS CALC_SEQ_NO,
26  o.ALLOC_FLAG AS ALLOC_FLAG,
27  o.CAN_PROC_OIL AS CAN_PROC_OIL,
28  o.CAN_PROC_GAS AS CAN_PROC_GAS,
29  o.CAN_PROC_WAT AS CAN_PROC_WAT,
30  o.CAN_PROC_GASLIFT AS CAN_PROC_GASLIFT,

```

```

31 o.CAN_PROC_GASINJ AS CAN_PROC_GASINJ,
32 o.CAN_PROC_WATINJ AS CAN_PROC_WATINJ,
33 o.CAN_PROC_COND AS CAN_PROC_COND,
34 o.CAN_PROC_DILUENT AS CAN_PROC_DILUENT,
35 o.DIAGRAM_LAYOUT_INFO AS DIAGRAM_LAYOUT_INFO,
36 o.POTEN_2ND_VALUE AS POTEN_2ND_VALUE,
37 o.POTEN_3RD_VALUE AS POTEN_3RD_VALUE,
38 o.CHOCKE_UOM AS CHOCKE_UOM,
39 o.STD_OIL_DENSITY_METHOD AS STD_OIL_DENSITY_METHOD,
40 o.STD_GAS_DENSITY_METHOD AS STD_GAS_DENSITY_METHOD,
41 o.CALC_METHOD_MASS AS CALC_METHOD_MASS,
42 o.INSTRUMENTATION_TYPE AS INSTRUMENTATION_TYPE,
43 o.OP_NODE_ID AS OP_NODE_ID,
44 o.GP_NODE_ID AS GP_NODE_ID,
45 o.WP_NODE_ID AS WP_NODE_ID,
46 o.CP_NODE_ID AS CP_NODE_ID,
47 o.GL_NODE_ID AS GL_NODE_ID,
48 o.DL_NODE_ID AS DL_NODE_ID,
49 o.GI_NODE_ID AS GI_NODE_ID,
50 o.WI_NODE_ID AS WI_NODE_ID,
51 o.OP_PRODUCTIONUNIT_ID AS OP_PRODUCTIONUNIT_ID,
52 o.OP_AREA_ID AS OP_AREA_ID,
53 o.OP_PRODUCTIONUNIT_CODE AS OP_PRODUCTIONUNIT_CODE,
54 o.GEO_AREA_ID AS GEO_AREA_ID,
55 o.OP_AREA_CODE AS OP_AREA_CODE,
56 o.GEO_AREA_CODE AS GEO_AREA_CODE,
57 o.WELL_HOLE_ID AS WELL_HOLE_ID,
58 o.WELL_HOLE_CODE AS WELL_HOLE_CODE,
59 o.OP_FCTY_1_ID AS OP_FCTY_1_ID,
60 o.OP_FCTY_1_CODE AS OP_FCTY_1_CODE,
61 o.GEO_FIELD_ID AS GEO_FIELD_ID,
62 o.GEO_FIELD_CODE AS GEO_FIELD_CODE,
63 o.DEF_COMPRESSOR_ID AS DEF_COMPRESSOR_ID,
64 o.DEF_COMPRESSOR_CODE AS DEF_COMPRESSOR_CODE,
65 o.DEF_EQPM_OTHER_ID AS DEF_EQPM_OTHER_ID,
66 o.DEF_EQPM_OTHER_CODE AS DEF_EQPM_OTHER_CODE,
67 o.DEF_FLOWLINE_ID AS DEF_FLOWLINE_ID,
68 o.DEF_FLOWLINE_CODE AS DEF_FLOWLINE_CODE,
69 o.DEF_PUMP_ID AS DEF_PUMP_ID,
70 o.DEF_PUMP_CODE AS DEF_PUMP_CODE,
71 o.DEF_FCTY_1_ID AS DEF_FCTY_1_ID,
72 o.DEF_FCTY_1_CODE AS DEF_FCTY_1_CODE,
73 o.OP_WELL_HOOKUP_ID AS OP_WELL_HOOKUP_ID,
74 o.OP_WELL_HOOKUP_CODE AS OP_WELL_HOOKUP_CODE,
75 o.DEF_CTRL_SAFETY_SYSTEM_ID AS DEF_CTRL_SAFETY_SYSTEM_ID,
76 o.DEF_CTRL_SAFETY_SYSTEM_CODE AS DEF_CTRL_SAFETY_SYSTEM_CODE,
77 o.DEF.Utility_ID AS DEF.Utility_ID,
78 o.DEF.Utility_CODE AS DEF.Utility_CODE,
79 o.DEF_POWER_DISTRIBUTION_ID AS DEF_POWER_DISTRIBUTION_ID,
80 o.DEF_POWER_DISTRIBUTION_CODE AS DEF_POWER_DISTRIBUTION_CODE,
81 o.DEF_GAS_PROC_ID AS DEF_GAS_PROC_ID,
82 o.DEF_GAS_PROC_CODE AS DEF_GAS_PROC_CODE,
83 o.PROCESS_TRAIN_ID AS PROCESS_TRAIN_ID,
84 o.PROCESS_TRAIN_CODE AS PROCESS_TRAIN_CODE,
85 pwel_sub_day_status.DAYTIME AS DAYTIME,
86 pwel_sub_day_status.SUMMER_TIME AS SUMMER_TIME,
87 pwel_sub_day_status.ON_STREAM_HRS AS ON_STREAM_HRS_HRS,

```

```
88
EC_CTRL_SYSTEM_ATTRIBUTE.ATTRIBUTE_VALUE(PWEL_SUB_DAY_STATUS.daytime, 'WELL_SUB_DAY_HRS', '<=' ) AS ON_STREAM_HRS_SYS_HRS,
89 pwel_sub_day_status.AVG_CHOKE_SIZE AS AVG_CHOKE_SIZE,
90 pwel_sub_day_status.AVG_WH_TEMP AS AVG_WH_TEMP_C,
91
EcDp_Unit.convertValue(pwel_sub_day_status.AVG_WH_TEMP, 'C', 'F', o.production_day,NULL) AS AVG_WH_TEMP_F,
92 pwel_sub_day_status.AVG_WH_PRESS AS AVG_WH_PRESS_BARG,
93
EcDp_Unit.convertValue(pwel_sub_day_status.AVG_WH_PRESS, 'BARG', 'PSIG',
o.production_day,NULL) AS AVG_WH_PRESS_PSI,
94
EcDp_Unit.convertValue(pwel_sub_day_status.AVG_WH_PRESS, 'BARG', 'INH2O',
,o.production_day,NULL) AS AVG_WH_PRESS_INH2O,
95 pwel_sub_day_status.AVG_BH_PRESS AS AVG_BH_PRESS_BARG,
96
EcDp_Unit.convertValue(pwel_sub_day_status.AVG_BH_PRESS, 'BARG', 'PSIG',
o.production_day,NULL) AS AVG_BH_PRESS_PSI,
97
EcDp_Unit.convertValue(pwel_sub_day_status.AVG_BH_PRESS, 'BARG', 'INH2O',
,o.production_day,NULL) AS AVG_BH_PRESS_INH2O,
98 pwel_sub_day_status.RECORD_STATUS,
99 pwel_sub_day_status.CREATED_BY,
100 pwel_sub_day_status.CREATED_DATE,
101 pwel_sub_day_status.LAST_UPDATED_BY,
102 pwel_sub_day_status.LAST_UPDATED_DATE,
103 pwel_sub_day_status.REV_NO,
104 pwel_sub_day_status.REV_TEXT
105 FROM RV_WELL o,pwel_sub_day_status
106 WHERE o.object_id = pwel_sub_day_status.object_id
107 AND o.production_day = pwel_sub_day_status.PRODUCTION_DAY
```

```

108 AND
ec_well_version.well_type(pwel_sub_day_status.object_id,pwel_sub_day_status.daytime,'<=')
in ('OP','GP','CP','OPGI','GPI')

```

#### Points of interest

Line(s)	Comment
06	The column production_<TSC> will be named according to the time scope code for the data class. It is called production_day for all time scope code less than day or null
90 - 91	Example of how the unit conversion is performed. AVG_WH_TEMP is stored as C in the database, converted to F in line 91.
105	Need to join with the data class table and not the dv-view. This must be done since the dv-view can exclude some attributes (attributes that only should be included in reporting views).
107	If the data class contains an attribute with name production_day, this attribute will be used in the join statement (o.production_day = >).

#### Report View: RV\_<INTERFACE>

The following view will be generated when executing EcDp\_GenClassCode.BuildReportLayer('CREATE','FACILITY');

```
1 CREATE OR REPLACE VIEW RV_FACILITY
2 (class_name, production_day, code, daytime, end_date, name,
object_end_date,
3 object_id, object_start_date, prod_day_start, total_beds,
record_status, created_by,
4 created_date, last_updated_by, last_updated_date, rev_no, rev_text)
5 AS
6 SELECT -- Generated by EcDp_GenClassCode
7 'FCTY_CLASS_1'
8 ,PRODUCTION_DAY
9 ,CODE
10 ,DAYTIME
11 ,END_DATE
12 ,NAME
13 ,OBJECT_END_DATE
14 ,OBJECT_ID
15 ,OBJECT_START_DATE
16 ,PROD_DAY_START
17 ,TOTAL_BEDS
18 ,RECORD_STATUS
19 ,CREATED_BY
20 ,CREATED_DATE
21 ,LAST_UPDATED_BY
22 ,LAST_UPDATED_DATE
23 ,REV_NO
24 ,REV_TEXT
25 FROM RV_FCTY_CLASS_1
26 UNION ALL SELECT 'FCTY_CLASS_2'
27 ,PRODUCTION_DAY
28 ,CODE
29 ,DAYTIME
30 ,END_DATE
31 ,NAME
32 ,OBJECT_END_DATE
33 ,OBJECT_ID
34 ,OBJECT_START_DATE
35 ,PROD_DAY_START
36 ,TOTAL_BEDS
37 ,RECORD_STATUS
38 ,CREATED_BY
39 ,CREATED_DATE
40 ,LAST_UPDATED_BY
41 ,LAST_UPDATED_DATE
42 ,REV_NO
43 ,REV_TEXT
44 FROM RV_FCTY_CLASS_2
```

**Points of interest**

Line(s)	Comment
01	This view looks similar as IV_FACILITY except that it is using reporting views instead of OV-views.
25,44	RV_FCTY_CLASS_1 and RV_FCTY_CLASS_2 must have been generated prior generating RV_FACILITY

#### Report View: RV\_<TABLE>

The following view will be created when executing EcDp\_genclasscode.BuildReportLayer('CREATE', 'STREAM\_SET');

```

1 CREATE OR REPLACE VIEW RV_STREAM_SET AS
2 SELECT
3   -- Generated by EcDp_GenClassCode
4   o.CODE
5   , o.NAME
6 FROM tv_STREAM_SET o

```

#### Generating an Object Trigger

The following are the "rules" by which metadata are converted into Object triggers. Every Object view should have an associated instead-of trigger unless it is being defined as a "read only" class. The trigger provides the functionality which allows the insertion and updating of data in the virtual table represented by the view. Note that it is not possible to delete Objects through the view, if this is required, the Object's end date should be set equal to its start date instead.

Trigger code is generated in blocks, or groups of lines, related to common topics such as Insert Check, Insert Relation, Create Object, Date Check, Update Relation, Update Attribute and Record Revision. To aid testing and debugging of triggers, block headings are written into the code to indicate the start and end point of each block. In general, in the following table the "rules" are considered in the context of the block of code they relate to. See Section 3.6.1.2 for an example of trigger source code.

No.	Description
1	The trigger will be called IUD_xxx, where xxx is the CLASS_NAME entry in table CLASS
2	The trigger will contain at least one variable for each non-disabled attribute defined in CLASS_ATTRIBUTE.ATTRIBUTE_NAME. For some attributes(e.g daytime) there will exist two variables, one holding the new value and one holding the old value. All variables are being initialised at the beginning of the trigger.
3	The variables are being initialised with values from the row with daytime closest to the updating version.
4	The trigger will also contain variables for all relations defined in CLASS_RELATION . These variables are also being initialised depending on inserting or updating mode.
5	Trigger actions defined in CLASS_TRIGGER_ACTION may appear both at the beginning and at the end of the trigger. Depending whether the action is defined as a befor/after insert/update action.
6	The trigger may also contain default values defined in CLASS_ATTRIBUTE.DEFAULT_VALUE. If inserting and not giving a value for an attribute, the defined default value for that attribute will be used. Functions may be used as default value as well.
7	Many aspects of code generation within triggers are automatic and do not rely on metadata. Examples of this include date validation. Several checks are required in respect of dates to ensure, for example, that attributes do not exist before the objects they belong to and that end times are after start times. Another example is the handling of missing items (but which are allowed to be null on calling the view) such as created by, last updated etc.

### Object Trigger Example

The following is an example of an object trigger generated from metadata. The example used is for the object class called GENERATOR.

#### Metadata

The metadata used in this example are the same as those used in the example of generating the GENERATOR object view. Please see Section 3.2.1.1 for those data.

#### Source code

The following is the source code of the trigger.

```

1 CREATE OR REPLACE TRIGGER IUD_GENERATOR
2 INSTEAD OF INSERT OR UPDATE OR DELETE ON OV_GENERATOR
3 FOR EACH ROW
4 - Generated by EcDp_GenClassCode
5 DECLARE
6 CURSOR c_old_version_row(cp_object_id VARCHAR2) IS
7 SELECT daytime
8 FROM EQPM_VERSION

```

```

9 WHERE object_id = cp_object_id
10 ORDER BY daytime;
11 CURSOR c_prev_version_row(cp_object_idVARCHAR2, cp_daytime DATE) IS
12 SELECT *
13 FROM EQPM_VERSION
14 WHERE object_id = cp_object_id
15 AND daytime <= cp_daytime AND Nvl(end_date, cp_daytime + 1) >
cp_daytime;
16 CURSOR c_next_version(cp_object_idVARCHAR2, cp_daytime DATE) IS
17 SELECT daytime
18 FROM EQPM_VERSION
19 WHERE object_id = cp_object_id
20 AND daytime >= cp_daytime AND Nvl(end_date, cp_daytime + 1) >
cp_daytime;
21 lr_version_row EQPM_VERSION%ROWTYPE;
22 lr_curr_main_row EQUIPMENT%ROWTYPE;
23 lr_curr_version_row EQPM_VERSION%ROWTYPE;
24 vt EcTp_GENERATOR.ver_tab_type := EcTp_GENERATOR.ver_tab_type();
25 lb_must_allert_children BOOLEAN := FALSE;
26 lb_update_main_table BOOLEAN := FALSE;
27 lb_update_version_table BOOLEAN := FALSE;
28 lb_update_version_not_dt BOOLEAN := FALSE;
29 ld_prev_daytime DATE := NULL;
30 ln_version_count NUMBER := 0;
31 lb_new_version BOOLEAN := FALSE;
32 lv2_code_changed VARCHAR2(1) := NULL;
33 n_record_status VARCHAR2(1);
34 n_rev_text VARCHAR2(4000);
35 n_created_by VARCHAR2(30);
36 n_created_date DATE;
37 n_last_updated_by VARCHAR2(30);
38 n_last_updated_date DATE;
39 n_init_daytime DATE;
40 n_daytime DATE := :NEW.daytime;
41 o_daytime DATE := :OLD.daytime;
42 n_object_id VARCHAR2(100) := SUBSTR(:NEW.object_id,1,32);
43 o_object_id VARCHAR2(100) := SUBSTR(:OLD.object_id,1,32);
44 n_object_start_date DATE := :NEW.object_start_date;
45 o_object_start_date DATE := :OLD.object_start_date;
46 n_object_end_date DATE := :NEW.object_end_date;
47 o_object_end_date DATE := :OLD.object_end_date;
48 n_code VARCHAR2(100) := :NEW.code;
49 n_description EQUIPMENT.DESCRIPTION%TYPE := :NEW.DESCRIPTION;
50 BEGIN
51
-----
52 - Start Before trigger action block
53 - Need to find object_id to foreign references given by code so
this is available to trigger actions
54 - Also set record status information
55 - Locate correct version used to initialize the version array
56 n_init_daytime := n_daytime;
57 IF UPDATING('OBJECT_START_DATE') THEN
58 IF n_object_start_date <= o_object_start_date THEN - Use first
version
59 n_init_daytime := o_object_start_date;

```

```

60 ELSE
61 n_init_daytime := n_object_start_date;
62 END IF;
63 ELSIF UPDATING('OBJECT_END_DATE') THEN
64 IF n_object_end_date IS NULL OR n_object_end_date >=
Nvl(o_object_end_date,n_object_end_date + 1) THEN - Use latest version
65 SELECT MAX(daytime) INTO n_init_daytime
66 FROM EQPM_VERSION
67 WHERE object_id = o_object_id;
68 ELSIF n_object_end_date <> n_object_start_date THEN
69 n_init_daytime :=
ec_eqpm_version.prev_daytime(o_object_id,n_object_end_date);
70 END IF;
71 END IF;
72 OPEN c_prev_version_row(o_object_id,n_init_daytime);
73 FETCH c_prev_version_row INTO lr_version_row;
74 CLOSE c_prev_version_row;
75 IF INSERTING OR UPDATING('CODE') OR UPDATING('DESCRIPTION') THEN
76 lb_update_main_table := TRUE;
77 END IF;
78 IF NOT DELETING THEN
79 vt.extend;
80 IF UPDATING THEN
81 vt(vt.LAST).object_id := o_object_id;
82 END IF;
83 IF INSERTING OR UPDATING('NAME') THEN
84 vt(vt.LAST).name := :NEW.NAME;
85 lb_update_version_table := TRUE;
86 ELSE
87 vt(vt.LAST).name := lr_version_row.NAME;
88 END IF;
89 IF INSERTING OR UPDATING('DAYTIME') THEN
90 vt(vt.LAST).daytime := :NEW.DAYTIME;
91 lb_update_version_table := TRUE;
92 ELSE
93 vt(vt.LAST).daytime := lr_version_row.DAYTIME;
94 END IF;
95 IF INSERTING OR UPDATING('END_DATE') THEN
96 vt(vt.LAST).end_date := :NEW.END_DATE;
97 lb_update_version_table := TRUE;
98 ELSE
99 vt(vt.LAST).end_date := lr_version_row.END_DATE;
100 END IF;
101 IF INSERTING OR UPDATING('SCREEN_STATUS') THEN
102 vt(vt.LAST).eqpm_status_scr := :NEW.SCREEN_STATUS;
103 lb_update_version_table := TRUE;
104 ELSE
105 vt(vt.LAST).eqpm_status_scr := lr_version_row.EQPM_STATUS_SCR;
106 END IF;
107 IF INSERTING OR UPDATING('OP_FCTY_1_CODE') THEN
108 vt(vt.LAST).op_fcty_class_1_code := :NEW.OP_FCTY_1_CODE;
109 lb_update_version_table := TRUE;
110 ELSE
111 vt(vt.LAST).op_fcty_class_1_code :=
lr_version_row.OP_FCTY_CLASS_1_CODE;
112 END IF;
113 IF INSERTING OR UPDATING('OP_FCTY_1_ID') THEN

```

```

114 vt(vt.LAST).op_fcty_class_1_id := :NEW.OP_FCTY_1_ID;
115 lb_update_version_table := TRUE;
116 ELSE
117 vt(vt.LAST).op_fcty_class_1_id :=
lr_version_row.OP_FCTY_CLASS_1_ID;
118 END IF;
119 IF NOT UPDATING('OP_AREA_CODE') THEN
120 vt(vt.LAST).op_area_code := lr_version_row.OP_AREA_CODE;
121 END IF;
122 IF NOT UPDATING('OP_AREA_ID') THEN
123 vt(vt.LAST).op_area_id := lr_version_row.OP_AREA_ID;
124 END IF;
125 IF NOT UPDATING('OP_PRODUCTIONUNIT_CODE') THEN
126 vt(vt.LAST).op_pu_code := lr_version_row.OP_PU_CODE;
127 END IF;
128 IF NOT UPDATING('OP_PRODUCTIONUNIT_ID') THEN
129 vt(vt.LAST).op_pu_id := lr_version_row.OP_PU_ID;
130 END IF;
131 IF INSERTING OR UPDATING('RECORD_STATUS') THEN
132 n_record_status := :NEW.RECORD_STATUS;
133 lb_update_version_table := TRUE;
134 ELSE
135 n_record_status := lr_version_row.RECORD_STATUS;
136 END IF;
137 IF INSERTING OR UPDATING('CREATED_BY') THEN
138 n_created_by := :NEW.CREATED_BY;
139 lb_update_version_table := TRUE;
140 ELSE
141 n_created_by := lr_version_row.CREATED_BY;
142 END IF;
143 IF INSERTING OR UPDATING('CREATED_DATE') THEN
144 n_created_date := :NEW.CREATED_DATE;
145 lb_update_version_table := TRUE;
146 ELSE
147 n_created_date := lr_version_row.CREATED_DATE;
148 END IF;
149 IF INSERTING OR UPDATING('LAST_UPDATED_BY') THEN
150 n_last_updated_by := :NEW.LAST_UPDATED_BY;
151 ELSE
152 n_last_updated_by := lr_version_row.LAST_UPDATED_BY;
153 END IF;
154 IF INSERTING OR UPDATING('LAST_UPDATED_DATE') THEN
155 n_last_updated_date := :NEW.LAST_UPDATED_DATE;
156 ELSE
157 n_last_updated_date := lr_version_row.LAST_UPDATED_DATE;
158 END IF;
159 IF INSERTING OR UPDATING('REV_TEXT') THEN
160 n_rev_text := :NEW.REV_TEXT;
161 vt(vt.LAST).rev_text := :NEW.rev_text;
162 ELSE
163 n_rev_text := lr_version_row.REV_TEXT;
164 vt(vt.LAST).rev_text := lr_version_row.REV_TEXT;
165 END IF;
166 n_daytime := vt(vt.LAST).daytime;
167 o_daytime := lr_version_row.daytime;
168 END IF; - Not deleting
169 - Support update of only rev_text or last_updated_by/date, need to

```

```

find correct table
170 IF UPDATING AND NOT lb_update_main_table AND NOT
lb_update_version_table AND NOT UPDATING('OBJECT_START_DATE') AND NOT
UPDATING('OBJECT_END_DATE') THEN
171 IF UPDATING('REV_TEXT') OR UPDATING('LAST_UPDATED_BY') OR
UPDATING('LAST_UPDATED_DATE') THEN
172 lr_curr_main_row := ec_EQUIPMENT.row_by_pk(o_object_id);
173 lr_curr_version_row :=
ec_EQPM_VERSION.row_by_pk(o_object_id,o_daytime);
174 IF
NVL(lr_curr_main_row.last_updated_date,lr_curr_main_row.created_date)
>
NVL(lr_curr_version_row.last_updated_date,lr_curr_version_row.created_
date) THEN
175 lb_update_main_table := TRUE;
176 ELSE
177 lb_update_version_table := TRUE;
178 END IF;
179 END IF;
180 END IF;
181 IF INSERTING THEN
182 IF EcDp_objects.GetObjIDFromCode('GENERATOR',n_code) IS NOT NULL
THEN Raise_Application_Error(-20105,'This code is already used within
this class.'); END IF;
183 IF :NEW.class_name IS NOT NULL AND rtrim(upper(:new.class_name))
<> rtrim(upper('GENERATOR')) THEN
Raise_Application_Error(-20104,'Given class_name do not correspond to
view class name.'); END IF;
184 n_object_id := EcDp_objects.GetInsertedObjectID(n_object_id);
185 n_daytime :=
EcDp_objects.GetInsertedDaytime(n_object_start_date,n_daytime,
n_object_end_date);
186 n_object_start_date := n_daytime;
187 n_created_by := Nvl(n_created_by,USER);
188 n_created_date :=
Nvl(n_created_date,EcDp_Date_time.getCurrentSysdate);
189 n_last_updated_by := NULL;
190 n_last_updated_date := NULL;
191 vt(vt.LAST).object_id := n_object_id;
192 vt(vt.LAST).daytime := n_daytime;
193 vt(vt.LAST).end_date := n_object_end_date;
194 vt(vt.LAST).created_by := n_created_by;
195 vt(vt.LAST).created_date := n_created_date;
196 vt(vt.LAST).last_updated_by := n_last_updated_by;
197 vt(vt.LAST).last_updated_date := n_last_updated_date;

```

---



---

1 - Check if Insert violates locked months, in that case this procedure will raise an appropriate error message

```

1
EcDp_Month_lock.validatePeriodForLockOverlap('INSERTING',n_daytime,n_o
bject_end_date,'CLASS: GENERATOR; OBJECT_CODE:' || n_code);

1  - Start Insert relation block
2  vt(vt.LAST).op_fcty_class_1_id :=
EcDp_Objects.GetInsertedRelationID('OP_FCTY_1','FCTY_CLASS_1',vt(vt.LA
ST).op_fcty_class_1_id,vt(vt.LAST).op_fcty_class_1_code,n_daytime);
3  vt(vt.LAST).op_fcty_class_1_code :=
ec_production_facility.object_code(vt(vt.LAST).op_fcty_class_1_id); -
Keep code and id in sync
4  vt := EcTp_GENERATOR.AlignParentVersions(vt,'operational');
5  - End Insert relation block -----
6  ELSIF UPDATING THEN
7  - Start other check
8  IF UPDATING('CLASS_NAME') AND rtrim(upper(:new.class_name)) <>
rtrim(upper('GENERATOR')) THEN Raise_Application_Error(-20104,'Given
class_name do not correspond to view class name.');
```

END IF;

```

9  IF Updating('OBJECT_ID') THEN
Raise_Application_Error(-20101,'Cannot update object_id '); END IF;
10 IF Updating('END_DATE') THEN Raise_Application_Error(-20101,'Cannot
update version end_date, controlled by system '); END IF;
11 - End other check
12
-----
-----
```

```

13 - Lock checking: First check if we are moving object_start_date or
object_end_date
14 IF n_object_start_date <> o_object_start_date THEN
15 EcDp_Month_Lock.validatePeriodForLockOverlap('UPDATING',
16 Least(n_object_start_date,o_object_start_date),
17 Greatest(n_object_start_date,o_object_start_date),
18 'CLASS: GENERATOR; OBJECT_CODE:' || n_code);
19 END IF;
20 IF nvl(n_object_end_date,EcDp_System_Constants.future_date)
21 <> nvl(o_object_end_date,EcDp_System_Constants.future_date) THEN
22 EcDp_Month_Lock.validatePeriodForLockOverlap('UPDATING',
23 LEAST(Nvl(n_object_end_date, EcDp_System_Constants.future_date),
24 Nvl(o_object_end_date,EcDp_System_Constants.future_date)),
25 GREATEST(Nvl(n_object_end_date,EcDp_System_Constants.future_date),
26 Nvl(o_object_end_date,EcDp_System_Constants.future_date)),
27 'CLASS: GENERATOR; OBJECT_CODE:' || n_code);
28 END IF;
29 - Check if we are updating any other main table attributes, Code or
non-versioned relations
30 IF lb_update_main_table THEN
31 EcDp_Month_Lock.validatePeriodForLockOverlap('UPDATING',
32 o_object_start_date,o_object_end_date,
33 'CLASS: GENERATOR; OBJECT_CODE:' || n_code);
34 END IF;
35 - OK, Main table locking checks handled.
36 - Change focus to the version table
37 - Check first if if we are creating a new version
38 IF UPDATING('DAYTIME') THEN

```

```

39 OPEN c_next_version(o_object_id,n_daytime);
40 FETCH c_next_version INTO ld_next_daytime;
41 CLOSE c_next_version;
42 EcDp_Month_Lock.validatePeriodForLockOverlap('UPDATING',
43 n_daytime,n_daytime,
44 'CLASS: GENERATOR; OBJECT_CODE:' || n_code);
45 END IF;
46 - Last check for simple attribute changes
47 IF lb_update_version_not_dt THEN
48 EcDp_Month_Lock.validatePeriodForLockOverlap('UPDATING',
49 n_daytime,vt(vt.LAST).end_date,
50 'CLASS: GENERATOR; OBJECT_CODE:' || n_code);
51 END IF;
52 - End lock check
53 -----
54 IF UPDATING('DAYTIME') THEN
55 SELECT count(1) INTO ln_version_count
56 FROM EQPM_VERSION
57 WHERE object_id = o_object_id AND daytime = n_daytime;
58 IF ln_version_count > 0 THEN Raise_Application_Error(-20105,'There
is already a version starting on the given daytime.');?>
59 IF n_daytime < n_object_start_date OR n_daytime >=
nvl(n_object_end_date,n_daytime+1) THEN
60 Raise_Application_Error(-20104,'Object is not valid for the date
specified. Daytime for new version must be between object start date
and object end date.');
61 END IF;
62 lb_new_version := TRUE;
63 vt(vt.LAST).end_date := n_object_end_date;
64 - Need check for previous and next versions to set end_date
correct;
65 FOR curVer IN c_old_version_row(n_object_id) LOOP
66 IF curVer.daytime < n_daytime THEN ld_prev_daytime :=
curVer.daytime; END IF;
67 IF curVer.daytime > n_daytime THEN
68 vt(vt.LAST).end_date := curVer.daytime;
69 EXIT; - Only interested in the next after the one we inserted
70 END IF;
71 END LOOP;
72 lb_must_allert_children := TRUE;
73 END IF;
74 IF UPDATING('CODE') THEN
75 lb_must_allert_children := TRUE;
76 END IF;
77 IF UPDATING('CODE') THEN
78 lv2_code_changed := 'Y';
79 END IF;
80 - Start check for updating read only columns
81 IF UPDATING('OP_PRODUCTIONUNIT_ID') OR
UPDATING('OP_PRODUCTIONUNIT_CODE') OR UPDATING('OP_AREA_ID') OR
UPDATING('OP_AREA_CODE') THEN
82 Raise_Application_Error(-20101,'Can only update group relations on
parent level!');
83 END IF;
84 - End check for updating read only columns
85 IF lb_update_main_table AND lb_update_version_table THEN
86 EcDp_User_Session.SetUserSessionParameter('JN_NOTES','COMMON');

```

```

87 ELSE
88
EcDp_User_Session.SetUserSessionParameter('JN_NOTES','INDEPENDENT');
89 END IF;
90 - Start update relation block
91 vt(vt.LAST).op_fcty_class_1_id :=
EcDp_Objects.GetUpdatedRelationID(UPDATING('OP_FCTY_1_ID'),UPDATING('O
P_FCTY_1_CODE'),'OP_FCTY_1','FCTY_CLASS_1',vt(vt.LAST).op_fcty_class_1
_id,vt(vt.LAST).op_fcty_class_1_code,n_daytime);
92 vt(vt.LAST).op_fcty_class_1_code :=
ec_production_facility.object_code(vt(vt.LAST).op_fcty_class_1_id); -
Keep code and id in sync
93 IF UPDATING('OP_FCTY_1_ID') OR UPDATING('OP_FCTY_1_CODE') OR
lb_new_version THEN
94 lb_must_allert_children := TRUE;
95 vt := EcTp_GENERATOR.AlignParentVersions(vt,'operational');
96 END IF;
97 - End update relation block
98 n_created_by := Nvl(n_created_by,USER);
99 n_created_date :=
Nvl(n_created_date,EcDp_Date_time.getCurrentSysdate);
100 IF NOT UPDATING('LAST_UPDATED_BY') OR n_last_updated_by IS NULL
THEN n_last_updated_by := USER; END IF;
101 IF NOT UPDATING('LAST_UPDATED_DATE') OR n_last_updated_date IS
NULL THEN n_last_updated_date := EcDp_Date_time.getCurrentSysdate; END
IF;
102 vt(vt.LAST).created_by := n_created_by;
103 vt(vt.LAST).created_date := n_created_date;
104 vt(vt.LAST).last_updated_by := n_last_updated_by;
105 vt(vt.LAST).last_updated_date := n_last_updated_date;
106 END IF;
107 - End Before Trigger action block
-----
-----*****-----
*****
108 - Start Trigger Action block
109 - Any code block defined as a BEFORE trigger-type in table
CLASS_TRIGGER_ACTION will be put here
110 - end Trigger Action block Class_trigger_actions before
-----*****
111 IF NOT DELETING THEN - check of mandatory columns and date logic
112 IF n_object_id IS NULL THEN
Raise_Application_Error(-20103,'Missing value for OBJECT_ID'); END IF;
113 IF n_code IS NULL THEN Raise_Application_Error(-20103,'Missing
value for CODE'); END IF;
114 IF n_object_start_date IS NULL THEN
Raise_Application_Error(-20103,'Missing value for OBJECT_START_DATE');
END IF;
115 FOR ci IN 1..vt.COUNT LOOP
116 IF vt(ci).name IS NULL THEN
Raise_Application_Error(-20103,'Missing value for NAME'); END IF;
117 IF vt(ci).daytime IS NULL THEN
Raise_Application_Error(-20103,'Missing value for DAYTIME'); END IF;
118 END LOOP;
119 END IF; - End mandatory attribute check
120 IF UPDATING THEN

```

```

121 IF UPDATING('OBJECT_START_DATE') AND
EcDp_Objects.IsValidObjStartDate(o_object_id, n_object_start_date) =
'Y' THEN
122 vt := EcTp_GENERATOR.SetObjStartDate(vt, o_object_id,
n_object_start_date, n_last_updated_by, n_last_updated_date);
123 o_daytime := vt(1).daytime;
124 IF n_object_start_date < o_object_start_date THEN - Alert children
since moving back in time
125 lb_must_alert_children := TRUE;
126 IF lb_update_version_table = FALSE AND vt.COUNT > 1 THEN
127 lb_update_version_table := TRUE;
128 END IF;
129 END IF;
130 END IF;
131 IF UPDATING('OBJECT_END_DATE') AND
EcDp_Objects.IsValidObjEndDate(o_object_id, n_object_end_date) = 'Y'
THEN
132 vt := EcTp_GENERATOR.SetObjEndDate(vt, o_object_id,
n_object_end_date, n_last_updated_by, n_last_updated_date);
133 IF nvl(n_object_end_date,o_object_end_date + 1) >
o_object_end_date THEN - Alert children since moving forth in time
134 lb_must_alert_children := TRUE;
135 IF lb_update_version_table = FALSE AND vt.COUNT > 1 THEN
136 lb_update_version_table := TRUE;
137 END IF;
138 END IF;
139 END IF;
140 END IF;
141 IF INSERTING THEN
142 INSERT INTO EQUIPMENT(object_id,CLASS_NAME,
OBJECT_CODE,START_DATE,
143 END_DATE,DESCRIPTION,created_by,created_date,last_updated_by,
144 last_updated_date,rev_no,rev_text,record_status)
VALUES(n_object_id,'GENERATOR',n_code,n_object_start_date,n_object_end
_date,
145 n_description,n_created_by,n_created_date,n_last_updated_by,n_
146 last_updated_date,0,n_rev_text,n_record_status);
147 ELSIF UPDATING AND lb_update_main_table THEN
148 UPDATE EQUIPMENT
149 SET OBJECT_CODE = n_code,
150 DESCRIPTION = n_description,
151 created_by = n_created_by,
152 created_date = n_created_date,
153 last_updated_by = n_last_updated_by,
154 last_updated_date = n_last_updated_date,
155 rev_no = nvl(rev_no,0) + 1,
156 rev_text = n_rev_text,
157 record_status = n_record_status
158 WHERE object_id = o_object_id;
159 END IF; --Inserting
160 IF INSERTING OR lb_new_version THEN
161 FOR ci IN 1..vt.COUNT LOOP
162 INSERT INTO EQPM_VERSION(object_id,daytime,end_date,
163 NAME,EQPM_STATUS_SCR,OP_FCTY_CLASS_1_ID,
164 OP_FCTY_CLASS_1_CODE,OP_PU_ID,OP_PU_CODE,
165 OP_AREA_ID,OP_AREA_CODE,
166 created_by,created_date,last_updated_by,

```

```

167 last_updated_date,rev_no,rev_text,record_status)
168 VALUES(n_object_id,vt(ci).daytime,vt(ci).end_date,
169 vt(ci).NAME,vt(ci).EQPM_STATUS_SCR,vt(ci).op_fcty_class_1_id,
170 vt(ci).op_fcty_class_1_code,vt(ci).op_pu_id,vt(ci).op_pu_code,
171 vt(ci).op_area_id,vt(ci).op_area_code,n_created_by,
172 n_created_date,n_last_updated_by,n_last_updated_date,0,
173 vt(ci).rev_text,n_record_status);
174 END LOOP;
175 - Need to set end_date on previous version
176 IF UPDATING AND ld_prev_daytime IS NOT NULL THEN
177 UPDATE EQPM_VERSION
178 SET end_date = vt(1).daytime,
179 last_updated_by = n_last_updated_by,
180 last_updated_date = n_last_updated_date
181 WHERE object_id = o_object_id and daytime = ld_prev_daytime;
182 END IF;
183 ELSIF UPDATING AND lb_update_version_table THEN - Update current
version
184 UPDATE EQPM_VERSION
185 SET daytime = vt(1).daytime,
186 end_date = vt(1).end_date,
187 NAME = vt(1).name,
188 EQPM_STATUS_SCR = vt(1).eqpm_status_scr,
189 OP_FCTY_CLASS_1_ID= vt(1).op_fcty_class_1_id,
190 OP_FCTY_CLASS_1_CODE= vt(1).op_fcty_class_1_code,
191 OP_PU_ID= vt(1).op_pu_id,
192 OP_PU_CODE= vt(1).op_pu_code,
193 OP_AREA_ID= vt(1).op_area_id,
194 OP_AREA_CODE= vt(1).op_area_code,
195 created_by = n_created_by,
196 created_date = n_created_date,
197 last_updated_by = n_last_updated_by,
198 last_updated_date = n_last_updated_date,
199 rev_no = nvl(rev_no,0) + 1,
200 rev_text = vt(1).rev_text,
201 record_status = n_record_status
202 WHERE object_id = o_object_id AND daytime = o_daytime;
203 FOR ci IN 2..vt.COUNT LOOP
204 INSERT INTO EQPM_VERSION(object_id,daytime,end_date,
205 NAME,EQPM_STATUS_SCR,OP_FCTY_CLASS_1_ID,OP_FCTY_CLASS_1_CODE,
206 OP_PU_ID,OP_PU_CODE,OP_AREA_ID,OP_AREA_CODE,
207 created_by,created_date,last_updated_by,last_updated_date,
208 rev_no,rev_text,record_status)
209 VALUES(n_object_id,vt(ci).daytime,vt(ci).end_date,
210 vt(ci).NAME,vt(ci).EQPM_STATUS_SCR,vt(ci).op_fcty_class_1_id,
211 vt(ci).op_fcty_class_1_code,vt(ci).op_pu_id,
212 vt(ci).op_pu_code,vt(ci).op_area_id,vt(ci).op_area_code,
213 n_created_by,n_created_date,n_last_updated_by,
214 n_last_updated_date,0,vt(ci).rev_text,n_record_status);
215 END LOOP;
216 END IF; - INSERTING OR lb_new_version
217 IF lb_must_allert_children THEN
218 EcTp_GENERATOR.SyncronizeChildren(vt,lv2_code_changed);
219 END IF;
220 IF DELETING THEN
221 Raise_Application_Error(-20102,'Object delete is not allowed, set
object end date. ');

```

```
222 END IF;  
223  
--*****  
*****  
224 - Start Trigger Action block  
225 - Any code block defined as a AFTER trigger-type in table  
CLASS_TRIGGER_ACTION will be put here  
226 --end Trigger Action block Class_trigger_actions after  
227
```

```
*****
*****
228 END;
```

**Points of interest**

Line(s)	Comment
12 - 16	Cursor finding the correct version used to initialising variables. Located the version closest to current daytime.
24	Array holding values for attributes mapped against the version table.
56-369	Before Trigger action block – Initial validation test get a new object_id if INSERTING and find any missing object referred to by code. Also set the record information columns. This is done to give any Trigger action a consistent set of variables to work with. Call to EcTp_GENERATOR.AlignParentVersions will ensure that changes done in the group hierarchy on parent level or above are replicated down to this level.
229-230 246-302	Lock checking included because LOCK_IND = 'Y', note lock checking will be the same for all object classes, LOCK_RULE is not used.
371 - 375 510 - 515	It is possible for a class to define BEFORE and AFTER Trigger actions in table CLASS_TRIGGER_ACTION. These code blocks will be included within the placeholders shown here. Note: there are no trigger actions for this example.
376 - 384	Mandatory column check – This is done after trigger action code, in case these values are set there. If key columns are set by triggers on underlying tables, they must not be set to mandatory, since the mandatory check takes place prior to insert on underlying tables.
388-411	Handling updating of OBJECT_START_DATE and OBJECT_END_DATE. The object is deleted if OBJECT_END_DATE = OBJECT_START_DATE.

415-433		Insert/Update Main tables – Insert into the main table in this case EQUIPMENT. Note for update we are updating all the columns, since we have no control over what a trigger action has done to the variables. This means that underlying triggers can not logic check on UPDATING for a column since this will always be true.
437 - 462		On Insert or updates of daytime that results in a new version: For each attribute there is an insert statement, note this section also handles setting end_date both on the inserted rows, and previous versions.
464 - 500		Update version table in this case EQPM_VERSION. Note we are also in this case updating all columns. Since the group hierarchy is being replicated we need to use an array holding the version variables. Changes done on a parent group level may result in new versions on sub levels.
502 - 504		Ensure that the replication of the group hierarchy is kept in sync.

### Generating a Data Trigger

The following are the "rules" by which metadata are converted into Data triggers. Every Data view should have an associated instead-of trigger. The trigger provides the functionality which allows the insertion, updating and deletion of records in the virtual table represented by the view.

Trigger code is generated in blocks, or groups of lines, related to common topics such as Insert Check, Insert Relation, Create Object, Date Check, Update Relation, Update Attribute and Record Revision. To aid testing and debugging of triggers, block headings are written into the code to indicate the start and end point of each block. In general, in the following table the "rules" are considered in the context of the block of code they relate to. See Section 3.7.1.2 for an example of trigger source code.

No.	Description
1	The trigger will be called IUD_xxx, where xxx is the CLASS_NAME entry in table CLASS
2	Non-disabled, standard, mandatory attributes defined in CLASS_ATTRIBUTE.ATTRIBUTE_NAME are written to a block of code which tests that they are present, i.e. enforces the NOT NULL constraint. If the standard attributes CODE and DAYTIME are not defined in CLASS_ATTRIBUTE, they will be written to the trigger anyway.
3	Non-disabled, non-standard, mandatory attributes defined in CLASS_ATTRIBUTE.ATTRIBUTE_NAME are written to a block of code which tests that they are present, i.e. enforces the NOT NULL constraint
4	Relations defined in CLASS_RELATION appear twice in the trigger code. Two blocks of IF - THEN - ELSE code are generated for each relationship, one for insert operations and one for updates.
5	Many aspects of code generation within triggers are automatic and do not rely on metadata. Examples of this include date validation. Several checks are required in respect of dates to ensure, for example, that attributes do not exist before the objects they belong to and that end times are after start times. Another example is the handling of missing items (but which are allowed to be null on calling the view) such as created by, last updated etc.

### Data Trigger Example

The following is an example of a data trigger generated from metadata. The example used is for the data class called PWEL\_DAY\_STATUS.

#### Metadata

**Table: CLASS (Transposed)**

	PWEL_DAY_STATUS
	null
	DATA
	EC_PROD
	DAY
	WELL

**Table: CLASS\_ATTRIBUTE (Transposed)**

CLASS_NAME	PWEL_DAY_STATUS	ditto	ditto
ATTRIBUTE_NAME	OBJECT_ID	DAYTIME	ON_STREAM_HRS
IS_KEY	Y	Y	N
IS_MANDATORY	Y	Y	N
CONTEXT_CODE	EC_PROD	EC_PROD	EC_PROD
DATA_TYPE	STRING	DATE	NUMBER
PRECISION	Null	null	Null
IS_PRIVATE	Null	null	Null
DEFAULT_VALUE			0.5
REPORT_ONLY_IND			

**Table: CLASS\_ATTR\_DB\_MAPPING**

CLASS_NAME	ATTRIBUTE_NAME	DB_MAPPING_TYPE	DB_SQL_SYNTAX
PWEL_DAY_STATUS	OBJECT_ID	COLUMN	OBJECT_ID
PWEL_DAY_STATUS	DAYTIME	COLUMN	DAYTIME
PWEL_DAY_STATUS	ON_STREAM_HRS	COLUMN	ON_STREAM_HRS

**Table: CLASS\_DB\_MAPPING**

CLASS_NAME	DB_OBJECT_TYPE	DB_OBJECT_OWNER	DB_OBJECT_NAME
PWEL_DAY_STATUS	TABLE	ECKERNEL	PWEL_DAY_STATUS

**Source code**

The following is the source code of the trigger. Line numbers have been added for reference.

```

1 CREATE OR REPLACE TRIGGER IUD_PWEL_DAY_STATUS
2 INSTEAD OF INSERT OR UPDATE OR DELETE ON DV_PWEL_DAY_STATUS
3 FOR EACH ROW
4 Generated by EcDp_GenClassCode
5 DECLARE
6 n_class_name VARCHAR2(30) := EcDB_Utils.ConditionNVL(NOT
Updating('CLASS_NAME'),:NEW.CLASS_NAME,:OLD.CLASS_NAME);
7 n_record_status VARCHAR2(1) := EcDB_Utils.ConditionNVL(NOT
Updating('RECORD_STATUS'),:NEW.record_status,:OLD.record_status);
8 n_rev_no NUMBER := NVL(:OLD.rev_no,0);
9 n_rev_text VARCHAR2(4000):= EcDB_Utils.ConditionNVL(NOT
Updating('REV_TEXT'),:NEW.rev_text,:OLD.rev_text);
10 n_created_by VARCHAR2(30):= EcDB_Utils.ConditionNVL(NOT
Updating('CREATED_BY'),to_char(:NEW.created_by),to_char(:OLD.created_b
y));
11 n_created_date DATE := EcDB_Utils.ConditionNVL(NOT
Updating('CREATED_DATE'),:NEW.created_date,:OLD.created_date);
12 n_last_updated_by VARCHAR2(30):= EcDB_Utils.ConditionNVL(NOT
Updating('LAST_UPDATED_BY'),to_char(:NEW.last_updated_by),to_char(:OLD

```

```

.last_updated_by);
13 n_last_updated_date DATE := EcDB_Utils.ConditionNVL(NOT
Updating('LAST_UPDATED_DATE'),:NEW.last_updated_date,:OLD.last_updated
_date);
14 n_OBJECT_ID VARCHAR2(4000) := EcDB_Utils.ConditionNVL(NOT
Updating('OBJECT_ID'),:NEW.OBJECT_ID,:OLD.OBJECT_ID);
15 n_object_code VARCHAR2(4000) := EcDB_Utils.ConditionNVL(NOT
Updating('OBJECT_CODE'),:NEW.object_code,:OLD.object_code);
16 o_OBJECT_ID VARCHAR2(4000) := :OLD.OBJECT_ID;
17 n_ON_STREAM_HRS NUMBER := EcDB_Utils.ConditionNVL(NOT
Updating('ON_STREAM_HRS'),:NEW.ON_STREAM_HRS,:OLD.ON_STREAM_HRS);
18 n_AVG_CHOKE_SIZE NUMBER := EcDB_Utils.ConditionNVL(NOT
Updating('AVG_CHOKE_SIZE'),:NEW.AVG_CHOKE_SIZE,:OLD.AVG_CHOKE_SIZE);
19 n_AVG_WH_TEMP NUMBER := EcDB_Utils.ConditionNVL(NOT
Updating('AVG_WH_TEMP'),:NEW.AVG_WH_TEMP,:OLD.AVG_WH_TEMP);
20 n_AVG_WH_PRESS NUMBER := EcDB_Utils.ConditionNVL(NOT
Updating('AVG_WH_PRESS'),:NEW.AVG_WH_PRESS,:OLD.AVG_WH_PRESS);
21 n_AVG_BH_PRESS NUMBER := EcDB_Utils.ConditionNVL(NOT
Updating('AVG_BH_PRESS'),:NEW.AVG_BH_PRESS,:OLD.AVG_BH_PRESS);
22 n_daytime DATE := EcDB_Utils.ConditionNVL(NOT
Updating('DAYTIME'),:NEW.DAYTIME,:OLD.DAYTIME);
23 o_daytime DATE := :OLD.DAYTIME;
24 BEGIN
25
-----
-----
26 Start Before Trigger action block
27 Need to find object_ids for foreign references given by code before
we leave the control to user exit
28 Also set record status columns in this section to allow user exits
to overrule them later
29 IF INSERTING THEN - set any default values from
CLASS_ATTRIBUTE.DEFAULT_VALUE
30 NULL; - In case there are no default values
31 IF n_ON_STREAM_HRS IS NULL THEN n_ON_STREAM_HRS :=
to_number('0.5'); END IF;
32 END IF; - set any default values
33 IF INSERTING OR UPDATING THEN
34 n_OBJECT_ID :=
EcDp_Objects.GetInsertedRelationID('OBJECT_CODE','WELL',n_object_id,
n_object_code, n_daytime);
35 IF :old.RECORD_STATUS IN ('V','A') THEN
36 n_rev_no := n_rev_no + 1;
37 END IF;
38 IF INSERTING THEN
39 n_created_by := Nvl(n_created_by,user);
40 n_created_date :=
Nvl(n_created_date,EcDp_Date_Time.getCurrentSysdate);
41 n_RECORD_STATUS := Nvl(n_RECORD_STATUS,'P');
42 ELSE - UPDATING
43 IF NOT UPDATING('LAST_UPDATED_BY') OR n_last_updated_by IS NULL
THEN n_last_updated_by := USER; END IF;
44 IF NOT UPDATING('LAST_UPDATED_DATE') OR n_last_updated_date IS NULL
THEN n_last_updated_date := EcDp_Date_Time.getCurrentSysdate; END IF;
45 END IF; - IF INSERTING
46 END IF; - INSERTING OR UPDATING
47 End Before Trigger Action block

```

```

-----
48
-----
-----
49 ****
*****
50 #NAME?
51 #NAME?
52 -
53 end Trigger Action block Class_trigger_actions before
54
-----*****
55
EcDp_month_lock.AddParameterToList(n_lock_columns,'CLASS_NAME','PWEL_D
AY_STATUS','STRING',NULL,NULL,NULL);
56
EcDp_month_lock.AddParameterToList(n_lock_columns,'TABLE_NAME','PWEL_D
AY_STATUS','STRING',NULL,NULL,NULL);
57
EcDp_month_lock.AddParameterToList(o_lock_columns,'CLASS_NAME','PWEL_D
AY_STATUS','STRING',NULL,NULL,NULL);
58
EcDp_month_lock.AddParameterToList(o_lock_columns,'TABLE_NAME','PWEL_D
AY_STATUS','STRING',NULL,NULL,NULL);
59
EcDp_month_lock.AddParameterToList(n_lock_columns,'OBJECT_ID','OBJECT_
ID','STRING','Y',EcDp_month_lock.isUpdating(UPDATING('OBJECT_ID')),any
data.ConvertVarChar(n_OBJECT_ID));
60
EcDp_month_lock.AddParameterToList(o_lock_columns,'OBJECT_ID','OBJECT_
ID','STRING','Y',EcDp_month_lock.isUpdating(UPDATING('OBJECT_ID')),any
data.ConvertVarChar(:OLD.OBJECT_ID));
61
EcDp_month_lock.AddParameterToList(n_lock_columns,'OBJECT_CODE','OBJEC
T_CODE','VARCHAR2','Y',EcDp_month_lock.isUpdating(UPDATING('OBJECT_ID'
)),anydata.ConvertVarChar(n_object_code));
62
EcDp_month_lock.AddParameterToList(o_lock_columns,'OBJECT_CODE','OBJEC
T_CODE','VARCHAR2','Y',EcDp_month_lock.isUpdating(UPDATING('OBJECT_ID'
)),anydata.ConvertVarChar(:OLD.OBJECT_CODE));
63
EcDp_month_lock.AddParameterToList(n_lock_columns,'DAYTIME','DAYTIME',
'DATE','Y',EcDp_month_lock.isUpdating(UPDATING('DAYTIME')),anydata.Con
vertdate(n_DAYTIME));
64
EcDp_month_lock.AddParameterToList(o_lock_columns,'DAYTIME','DAYTIME',
'DATE','Y',EcDp_month_lock.isUpdating(UPDATING('DAYTIME')),anydata.Con
vertdate(:OLD.DAYTIME));
65
EcDp_month_lock.AddParameterToList(n_lock_columns,'ON_STREAM_HRS','ON_
STREAM_HRS','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING('ON_STREA
M_HRS')),anydata.ConvertNumber(n_ON_STREAM_HRS));
66
EcDp_month_lock.AddParameterToList(o_lock_columns,'ON_STREAM_HRS','ON_
STREAM_HRS','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING('ON_STREA
M_HRS'))

```

```

M_HRS')),anydata.ConvertNumber(:OLD.ON_STREAM_HRS));
67
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_CHOKE_SIZE','AV
G_CHOKE_SIZE','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING('AVG_CH
OKE_SIZE')),anydata.ConvertNumber(n_AVG_CHOKE_SIZE));
68
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_CHOKE_SIZE','AV
G_CHOKE_SIZE','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING('AVG_CH
OKE_SIZE')),anydata.ConvertNumber(:OLD.AVG_CHOKE_SIZE));
69
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_WH_TEMP','AVG_W
H_TEMP','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING('AVG_WH_TEMP'
)),anydata.ConvertNumber(n_AVG_WH_TEMP));
70
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_WH_TEMP','AVG_W
H_TEMP','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING('AVG_WH_TEMP'
)),anydata.ConvertNumber(:OLD.AVG_WH_TEMP));
71
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_WH_PRESS','AVG_
WH_PRESS','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING('AVG_WH_PRE
SS')),anydata.ConvertNumber(n_AVG_WH_PRESS));
72
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_WH_PRESS','AVG_
WH_PRESS','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING('AVG_WH_PRE
SS')),anydata.ConvertNumber(:OLD.AVG_WH_PRESS));
73
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_BH_PRESS','AVG_
BH_PRESS','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING('AVG_BH_PRE
SS')),anydata.ConvertNumber(n_AVG_BH_PRESS));
74
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_BH_PRESS','AVG_
BH_PRESS','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING('AVG_BH_PRE
SS')),anydata.ConvertNumber(:OLD.AVG_BH_PRESS));
75
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_DH_PUMP_SPEED',
'AVG_DH_PUMP_SPEED','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING(
'AVG_DH_PUMP_SPEED')),anydata.ConvertNumber(n_AVG_DH_PUMP_SPEED));
76
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_DH_PUMP_SPEED',
'AVG_DH_PUMP_SPEED','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING(
'AVG_DH_PUMP_SPEED')),anydata.ConvertNumber(:OLD.AVG_DH_PUMP_SPEED));
77
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_DILUENT_RATE',
'AVG_DILUENT_RATE','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING(
'AVG_DILUENT_RATE')),anydata.ConvertNumber(n_AVG_DILUENT_RATE));
78
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_DILUENT_RATE',
'AVG_DILUENT_RATE','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING(
'AVG_DILUENT_RATE')),anydata.ConvertNumber(:OLD.AVG_DILUENT_RATE));
79
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_GL_CHOKE_SIZE',
'AVG_GL_CHOKE','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING(
'AVG_GL_CHOKE_SIZE')),anydata.ConvertNumber(n_AVG_GL_CHOKE_SIZE));
80
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_GL_CHOKE_SIZE',
'AVG_GL_CHOKE','NUMBER','N',EcDp_month_lock.isUpdating(UPDATING(
'AVG_GL_CHOKE_SIZE')),anydata.ConvertNumber(:OLD.AVG_GL_CHOKE_SIZE));

```

```

81
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_GL_DIFF_PRESS',
'AVG_GL_DIFF_PRESS','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_GL_DIFF_PRESS')),anydata.ConvertNumber(n_AVG_GL_DIFF_PRESS));
82
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_GL_DIFF_PRESS',
'AVG_GL_DIFF_PRESS','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_GL_DIFF_PRESS')),anydata.ConvertNumber(:OLD.AVG_GL_DIFF_PRESS));
83
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_OIL_RATE','AVG_
OIL_RATE','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_OIL_RATE')),anydata.ConvertNumber(n_AVG_OIL_RATE));
84
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_OIL_RATE','AVG_
OIL_RATE','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_OIL_RATE')),anydata.ConvertNumber(:OLD.AVG_OIL_RATE));
85
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_GAS_RATE','AVG_
GAS_RATE','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_GAS_RATE')),anydata.ConvertNumber(n_AVG_GAS_RATE));
86
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_GAS_RATE','AVG_
GAS_RATE','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_GAS_RATE')),anydata.ConvertNumber(:OLD.AVG_GAS_RATE));
87
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_FLOW_RATE','AVG_
FLOW_RATE','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_FLOW_RATE')),anydata.ConvertNumber(n_AVG_FLOW_RATE));
88
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_FLOW_RATE','AVG_
FLOW_RATE','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_FLOW_RATE')),anydata.ConvertNumber(:OLD.AVG_FLOW_RATE));
89
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_WATER_RATE','AVG_
WATER_RATE','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_WATER_RATE')),anydata.ConvertNumber(n_AVG_WATER_RATE));
90
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_WATER_RATE','AVG_
WATER_RATE','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_WATER_RATE')),anydata.ConvertNumber(:OLD.AVG_WATER_RATE));
91
EcDp_month_lock.AddParameterToList(n_lock_columns,'AVG_GL_RATE','AVG_G
L_RATE','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_GL_RATE')),anydata.ConvertNumber(n_AVG_GL_RATE));
92
EcDp_month_lock.AddParameterToList(o_lock_columns,'AVG_GL_RATE','AVG_G
L_RATE','NUMBER','N',EcDp_month_lock.isUpdating('
AVG_GL_RATE')),anydata.ConvertNumber(:OLD.AVG_GL_RATE));
93 IF INSERTING THEN
94
EcDp_Production_lock.CheckDirectLock('INSERTING',n_lock_columns,o_lock_
_columns);
95 ELSIF UPDATING THEN
96
EcDp_Production_lock.CheckDirectLock('UPDATING',n_lock_columns,o_lock_
columns);
97 ELSIF DELETING THEN

```

```

98
EcDp_Production_lock.CheckDirectLock('DELETING',n_lock_columns,o_lock_
columns);
99 END IF;
100 IF INSERTING THEN
101 Start Insert check block -----
102
103 IF n_Object_id IS NULL THEN
Raise_Application_Error(-20103,'Missing value for object_id/object
code'); END IF;
104 IF n_DAYTIME IS NULL THEN Raise_Application_Error(-20103,'Missing
value for DAYTIME'); END IF;
105 End Insert check block -----
106 Start Insert relation block
-----
107 IF
ecdp_objects.isValidOwnerReference('PWEL_DAY_STATUS',n_OBJECT_ID) =
'N' THEN
108 Raise_Application_Error(-20106,'Given object id is not of the same
class as the owner class for this data class.');
109 END IF;
110 IF EcDp_Objects.getObjStartDate(n_OBJECT_ID) > n_daytime THEN
Raise_Application_Error(-20109,' Referred object_id can not have a
start date later than this objects start date. ');
END IF;
111 IF n_Daytime >
nvl(EcDp_Objects.getObjEndDate(n_object_id),n_Daytime + 1) THEN
Raise_Application_Error(-20109,'Daytime cannot be greater than owner
objects end date. ');
END IF;
112 End Insert relation block -----
113 INSERT INTO
ECKERNEL_80I2.PWEL_DAY_STATUS(OBJECT_ID,ON_STREAM_HRS,AVG_CHOKE_SIZE,A
VG_WH_TEMP,AVG_WH_PRESS,AVG_BH_PRESS,DAYTIME,CREATED_BY,CREATED_DATE,L
AST_UPDATED_BY,LAST_UPDATED_DATE,REV_NO,REV_TEXT,RECORD_STATUS)
114
VALUES(n_OBJECT_ID,n_ON_STREAM_HRS,n_AVG_CHOKE_SIZE,n_AVG_WH_TEMP,n_AV
G_WH_PRESS,n_AVG_BH_PRESS,n_DAYTIME,n_CREATED_BY,n_CREATED_DATE,n_LA
ST_UPDATED_BY,n_LAST_UPDATED_DATE,n_REV_NO,n_REV_TEXT,n_RECORD_STATUS)
115 RETURNING OBJECT_ID,DAYTIME INTO n_OBJECT_ID,n_DAYTIME ;
116 ELSIF UPDATING THEN
117
118 Start Update check block -----
119 IF n_DAYTIME IS NULL THEN Raise_Application_Error(-20103,'Missing
value for DAYTIME'); END IF;
120 Start Update relation block
121 IF Updating('OBJECT_ID') THEN
Raise_Application_Error(-20101,'Cannot update object_id ');
END IF;
122 IF Updating('DAYTIME') THEN
123 IF EcDp_Objects.getObjStartDate(n_OBJECT_ID) > n_Daytime THEN
Raise_Application_Error(-20109,'Daytime cannot be set prior to owner
objects start date. ');
END IF;
124 IF n_Daytime >
nvl(EcDp_Objects.getObjEndDate(n_OBJECT_ID),n_Daytime + 1) THEN
Raise_Application_Error(-20109,'Daytime cannot be greater than owner
objects end date. ');
END IF;
125 END IF;
126 UPDATE ECKERNEL_80I2.PWEL_DAY_STATUS SET OBJECT_ID =
n_OBJECT_ID,ON_STREAM_HRS = n_ON_STREAM_HRS,AVG_CHOKE_SIZE =

```

```
n_AVG_CHOKE_SIZE,AVG_WH_TEMP = n_AVG_WH_TEMP,AVG_WH_PRESS =
n_AVG_WH_PRESS,AVG_BH_PRESS = n_AVG_BH_PRESS,DAYTIME =
n_DAYTIME,CREATED_BY = n_CREATED_BY, CREATED_DATE = n_CREATED_DATE ,
LAST_UPDATED_BY = n_LAST_UPDATED_BY,LAST_UPDATED_DATE =
n_LAST_UPDATED_DATE ,REV_NO = n_rev_no, REV_TEXT =
n_REV_TEXT,RECORD_STATUS = n_RECORD_STATUS
127 WHERE OBJECT_ID= o_OBJECT_ID AND DAYTIME= o_DAYTIME;
128 ELSE - Deleting
129
130 DELETE FROM ECKERNEL_80I2.PWEL_DAY_STATUS
131 WHERE OBJECT_ID= o_OBJECT_ID AND DAYTIME= o_DAYTIME;
132 END IF; - IF INSERTING
133
--*****-
*****-
134 Start Trigger action block
135 Any code block defined as a AFTER trigger-type in table
CLASS_TRIGGER_ACTION will be put here
136 -
137 end user exit block Class_trigger_actions after
138
```

```
*****
*****
139 END; - TRIGGER IUD_PWEL_DAY_STATUS
```

### Points of interest

Line(s)	Comment
4	Version number of the View Generator used, along with date and time the trigger was generated
29-32	Setting default values, On Insert the trigger picks up any default value statements defined in table CLASS_ATTRIBUTE, this can be constants or function calls.
33-47	Before trigger action block – Picking up object_ids when row referenced by code, setting record status information. NOTE: If you are updating both the OBJECT_ID and the OBJECT_CODE, the value for OBJECT_ID will be used, and the code value will be ignored.
49-52 84-89	Trigger Action block: Any code from CLASS_TRIGGER_ACTION defined as TRIGGER_TYPE = BEFORE/AFTER, for this class will be put here. This class has no trigger actions, so you only see the placeholders.
53-100	Lock check, populating parameter structure and calling given LOCK_RULE.
103-117	Insert block – Check that we have values for mandatory columns and relations, that relations is of correct class type an does not violate start/end-date settings. If these checks are ok then do the physical Insert using the declared n-variables that can be manipulated by trigger actions.
118-129	Update block – much of the same logic as for insert.
130-133	Deletion handling.

### View behaviour

The following covers the functionality of views, i.e. the behaviour of views as provided by their associated triggers. The four standard operations of Insert, Update, Delete and Select are considered for each type of view where appropriate.

The valid permutations are as follows.

	View Type		
	Object	Interface	Data
Operations	Select	Select	Select
	Insert	Insert	Insert
	Update	Update	Update
			Delete

### ***Object View – Select Operation***

What happens when a Select operation is performed on an Object view is described here. Please see the information and examples provided in Section 0 for a complete understanding.

Step	Description
1	If the Object being selected does not have any relations, then the select is a simple query using the mappings between the attributes and their locations.
2	If the Object has one or more relation, then, before displaying the query results, xxx_CODE is derived from the object ID of the corresponding relation ID (where xxx is the appropriate CLASS_RELATION.ROLE_NAME entry). Although only xxx_ID is stored, both it and xxx_CODE are displayed on Select.

### ***Object View – Insert Operation***

What happens when an Insert operation is performed on an Object view is described here. Please see the information and examples provided in Section 0 for a complete understanding.

Step	Description
1	A check is made to see if all mandatory attributes are in the parameter list of the insert statement. If not, an exception is raised.
2	If the inserted Object has a relation defined, then if xxx_ID is null (where xxx is the appropriate CLASS_RELATION.ROLE_NAME entry), but the xxx_CODE value is not null, then the code looks-up the object ID of xxx. If this object ID does not exist, a "Foreign key not found" exception is raised. If it does exist, the object ID is returned and written to the database. Note that this also happens if, in the calling line, the referred object has a start date > daytime. As an example: If country Norway has a start date 01-Jan-1030, and is referred by company "Stone Age Inc." where OBJECT_START_DATE and DAYTIME are 01-Jan-0600, then a "Foreign key not found" error is raised because the country was undefined before 01-Jan-1030. Note that for "ordinary relations" only xxx_ID is stored, not xxx_CODE. For group relations both xxx_ID and xxx_CODE are stored, group relations are always stored in version tables. If both the ID and code is given for a relation, and there is an inconsistency between the 2, the object_id will be used, there will not be raised an exception on this.
3	Step 2 is repeated for all relations
4	If an OBJECT_ID is included in the insert statement then this is used when writing to the database. If it is not included, one is generated automatically
5	Checks that the Object's end date is after its start date, or NULL. If not, an exception is raised.
6	Checks that the start time for attributes is not before the start time of the object itself, i.e. the attribute cannot exist before the object. If not, an exception is raised.
7	If CREATED_BY & CREATED_DATE are null, the Oracle user account and sysdate are used. If not null, the values passed in the insert statement are used. In case of customers having cryptic user id's , a user exit can be invoked in order to resolve the user id to a more readable name. In order to enable this feature the attribute_text for the attribute_type 'USE_UE_USER' should be set as 'Y' in 'CTRL_SYSTEM_ATTRIBUTE' table. The user exit package function that would be invoked is UE_USER.getUserName.  NOTE: the code does not check to see whether the date is sensible, i.e. in relation to the life of the object and attributes
8	Record status is set to P, if left blank (NULL) on insert
9	Rev_no will always be controlled by system, any given revision number will be ignored.

### Object View – Update Operation

What happens when an Update operation is performed on an Object view is described here. Please see the information and examples provided in Section 0 for a complete understanding.

Step	Description
1	Checks that the Object's end date is after its start date. If not, an exception is raised.
2	Checks that the start time for attributes is not before the start time of the object itself, i.e. the attribute cannot exist before the object. If not, an exception is raised.
3	If the object has one or more relations and the update statement attempts to update xxx_CODE (but not xxx_ID) then xxx_ID is derived from the object ID of xxx. If this object ID does not exist, a "Foreign key not found" exception is raised. If it does exist, the object ID is returned and used in the update.
4	If attempting to update an attribute's daytime, a new version (copy) of the record is created in the version table, even if only the attribute's daytime is updated, and not any other attributes. This will not result in a journal entry since old data will not be overwritten.
5	If updating attributes, but not daytime, then a copy of the record is not created in the version table, but a copy is made in the journal table
6	If an attempt is made to update OBJECT_ID an exception is raised as this is not allowed
7	If an attempt is made to update END_DATE, i.e. the end date of the entry in the version table an exception is raised as this is not allowed. Updates to this end date are handled automatically by system triggers to ensure against the creation of overlapping versions of the Object.
8	If an attempt is made to update CLASS_NAME an exception is raised as this is not allowed. Note, however, that it is possible to update the name to be the same as it already was.
9	If LAST_UPDATED_BY & LAST_UPDATED_DATE are null, Oracle user account details are used. If not null, the values passed in the update statement are used. In case of customers having cryptic user id's , a user exit can be invoked in order to resolve the user id to a more readable name. In order to enable this feature the attribute_text for the attribute_type 'USE_UE_USER' should be set as 'Y' in 'CTRL_SYSTEM_ATTRIBUTE' table. The user exit package function that would be invoked is UE_USER.getUserName.  NOTE: the code does not check to see whether the dates are sensible, i.e. in relation to the life of the object and attributes

### Object View – Delete Operation

What happens when a Delete operation is performed on an Object view is described here. Please see the information and examples provided in Section 0 for a complete understanding.

Step	Description
1	If an attempt is made to delete an object an exception is raised as this is not allowed. If an object is not required, OBJECT_END_DATE should be set equal to OBJECT_START_DATE instead. This will prevent the object from appearing in views.

### ***Interface View – Select Operation***

What happens when a Select operation is performed on an Interface view is described here. Please see the information and examples provided in Section 0 for a complete understanding.

Step	Description
1	The select is a simple query built on the underlying Objects, so should return an aggregation of all the data from each of those Objects where the class name matches the interface class

### ***Interface View – Insert, Update and Delete Operations***

For Insert operation this is done by an INSERT on the underlying OV-view, so the same checks and rules as for OV-views applies. For updates the trigger builds up a dynamic update list, and run an update on the underlying tables.

### ***Data View – Select Operation***

What happens when a Select operation is performed on a Data view is described here. Please see the information and examples provided in Section 0 for a complete understanding.

Step	Description
1	If the Data being selected do not have any relations, then the select is a simple query using the mappings between the attributes and their locations.
2	If the Data have one or more relation, then, before displaying the query results, xxx_CODE is derived from the object ID of the corresponding relation ID (where xxx is the appropriate CLASS_RELATION.ROLE_NAME entry). Although only xxx_ID is stored, both it and xxx_CODE are displayed on Select.

### ***Data View – Insert Operation***

What happens when an Insert operation is performed on a Data view is described here. Please see the information and examples provided in Section 0 for a complete understanding.

Step	Description
1	A check is made to see if all mandatory attributes are in the parameter list of the insert statement. If not, an exception is raised.
2	If the inserted Object has a relation defined, then if xxx_ID is null (where xxx is the appropriate CLASS_RELATION.ROLE_NAME entry), but the xxx_CODE value is not null, then the code looks-up the object ID of xxx. If this object ID does not exist, a "Foreign key not found" exception is raised. If it does exist, the object ID is returned and written to the database. Note that only xxx_ID is stored, not xxx_CODE
3	Step 2 is repeated for all relations
4	If an OBJECT_ID is included in the insert statement then this is used when writing to the database. If it is not included, one is provided automatically
5	If CREATED_BY & CREATED_DATE are null, Oracle user account details are used. If not null, the values passed in the insert statement are used. In case of customers having cryptic user id's , a user exit can be invoked in order to resolve the user id to a more readable name. In order to enable this feature the attribute_text for the attribute_type 'USE_UE_USER' should be set as 'Y' in 'CTRL_SYSTEM_ATTRIBUTE' table. The user exit package function that would be invoked is UE_USER.getUserName.  NOTE: the code does not check to see whether the date is sensible, i.e. in relation to the life of the parent object and attributes
6	Record status is set to P if left blank (NULL)
7	Rev_no will always be controlled by system; any given value will be ignored.

#### Notes

- There is a unique key on the combination of CLASS\_NAME, OBJECT\_CODE and DAYTIME, so an exception will occur if an attempt is made to insert the same row twice, even if the OBJECT\_ID for each is different.
- An insert operation does not become an update operation if a primary key violation occurs.

#### **Data View – Update Operation**

What happens when an Update operation is performed on a Data view is described here. Please see the information and examples provided in Section 0 for a complete understanding.

Step	Description
1	If the object has one or more relations and the update statement attempts to update xxx_CODE (but not xxx_ID) then xxx_ID is derived from the object ID of xxx. If this object ID does not exist, a "Foreign key not found" exception is raised. If it does exist, the object ID is returned and used in the update.
2	If an attempt is made to update OBJECT_ID, an exception is raised as this is not allowed
3	If an attempt is made to update CLASS_NAME an exception is raised as this is not allowed. Note, however, that it is possible to update the name to be the same as it already was.
4	If LAST_UPDATED_BY & LAST_UPDATED_DATE are null, Oracle user account details are used. If not null, the values passed in the update statement are used. In case of customers having cryptic user id's, a user exit can be invoked in order to resolve the user id to a more readable name. In order to enable this feature the attribute_text for the attribute_type 'USE_UE_USER' should be set as 'Y' in 'CTRL_SYSTEM_ATTRIBUTE' table. The user exit package function that would be invoked is UE_USER.getUserName.

### Data View – Delete Operation

What happens when a Delete operation is performed on an Object view is described here. Please see the information and examples provided in Section 0 for a complete understanding.

Step	Description
1	A data record will be deleted based on the key values passed to the view

Note that it is assumed that there are not any dependencies between data classes as the code does not prevent the deletion of a parent before the deletion of any child data.

### Journaling

In an Energy Components database, all stored data which may change during production operations can be selected for journaling. Identifying when journaling will occur is configured during initial database set-up.

In the generated view layer, all Object and Data classes each have a generated journal view (ending in \_JN). Journal rules are set on class level, but the actual logging is handled at a physical table level. The journal views present old versions of the entries in the main view, in the same way as if the view was a physical table with an associated journal table.

For data classes, all updates result in an increment on the revision number, and a journal entry made for the old row.

For objects views, changes to attributes will result in journaling if it is an update to the same version of the object (daytime not changed).

### Locking

From release 8.3 it is possible to lock both master- and transactional- data classes. The view generator plays an important role in enforcing this locking. The Locking is enforced on class level based on the class attributes LOCK\_IND and LOCK\_RULE. The details are described in the documentation for CO.1013 Create and maintain class [com.ec.frmw.co.screens] and HA.0007 Monthly Data locking [com.ec.prod.ha.screens].

### General Notes

The following are notes which are pertinent to all metadata, regardless of Class type.



**Reserved words**

If any of the standard revision column names are used as attribute names, the view generation will fail. At present there is not a test for this in the application, so the user will not receive a meaningful error message. Names to avoid include: CREATED\_BY, CREATED\_DATE, LAST\_UPDATED\_BY, LAST\_UPDATED\_DATE, REV\_NO and REV\_TEXT.

**Spreadsheet entry**

If a spreadsheet is used to hold metadata for entry and manipulation prior to insertion in the database, it is important to ensure that the schema name is set correctly in CLASS\_DB\_MAPPING.DB\_OBJECT\_OWNER.

**Language**

English is the official language used throughout the Energy Components product set. It is important, particularly for long-term support purposes, that this is carried through to Class names, attribute names etc. Developers are reminded that all metadata should be defined in English.

## Utilities

This section covers utilities which have been produced by Tieto to aid developers working with the View Generator.

### **EcDesc**

The 'describe' command within Oracle produces misleading results when applied to views generated by the View Generator. For this reason a utility is available which developers should find helpful when interrogating views, as it identifies those columns which are mandatory and are part of the key.

The standard Oracle describes command, when applied to a view such as OV\_COMPONENT, yields the following:

```

1 SQL> desc ov_component
2 Name Null? Type
3 -----
4 CLASS_NAME NOT NULL VARCHAR2(24)
5 OBJECT_ID NOT NULL VARCHAR2(32)
6 OBJECT_START_DATE DATE
7 OBJECT_END_DATE DATE
8 CODE NOT NULL VARCHAR2(32)
9 DAYTIME NOT NULL DATE
10 END_DATE DATE
11 NAME VARCHAR2(240)
12 COMP_ISO_STD_CV_MJM3 NUMBER
13 COMP_DENS_KGM3 NUMBER
14 COMP_CV_MJKG NUMBER
15 COMP_STD_MOLWT NUMBER
16 RECORD_STATUS VARCHAR2(1)
17 CREATED_BY NOT NULL VARCHAR2(30)
18 CREATED_DATE NOT NULL DATE
19 LAST_UPDATED_BY VARCHAR2(30)
20 LAST_UPDATED_DATE DATE
21 REV_NO NUMBER
22 REV_TEXT VARCHAR2(240)
23 SQL>
24 The EcDesc utility, when applied to the same view, yields the
following:
25 SQL> @ecdesc
26 Enter value for class_name: ov_component
27 old 6: WHERE upper(table_name) = upper('&class_name')
28 new 6: WHERE upper(table_name) = upper('ov_component')
29 COLUMN_NAME DATA_TYPE MANDATORY ISKEY
30 -----
31 CLASS_NAME VARCHAR2
32 OBJECT_ID VARCHAR2 ID OR CODE NOT NULL
33 OBJECT_START_DATE DATE
34 OBJECT_END_DATE DATE
35 CODE VARCHAR2 ID OR CODE NOT NULL KEY
36 DAYTIME DATE NOT NULL
37 END_DATE DATE
38 NAME VARCHAR2 NOT NULL
39 COMP_ISO_STD_CV_MJM3 NUMBER
40 COMP_DENS_KGM3 NUMBER
41 COMP_CV_MJKG NUMBER
42 COMP_STD_MOLWT NUMBER
43 RECORD_STATUS VARCHAR2
44 CREATED_BY VARCHAR2
45 CREATED_DATE DATE
46 LAST_UPDATED_BY VARCHAR2
47 LAST_UPDATED_DATE DATE
48 REV_NO NUMBER
49 REV_TEXT VARCHAR2

```

Thus, the developer can more clearly see the constraints associated with the view. It is recommended that developers use the EcDesc command instead of the Oracle desc command because of the more complete information and clarity

provided.

The source code for the utility is provided below for reference.

```

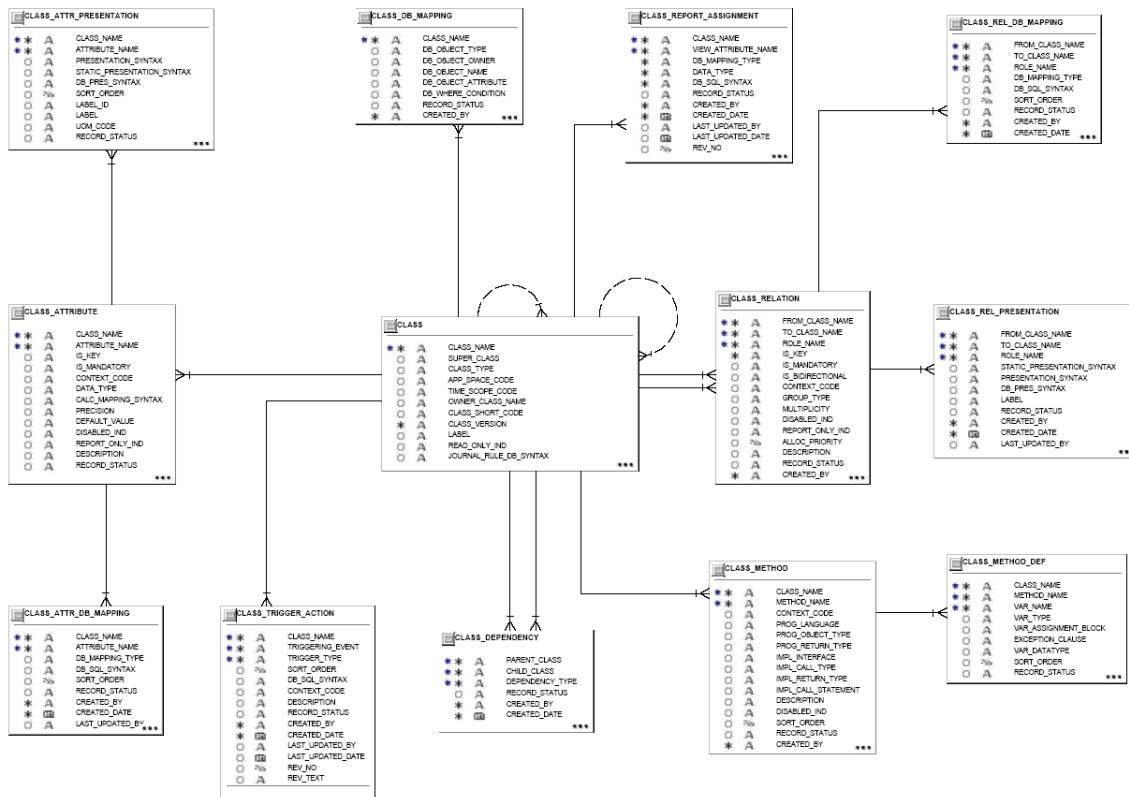
1 EcDesc.sql
2 COLUMN column_name format a30
3 COLUMN data_type format a15
4 COLUMN mandatory format a30
5 COLUMN iskey format a10
6 select column_name ,
7 data_type,
8 ecdp_classMeta.getViewColumnMandatory(table_name,column_name)
MANDATORY,
9 ecdp_classmeta.getViewColumnIsKey(table_name,column_name) ISKEY
10 FROM all_tab_columns
11 WHERE upper(table_name) = upper('&class_name') ;

```

## Future Enhancements

The following is a list of enhancements which are under consideration at the time of writing for inclusion in forthcoming releases of the View Generator.

## Relationship diagram



The diagram above gives an overview of the most important tables in the object class model.

## Detailed table descriptions

This section contains more detailed descriptions of all the tables referred to in this document.

## CLASS

Class definitions from the object class model are stored here.

	CLASS
PK	<b>CLASS_NAME</b>
FK	SUPER_CLASS CLASS_TYPE APP_SPACE_CODE TIME_SCOPE_CODE OWNER_CLASS_NAME CLASS_SHORT_CODE <b>CLASS_VERSION</b> LABEL READ_ONLY_IND LOCK_RULE LOCK_IND JOURNAL_RULE_DB_SYNTAX DESCRIPTION RECORD_STATUS <b>CREATED_BY</b> <b>CREATED_DATE</b> LAST_UPDATED_BY LAST_UPDATED_DATE REV_NO REV_TEXT
FK	

Column	TYPE	DESCRIPTION
class_name	varchar2(24)	Name of class in object class model
super_class	varchar2(24)	Super class in object class model
class_type	varchar2(32)	Class type (DATA, OBJECT, INTERFACE, TABLE, REPORT)
OWNER_CLASS_NAME	varchar2(24)	All data classes must have an object class as owner.
CLASS_SHORT_CODE	varchar2(4)	For future use.
CLASS_VERSION	VARCHAR2(30)	This is intended for numbering class versions and defaults to 1.0
app_space_code	varchar2(32)	Object classes are grouped according to product context, referred to as an EC AppSpace, e.g. EC Production could represent one AppSpace
time_scope_code	varchar2(32)	Time dependent data classes can be periodic, such as 30min, hourly, daily, weekly monthly, quarterly and annual, or they could be event driven where each timestamp represents a new (normally changed) value set. Values (DAY, YR, WEEK, MTH, 1HR, 2HR, 30MIN, SAMPLE, EVENT, VERSIONED, EVENT_CONTINUES)
label	varchar2(64)	Used as display label/title in screens.
READ_ONLY_IND	VARCHAR2(1)	Indicates that the generated view is a read only view. There will not be generated trigger for read only views.
LOCK_RULE	VARCHAR2(240)	Locking check rule, only applicable for Data and table classes. Must be defined with defined parameter-set.
LOCK_IND	VARCHAR2(1)	'Y' means that trigger is generated with lock check.
JOURNAL_RULE_DB_SYNTAX	VARCHAR2(240)	Contains a Boolean expression. This expression will be incorporated into the generated trigger. When evaluates to true, a new revision is created.
description	varchar2(2000)	Description of class.
RECORD_STATUS	VARCHAR2(1)	
CREATED_BY	VARCHAR2(30)	
CREATED_DATE	DATE	
LAST_UPDATED_BY	VARCHAR2(30)	
LAST_UPDATED_DATE	DATE	
REV_NO	NUMBER	
REV_TEXT	VARCHAR2(240)	

**DATA Class Type**

Although transactional data could be considered as being a property of an object, e.g. at 22:30 on 30-Dec-2002, the value of the bottomhole pressure of well A-10 was 94.14, this could be thought of as a property (that changes frequently), we have chosen not to represent such data as properties, but rather as a dependent class, called a data class. In the example, this reading may be part of the Data Class PWEL\_30MIN\_STATUS and be part of a record containing many meter readings (as defined in the data class) at 22:30 for the same object A-10 (meaning that we maintain a record structure for this type of data).

Data classes are always dependent on one parent object class, and its instances are always linked implicitly to an object instance of the class.

In Energy Components many data are time-dependent, but we also have many examples of transactional data not being time dependent. For example, when we invoice a sale, the invoice is modelled in Energy Components with the DOCUMENT, TRANSACTION and LINE\_ITEM data classes all referring to their parent CONTRACT object class. Each of the data classes has the combination of object\_id and its own id, e.g. document\_id, transaction\_id and line\_item\_id, where the combination of object\_id and own id must be unique.

Data classes can be linked to other object classes than their parent, e.g. the LINE\_ITEM data class is also linked to the PRICE\_ELEMENT object class from which it can get its price data. Such links are always foreign key links and do not use the object class relation structure defined above.

Most EC Data classes will be of the form object\_id, daytime and some more properties. However, there will also be situations where multiple dimensions must be modelled in a class, e.g. the stream's shipper component allocated mass requires the stream\_id, shipper\_id and component\_id in order to identify the record.

#### **OBJECT Class Type**

This specifies that the class will be used to instantiate objects used for system configuration.

#### **INTERFACE Class Type**

This specifies that the class is an interface which will be implemented by other classes. See the description under CLASS\_DEPENDENCY.

#### **TABLE Class Type**

The TABLE Class type is used for system tables such as user access tables, verification tables, etc. It is needed because generic screens based on EnergyX must retrieve values through the DAO, and there they have to be defined as Classes.

The view layer for these is simpler than Object, Data and Interface classes because it is usually only a 1:1 mapping to the underlying table or view, but with the possibility of also defining virtual attributes for these classes.

These views/columns will be updateable if the underlying table/view is updateable, virtual attributes are not updateable.

TABLE views are system objects and will not be regenerated every time the view layer is regenerated. To generate table views the following command has to be run :

1. execute EcDp\_GenClassCode.BuildTableViews.

#### **Class Naming**

EC class names are used to generate Oracle objects which currently cannot have more than 30 characters. This means that EC class names shall be a maximum of 24 characters to allow for prefixing the various types of Oracle objects being generated.

Each object class which owns one or more data classes shall have a 4 letter abbreviation which is used to prefix the data class name, e.g. STRM\_1HR\_DATA.

#### **CLASS\_ATTRIBUTE**

Properties associated with a class (class type OBJECT, INTERFACE or DATA) are stored here.

CLASS_ATTRIBUTE	
PK,FK1 PK	<u>CLASS_NAME</u> <u>ATTRIBUTE_NAME</u>
	IS_KEY IS_MANDATORY CONTEXT_CODE DATA_TYPE CALC_MAPPING_SYNTAX PRECISION DEFAULT_VALUE DISABLED_IND REPORT_ONLY_IND DESCRIPTION RECORD_STATUS <b>CREATED_BY</b> <b>CREATED_DATE</b> LAST_UPDATED_BY LAST_UPDATED_DATE REV_NO REV_TEXT

Column	tYPE	DESCRIPTION
CLASS_NAME	VARCHAR2(24)	Class to which this property belongs.
ATTRIBUTE_NAME	VARCHAR2(24)	Name of class property (will become a view attribute).
IS_KEY	VARCHAR2(1)	Is this property part of key within class (Y, N)
IS_MANDATORY	VARCHAR2(1)	Is this property mandatory during an insert (Y, N)
CONTEXT_CODE	VARCHAR2(32)	Object classes are grouped according to product context, referred to as an EC AppSpace, e.g. Ec Production could represent one AppSpace.
DATA_TYPE	VARCHAR2(32)	Data type of this property (STRING, DATE, NUMBER)
CALC_MAPPING_SYNTAX	VARCHAR2(2000)	Contains information used by the Allocation Engine. Not applicable for the view generator.
PRECISION	VARCHAR2(32)	Decimal format for number attributes.
DISABLED_IND	VARCHAR2(1)	Is this attribute disabled(Y,N). If Y then the property is not exposed.
DEFAULT_VALUE	VARCHAR2(2000)	Constant or function call that will be used in the instead of trigger.
REPORT_ONLY_IND	VARCHAR2(1)	Use this to exclude attributes from DV Views. Intended for attributes only used in report views, like YTD columns etc.
DESCRIPTION	VARCHAR2(2000)	Description of property.
RECORD_STATUS	VARCHAR2(1)	
CREATED_BY	VARCHAR2(30)	
CREATED_DATE	DATE	
LAST_UPDATED_BY	VARCHAR2(30)	
LAST_UPDATED_DATE	DATE	
REV_NO	NUMBER	
REV_TEXT	VARCHAR2(240)	

### CLASS\_ATTR\_DB\_MAPPING

Defines where class attributes are actually stored or derived from, e.g. column (& name) or function (& package name).

CLASS_ATTR_DB_MAPPING	
PK,FK1	<u>CLASS_NAME</u> <u>ATTRIBUTE_NAME</u>
	DB_MAPPING_TYPE DB_SQL_SYNTAX SORT_ORDER RECORD_STATUS <b>CREATED_BY</b> <b>CREATED_DATE</b> LAST_UPDATED_BY LAST_UPDATED_DATE REV_NO REV_TEXT

Column	tYPE	DESCRIPTION
CLASS_NAME	VARCHAR2(24)	Class to which this SQL belongs.
ATTRIBUTE_NAME	VARCHAR2(24)	Class property to which this SQL belongs.
DB_MAPPING_TYPE	VARCHAR2(32)	Context for SQL code (COLUMN, FUNCTION, ATTRIBUTE)
DB_SQL_SYNTAX	VARCHAR2(2000)	Column for which the attribute is mapped. Function calls can be defined here.
SORT_ORDER	NUMBER	Relative sorting sequence among other attributes.
RECORD_STATUS	VARCHAR2(1)	
CREATED_BY	VARCHAR2(30)	
CREATED_DATE	DATE	
LAST_UPDATED_BY	VARCHAR2(30)	
LAST_UPDATED_DATE	DATE	
REV_NO	NUMBER	
REV_TEXT	VARCHAR2(240)	

#### CLASS\_ATTR\_PRESENTATION

Holds details of how attributes should be displayed, e.g. what units should be used.

	CLASS_ATTR_PRESENTATION
PK, FK PK, FK	<b>CLASS_NAME</b> <b>ATTRIBUTE_NAME</b>
	PRESENTATION_SYNTAX STATIC_PRESENTATION_SYNTAX DB_PRES_SYNTAX SORT_ORDER LABEL_ID LABEL UOM_CODE RECORD_STATUS <b>CREATED_BY</b> <b>CREATED_DATE</b> LAST_UPDATED_BY LAST_UPDATED_DATE REV_NO REV_TEXT

Column	Type	Description
CLASS_NAME	VARCHAR2(24)	Class to which this presentation information belongs.
ATTRIBUTE_NAME	VARCHAR2(24)	Class property to which this presentation information belongs.
PRESENTATION_SYNTAX	VARCHAR2(2000)	Presentation syntax, e.g. Pop-ups
STATIC_PRESENTATION_SYNTAX	VARCHAR2(2000)	Static Presentation syntax.
DB_PRES_SYNTAX	VARCHAR2(2000)	Presentation syntax defined as a lookup to the database.
SORT_ORDER	NUMBER	Relative sorting sequence among other attributes.
LABEL_ID	VARCHAR2(32)	Unique id for label - look-up required for multilingual support.
LABEL	VARCHAR2(64)	Label text.
UOM_CODE	VARCHAR2(32)	Default unit of measurement to be used for presentation - user settings will override.
RECORD_STATUS	VARCHAR2(1)	
CREATED_BY	VARCHAR2(30)	
CREATED_DATE	DATE	
LAST_UPDATED_BY	VARCHAR2(30)	
LAST_UPDATED_DATE	DATE	
REV_NO	NUMBER	
REV_TEXT	VARCHAR2(240)	

### CLASS\_DB\_MAPPING

Mappings from a class to a database table or view are stored here. Note that, although in principle CLASS\_DB\_MAPPING is used to specify where attributes' data are stored, it is used mainly for data classes.

	CLASS_DB_MAPPING
PK, FK	<b>CLASS_NAME</b>
	DB_OBJECT_TYPE DB_OBJECT_OWNER DB_OBJECT_NAME DB_OBJECT_ATTRIBUTE DB_WHERE_CONDITION RECORD_STATUS <b>CREATED_BY</b> <b>CREATED_DATE</b> LAST_UPDATED_BY LAST_UPDATED_DATE REV_NO REV_TEXT

Column	tYPE	DESCRIPTION
CLASS_NAME	VARCHAR2(24)	Class name.
DB_OBJECT_TYPE	VARCHAR2(32)	Type of database object (TABLE, VIEW)
DB_OBJECT_OWNER	VARCHAR2(30)	Database object owner.
DB_OBJECT_NAME	VARCHAR2(30)	Base table or view name.
DB_OBJECT_ATTRIBUTE	VARCHAR2(30)	Name of any associated attribute table.
DB_WHERE_CONDITION	VARCHAR2(4000)	This is useful in limiting the context of generated views, particularly table views.
RECORD_STATUS	VARCHAR2(1)	
CREATED_BY	VARCHAR2(30)	
CREATED_DATE	DATE	
LAST_UPDATED_BY	VARCHAR2(30)	
LAST_UPDATED_DATE	DATE	
REV_NO	NUMBER	
REV_TEXT	VARCHAR2(240)	

## CLASS\_RELATION

Class relationships in the object class model are stored here. In the EC Object class model, links, relations and group connection(relation) can exist between objects.

Any one relation can only link 2 objects (there are provisions in the model for a third dimension, but this should not be used as complexity increases by another dimension). A class relation is always between 2 object classes and never between data classes, nor between data classes and object classes.

A relation has direction which follows cardinality (from parent to child), or in the case of a 1:1 relation, the dependency rule says that the object being the more independent of the two takes precedence and becomes the "from object", the other the "to object". A relationship has cardinality 1:1 or 1:N.

Note that 'relation' between object classes has a different meaning in this context than in a relationship in a traditional ER model. In EC, a relation is a concept on its own (in an ER model, one would represent this with a connection entity), and we can refer to properties of a relation.

In the EC Object Class model, M:N (many to many) relationships are not supported in the view generation. The two classes must instead both be created as supporting a 1:N multiplicity relationship.

A 1:1 or 1:N relation can also be thought of as a property of an object, e.g. COUNTRY of registration for a COMPANY is a relation between COUNTRY and COMPANY, but seen from the COMPANY object it is just a property. In EC we therefore want to expose such relations as properties.

A group relation is a different kind of relations, it can only exist group relations between objects. The column CLASS\_RELATION.GROUP\_TYPE indicates that this relation is a group relation.

CLASS_RELATION		
PK,FK1	FROM_CLASS_NAME	
PK,FK2	TO_CLASS_NAME	
PK	ROLE_NAME	
	IS_KEY	
	IS_MANDATORY	
	IS_BIDIRECTIONAL	
	CONTEXT_CODE	
	GROUP_TYPE	
	MULTIPLICITY	
	DISABLED_IND	
	REPORT_ONLY_IND	
	ALLOC_PRIORITY	
	DESCRIPTION	
	RECORD_STATUS	
	CREATED_BY	
	CREATED_DATE	
	LAST_UPDATED_BY	
	LAST_UPDATED_DATE	
	REV_NO	
	REV_TEXT	

Column		tYPE		DESCRIPTION
FROM_CLASS_NAME		VARCHAR2(24)		From class name.
TO_CLASS_NAME		VARCHAR2(24)		To class name.
ROLE_NAME		VARCHAR2(24)		Name of relationship (there could be many different relationships between two objects).
IS_KEY		VARCHAR2(1)		<p>Needed when a data class is made unique by a combination of the owner objects.</p> <p>Example, STRM_GAS_COMPONENT is made unique by a combination of STREAM_ID, COMPONENT_ID and daytime so the relation between COMPONENT and STRM_GAS_COMPONENT has IS_key = 'Y'</p>

IS_MANDATORY		VARCHAR2(1)		Mandatory relationship (Y, N)
IS_BIDIRECTIONAL		VARCHAR2(1)		Bi-directional relationship (Y, N)
CONTEXT_CODE		VARCHAR2(32)		Object classes are grouped according to product context, referred to as an EC AppSpace, e.g. Ec Production could represent one AppSpace
GROUP_TYPE		VARCHAR2(32)		Contains the name of the group type. If this field <> null it indicates that the relation should be treated as a group relation.
MULTIPLICITY		VARCHAR2(32)		Multiplicity relationship (1:N, 1:1). In the EC Object Class model, M:N (many to many) relationships are not directly supported in the view generation. The two classes must instead both be created as supporting a 1:N multiplicity relationship.
DISABLED_IND		VARCHAR2(1)		Is this a disabled property of the class (Y, N). If Y then property is not exposed.
REPORT_ONLY_IN_D	VARCHAR2(1)		Use this to exclude attributes from DV Views. Intended for attributes only used in report views, like YTD columns etc.	
LABEL		VARCHAR2(64)		Used as display label in screens.
DESCRIPTION		VARCHAR2(2000)		Description of relation
RECORD_STATUS		VARCHAR2(1)		
CREATED_BY		VARCHAR2(30)		
CREATED_DATE		DATE		
LAST_UPDATED_BY		VARCHAR2(30)		
LAST_UPDATED_DATE		DATE		
REV_NO		NUMBER		

REV_TEXT		VARCHAR2(240)		
----------	--	---------------	--	--

### CLASS\_REL\_DB\_MAPPING

Defines where relationships are actually stored or derived from, e.g. column (& name) or function (& package name).

CLASS_REL_DB_MAPPING	
PK	<u>FROM_CLASS_NAME</u>
PK	<u>TO_CLASS_NAME</u>
PK	<u>ROLE_NAME</u>
	DB_MAPPING_TYPE DB_SQL_SYNTAX RECORD_STATUS CREATED_BY CREATED_DATE LAST_UPDATED_BY LAST_UPDATED_DATE REV_NO REV_TEXT SORT_ORDER

Column	tYPE	DESCRIPTION
FROM_CLASS_NAME	VARCHAR2(24)	From class name within relationship.
TO_CLASS_NAME	VARCHAR2(24)	To class name within relationship.
ROLE_NAME	VARCHAR2(24)	Name of relationship to which this SQL belongs.
DB_MAPPING_TYPE	VARCHAR2(32)	Context for SQL code (COLUMN, FUNCTION, SUB_SELECT)
DB_SQL_SYNTAX	VARCHAR2(2000)	Function calls can be defined here
SORT_ORDER	NUMBER	Relative sorting sequence among other attributes.
RECORD_STATUS	VARCHAR2(1)	
CREATED_BY	VARCHAR2(30)	
CREATED_DATE	DATE	
LAST_UPDATED_BY	VARCHAR2(30)	
LAST_UPDATED_DATE	DATE	
REV_NO	NUMBER	
REV_TEXT	VARCHAR2(240)	

### CLASS\_DEPENDENCY

This table records which classes implement which interfaces.

CLASS_DEPENDENCY		
PK,FK1	PARENT_CLASS	
PK,FK2	CHILD_CLASS	
PK	DEPENDENCY_TYPE	
	RECORD_STATUS CREATED_BY CREATED_DATE LAST_UPDATED_BY LAST_UPDATED_DATE REV_NO REV_TEXT	

**Example:**

Define two classes, Primary Pipeline User (PPU) and Secondary Pipeline User (SPU), which implement the Interface Class NODE.

NODE is defined as a Class of type Interface and will have some of the same (common) attributes as PPU and SPU, but NODE will be a read only view, and data are not stored with CLASS\_NAME = NODE.

The link between NODE and PPU / SPU is reflected in table CLASS\_DEPENDENCY as shown below:

1. PARENT CHILD DEPENDENCY
2. CLASS CLASS TYPE
3. -----
4. NODE PRIMARY\_USER IMPLEMENTS
5. NODE SECONDARY\_USER IMPLEMENTS

The limitation of this concept is that an Interface class has to build on one or more Object classes; interfaces cannot be built on top of other Interfaces.

The attribute names in all classes associated with an Interface must be the same, otherwise they will not be matched, i.e. if the interface has an attribute called EXAMPLE\_TEXT, all of the underlying Classes must have an attribute of the same name.

Note, however, that while the attribute names must be the same, their locations need not be, i.e. the EXAMPLE\_TEXT attribute may be mapped to OBJECTS\_ATTRIBUTE\_ROW.TEXT\_1 in the first of the underlying Objects, but it may be mapped to OBJECTS\_ATTRIBUTE\_ROW.TEXT\_11 for the second Object.

It is assumed that there are not any dependencies between data classes as the trigger code does not prevent the deletion of a parent before the deletion of any child data.

Column	TYPE	DESCRIPTION
PARENT_CLASS	VARCHAR2(24)	Parent class
CHILD_CLASS	VARCHAR2(24)	Child class
DEPENDENCY_TYPE	VARCHAR2(30)	(IMPLEMENTS)
RECORD_STATUS	VARCHAR2(1)	
CREATED_BY	VARCHAR2(30)	
CREATED_DATE	DATE	
LAST_UPDATED_BY	VARCHAR2(30)	
LAST_UPDATED_DATE	DATE	
REV_NO	NUMBER	
REV_TEXT	VARCHAR2(240)	

## CLASS\_Trigger\_ACTION

This extension makes it possible to link in user code before and after the update on physical tables. This code will be linked into the generated code and must be a valid Oracle PL/SQL code only using available functions and variables. The user code is held in a Table called CLASS\_TRIGGER\_ACTION where the View generator picks it up when it generates the Instead of Triggers. This means that any changes to the trigger action code will only be picked up after the Instead of trigger has been generated.

CLASS_TRIGGER_ACTION	
PK	CLASS_NAME
PK	TRIGGERING_EVENT
PK	TRIGGER_TYPE
SORT_ORDER DB_SQL_SYNTAX CONTEXT_CODE DESCRIPTION RECORD_STATUS <b>CREATED_BY</b> <b>CREATED_DATE</b> LAST_UPDATED_BY LAST_UPDATED_DATE REV_NO REV_TEXT	

Column	Data Type	Key	
CLASS_NAME	VARCHAR2(24)	Y	
TRIGGERING_EVENT	VARCHAR2(250)	Y	INSERTING OR UPDATING OR DELETING (Must be a valid IF criteria in Oracle)
TRIGGER_TYPE	VARCHAR2(30)	Y	BEFORE / AFTER
SORT_ORDER	NUMBER		Sequence if more than 1 trigger action on same event.
DB_SQL_SYNTAX	VARCHAR2(4000)		VALID PL/SQL code in the given trigger context
CONTEXT_CODE	VARCHAR2(32)		EC_PROD / EC_TRAN / EC_FRMW / "user"
DESCRIPTION	VARCHAR2(2000)		
DISABLED_IND	VARCHAR2(1)		Is this attribute disabled(Y,N). If Y then the trigger is not exposed.

### Integration of Trigger actions

When the view is generated these trigger action will be pasted into predefined areas within the trigger,

- the Instead of trigger will first do a lookup of any missing object\_id based on code, and set record status information,
- then it leaves the control to any defined trigger action of type 'BEFORE'. The trigger action can then operate on a set of defined variables corresponding to the views columns.
- After that, the Instead of trigger takes over check mandatory columns and does the operation on the physical tables
- Finally it leaves the control to any defined trigger action of type 'AFTER'. The trigger action can then use the set

of defined variables corresponding to the views columns.

Note: The same rule applies to this code as to any trigger code, this is not the place to put your main business logic, it is only intended for smaller maintenance operations.

## How to Setup Customer Defined Password Validation

### EC default password validation

Below are the default password validation rules defined in EC:

1. Current password must be correct.
2. New password must not be empty (null).
3. New password must match the retyped new password.
4. New password must not match the old password.
5. New password must not match the user name.
6. New password must not be less than 6 characters.
7. New password must contain alphanumeric characters.

Administrators can configure the error messages for these rules, so that they can exactly match customer requirements. This can be done by going to the screen *Text Translation* (Configuration / System / Translation) and updating the target for the following list of source error messages:

- Current password is incorrect.
- New password cannot be NULL.
- New password and retyped new password differ.
- New password must differ from old password.
- Password cannot match the user name.
- New password must be at least 6 characters.
- Password needs to contain at least one number and one letter character.

The current default target (and hence the error the user will see) is:  
*"The current or new password is incorrect, or violates some validation rules."*

### Setup customer defined password validation in EC

Customers are allowed to specify their own customized password validation extension by creating a PL/SQL function in the database. The user exit function in EC defaults to calling the customer password validation extension from the database after executing the EC default password validation. The call interfaces which serve this purpose are:

1. z\_PasswordValidation.is\_valid(username varchar2, oldpassword varchar2, newpassword varchar2)
2. z\_PasswordValidation.is\_valid(oldpassword varchar2, newpassword varchar2)

Only a return value of '0' from this function is considered a valid password.

## How to Setup Timezone

### Overview

This document describes how to set up time zone information and Daylight saving switchover times for an EC system. The information has to be set correctly in several places to get a correct setup for the whole system.

EC supports only one time zone per database schema. However, there are a few exceptions to this limitation. It is possible to let some objects related to production day operate with a different time zone. Note that there are clear limitations on this support. For more details see the [Time zone setup for Production Day](#) section.

The first section describes how to set up time zone for the Database Instance and the EcKernel Schema.

### Time zone setup for Instance and schema level

#### Set Oracle DbTimeZone on DB Instance level

Make sure that you have an Oracle database instance created with version 9.2 or higher. The database must be created with the correct time zone setting that corresponds to the server clock returned by the sysdate function. This can be verified with the following query:

```
select to_char(Sysdate,'YYYY-MM-DD HH24:MI'), DbTimeZone from dual;
```

The EC database release should also be deployed.

#### Configuration and usage

All data related to a current operation are stamped and maintained with the date and time using the time zone where they originally arise. Every EC database installation will have a default time zone region equal to 'MET' configured. Identify the time zone that belongs to the operation that is going to be represented in the database. If it differs from 'MET' the default time zone should be updated to the identified one accordingly. Check among the valid entries in the SYS.V\_\$TIMEZONE\_NAMES view by selecting one of the time zone names (TZNAME) from the list returned by the query:

```
SELECT * FROM sys.V_$TIMEZONE_NAMES;
```

#### Set Schema (EcKernel) database configuration

Note that each EC schema can only handle one time zone, but the DB instance can contain several schemas for different local time zones.

There are 2 tables you need to update for each EC schema:

##### 1. T\_PREFERANSE

Update the time zone region name in the T\_PREFERANSE table. Example below:

```
ALTER TRIGGER iu_t_preferanse DISABLE
/
UPDATE t_preferanse set pref_verdi = 'Australia/Melbourne' where
pref_id = 'TIME_ZONE_REGION';
ALTER TRIGGER iu_t_preferanse ENABLE
/
```

An invalid name will cause errors when trying to create data in the database.

Database objects like packages, triggers and views use a new function that returns the local time for the operation in charge. The function is implemented in the EcDp\_Date\_Time package. The function also takes into account daylight

savings time. Developers should use getCurrentSysdate instead of the oracle sysdate function.

```
FUNCTION getCurrentSysdate RETURNDATE;
```

## 2. CTRL\_SYSTEM\_ATTRIBUTE

This is where you set up how the database will convert between UTC and local time. The most important part here is the time for the Daylight saving switchover.

Verify this with the following query:

```
select to_char(daytime,'dd.mm.yyyy hh24:mi'), attribute_type,
attribute_value
  from ctrl_system_attribute
 where attribute_type like 'UTC2LOC%'
  order by attribute_type, daytime;
```

This will list something like this:

DAYTIME	ATTRIBUTE_TYPE	ATTRIBUTE_VALUE
01.01.1900 00:00	UTC2LOCALDEFAULT	1
30.03.2008 01:00	UTC2LOCAL_DIFF	1
26.10.2008 01:00	UTC2LOCAL_DIFF	0
29.03.2009 01:00	UTC2LOCAL_DIFF	1
25.10.2009 01:00	UTC2LOCAL_DIFF	0

The UTC2LOCALDEFAULT attribute gives the standard difference between GMT/UTC time zone and the local time zone for the given schema. The example over is for Central European Time (CET), which in the winter is 1 hour behind UTC.

If you are in Houston, US the UTC2LOCALDEFAULT should be -6. For Sydney, Australia UTC2LOCALDEFAULT should be 10.

The other attribute called UTC2LOCAL\_DIFF gives the switchover time between standard and Daylight saving time. For these, Daytime gives the Switchover time in UTC time. Again the example above is for CET, where the switchover always takes place at 01.00 UTC the last Sunday in March and October.

Note that the local switchover times can vary around the globe, and some places the switchover will always refer to a given local time. But EC still requires that the switchover is given in UTC.

Example: In Houston 2008:

DST started Sunday 9. March 2008 02:00 local standard time  
DST ends on Sunday 2. November 2008 02:00 local daylight time

DAYTIME	ATTRIBUTE_TYPE	ATTRIBUTE_VALUE
09.03.2008 08:00	UTC2LOCAL_DIFF	1
02.11.2009 07:00	UTC2LOCAL_DIFF	0

The attribute value for UTC2LOCAL\_DIFF gives the added offset between standard UTC offset and the offset starting at the given daytime. When DST is not active (in the winter) the attribute value should be 0. When DST is active the value will typically be 1 most places in the world.

Note that some EC components have their own time zone configuration. For these please refer to the documentation on the components.

## Time zone setup for Production Day

Even if the system/schema is set up with one controlling time zone, it is possible to operate with different time zones for some objects related to production day, here are the rules:

- ECIS will support different time zones for all data classes having an object class that links to the object class production\_day. These are (as of today):
  - Well
  - Stream
  - Tank
  - Flowline
  - Production Seperator
  - Facility Class 1
  - Facility Class 2
- ECIS will convert from UTC daytime to local daytime and summer time depending on owner object class connection to production\_day.
- EC will be able to calculate correct "hours in production day" based on owner object class connection to production\_day.
- The customer can create new time zones through the production\_day BF.
- New time zones will not work on any other dates in the system, e.g. not for created\_date, last\_updated\_date, scheduled jobs, instantiation, objects not having link to production\_day or date logic hardcoded in java.
- A user will only see data in the "data local time zone" and not converted to the user local time zone. The same applies to all reporting views.

Note the following limitations in this solution:

1. An object can not switch time zone over time. The system will pick the first production day the object is associated with to determine the time zone.
2. A production day can not switch time zone over time. The system will pick the time zone from the first version row in the production day.
3. For tag based ECIS, the link to the production day object is based on an OBJECT\_ID reference in one of the PK\_ATTR\_x columns, meaning that for data classes the production day link will always be given by the owner class.

### Setting up time zones for production day

The possible time zones that can be selected for a production day must first be defined in EC Codes (Prosty Codes table).

Example:

```
Insert into prosty_codes(code_type,code,code_text, alt_code) values
('TZ_NAME','CET','Central European Time','+01:00');
```

Note that the ALT\_CODE contains the default offset from UTC (same as UTC2LOCALDEFAULT in CTRL\_SYSTEM\_ATTRIBUTE). The EC handling of daylight saving is operating with the same time zone all year and the default offset is the difference to UTC when daylight saving is not active (in winter).

In the Production Day screen you will be able to select from the time zones you defined over in a drop down. The default offset will be shown in the screen as a virtual attribute.

In the Production Day Daylight Saving Time screen: For each production day that is associated with a time zone that has daylight saving, the switchover times must be registered with the daytime in UTC and the corresponding DST flag. 1 means daylight saving starts on that time, 0 means that daylight saving ends at the given time. If the time zone doesn't have daylight saving, you don't have to register any switchover times.

Most of the calculation between UTC and local time is handled by a PL/SQL package called ECDP\_DATE\_TIME. The functions in this package will typically take production\_day as a default parameter, or look it up based on other object references.

The logic will always be that if there is a production day association, the time zone for that production day will be used. If there is no production day sent in, or the production day doesn't have a defined time zone region, the system will always go back to using the time zone configured on schema level.

If you want an object to follow the system/schema time zone, it should not be associated with a production day that has a different time zone set. It is not a good idea to set a different time zone for the default production day object in EC.

## How to Use Date Macro Parameter

### When to use Date Macro?

Date macros are used when there is a need to replace static date parameter values with dynamic values. This means that there is no need for users to change the parameter values manually; the system will change it accordingly. Date macro parameter values can be defined to follow scheduled firing time or actual firing time. Additional macros (various periods such as 'Yesterday', 'Tomorrow', 'Minus 1 Hour', 'Plus 1 Hour', etc) can be used to calculate the actual date parameter value.

### Steps to use Date Macro.

The first step to do is to setup a parameter in the "Business Actions" screen to have the DATE\_MACRO as its type (Ref: figure 1.0).

The screenshot shows the 'Business Actions' screen with a toolbar at the top. Below the toolbar, there are two tabs: 'Name' and 'Java Class'. The 'Name' tab is active, displaying a list of actions. One action, 'UnitTest1', is selected and highlighted with a blue border. Below the list is a navigation bar with page numbers from 1 to 15, where page 7 is highlighted. At the bottom of the list, there is a search bar with the placeholder 'Name' and a dropdown menu for 'Type' set to 'DATE\_MACRO'. To the right of the list, there is a section titled 'Name' with a table header and a message 'No records found.'

Name	Type	Sub Type	Mandatory	Description	Name	Static Value
startDate	EC Code Type	DATE_MACRO	<input type="checkbox"/>			No records found.

Figure 1.0

Next, fill in the macro in the "Schedules" screen.

The screenshot shows the 'SCHEDULE' screen with a toolbar at the top. Below the toolbar, there is a table with columns: Name, Description, Functional Area, Status, Pinned To, Next Fire Time, and Enabled. A row for 'UnitTest1' is selected and highlighted with a blue border. To the right of the table is a 'RUN NOW' button. Below the table, there is a navigation bar with page numbers from 1 to 15, where page 15 is highlighted. At the bottom of the screen, there are tabs: DETAILS, BUSINESS ACTION, SCHEDULE, and MONITOR. The BUSINESS ACTION tab is active, showing a table with columns: Action, Description, Sequence#, and Isolate Transaction. A row for 'UnitTest12' is selected and highlighted with a blue border. Below the BUSINESS ACTION table, there are two sections: 'PARAMETERS' and 'MACRO'. The 'PARAMETERS' section has a table with columns: Name, Value, Sequence#, and Name. The 'MACRO' section has a table with columns: Sequence#, Name, and Value. Both sections show a message 'No records found.'

Name	Description	Functional Area	Status	Pinned To	Next Fire Time	Enabled
TransInvCalc	Run Transaction Inventory Calc	Revenue	EXPIRED			<input type="checkbox"/>
UnitTest1	UnitTest1	EC	WAITING		2014-12-09 1	<input type="checkbox"/>
UpdateSplitKey	Setup the inter-group conversion factor and queue quantity r	Revenue	EXPIRED			<input type="checkbox"/>
UpdateStreamItem	Update Stream Item	Revenue	EXPIRED	WS003040.U		<input checked="" type="checkbox"/>
ValidateDayVO	Validate Daily Numbers	Revenue	EXPIRED			<input type="checkbox"/>

Action	Description	Sequence#	Isolate Transaction
UnitTest12		1	<input type="checkbox"/>

Name	Value	Sequence#	Name	Name	Value
startDate	Schedule Time	1	Yesterday		No records found.
		2	Minus 1 Hour		

Figure 2.0

Here we have selected the symbolic time "Schedule Time" as the starting point.

Then we have added two macros on top of this. The first one is the "Yesterday" macro and second one is the "Minus 1 Hour" macro.

*Example:*

The schedule fires on "2014-12-09T12:00". Based on the macro setting the system will calculate the value of parameter startDate="2014-12-09T12:00" (Ref: figure 2.0, 3.0).

The screenshot shows the SCHEDULE tab of a software interface. At the top right is a green 'RUN NOW' button. Below it is a table with columns: Name, Description, Functional Area, Status, Pinned To, Next Fire Time, and Enabled. The table lists several jobs: TransInvCalc, UnitTest1, UpdateSplitKey, UpdateStreamItem, and ValidateDayVO. UnitTest1 is selected, showing its details below. The 'SCHEDULE' tab is active. Under 'VALIDATION', 'Valid From' is set to '2014-12-09 00:00' and 'Valid To' is set to 'Daily'. Under 'SCHEDULE', the job is set to run 'Every weekday' at 'AT' 12:00:00. Other options like 'Every' and 'day(s)' are also present.

Figure 3.0

If we selected the symbolic time "Actual Time" and if the job was actually triggered at "2014-12-09T12:03" because of the server being busy, then the value of parameter startDate="2014-12-09T12:03" (Ref: figure 4.0).

The screenshot shows the BUSINESS ACTION tab of the same software interface. The top part is identical to Figure 3.0, showing the SCHEDULE tab with a list of jobs and a selected job's details. Below the SCHEDULE tab is the BUSINESS ACTION tab, which is active. It shows a table for 'PARAMETERS' and 'MACRO'. In the 'PARAMETERS' section, 'startDate' is set to 'Actual Time'. In the 'MACRO' section, sequence numbers 1 and 2 are defined: 1 is 'Yesterday' and 2 is 'Minus 1 Hour'. The 'FIXED PARAMETERS' section shows a table with columns: Name, Value, Sequence#, and Name. There are no records found. A watermark 'new' is visible in the bottom right corner.

Figure 4.0

## Macro Sequence Number Validation

When entering the sequence number, ensure that the number is entered correctly. This is to inform the system of the correct sequence of the calculations for date parameter values. Upon clicking the save icon, an error message - "Record already exists" - will appear if the same sequence number is entered (Ref: figure 5.0).

Record already exists.  
 You are trying to INSERT a record that matches an existing record.  
 Please change your input so this record does not match any existing records.

No data has been saved.

**SCHEDULE**

Name	Description	Functional Area	Status	Pinned To	Next Fire Time	Enabled
TransInvCalc	Run Transaction Inventory Calc	Revenue	EXPIRED			<input type="checkbox"/>
UnitTest1	UnitTest1	EC	WAITING		2014-12-09 1	<input type="checkbox"/>
UpdateSplitKey	Setup the inter-group conversion factor and queue quantity	Revenue	EXPIRED			<input type="checkbox"/>
UpdateStreamItem	Update Stream Item	Revenue	EXPIRED	WS003040.U		<input checked="" type="checkbox"/>
ValidateDayVO	Validate Daily Numbers	Revenue	EXPIRED			<input type="checkbox"/>

(15 of 16) < < 7 8 9 10 11 12 13 14 15 16 > >> 5 ▼

**DETAILS BUSINESS ACTION SCHEDULE MONITOR**

**BUSINESS ACTION**

Action	Description	Sequence#	Isolate Transaction
UnitTest12		1	<input type="checkbox"/>

**PARAMETERS**

Name	Value	Sequence#	Name	Name	Value
startDate	Actual Time	1	Yesterday		
		1	Minus 1 Hour		

**MACRO**

Sequence#	Macro
1	Yesterday
1	Minus 1 Hour

**FIXED PARAMETERS**

Name	Value
No records found.	

Figure 5.0

## List of Date Macros

The following is a list of date macros which can be used with the DATE\_MACRO parameter.

Example: Date=05-OCT-2006 Time=10.30 AM

Macro Name	Example Parameter Value
First of Last Year	01-JAN-2005 10.30 AM
First of Month	01-OCT-2006 10.30 AM
First of Next Month	01-NOV-2006 10.30 AM
First of Next Year	01-JAN-2007 10.30 AM
First of Prev Month	01-SEP-2006 10.30 AM
First of Year	01-JAN-2006 10.30 AM
Last of Month	31-OCT-2006 10.30 AM
Last of Next Month	30-NOV-2006 10.30 AM
Last of Prev Month	30-SEP-2006 10.30 AM
Last of Year	31-DEC-2006 10.30 AM
Next Year	05-OCT-2007 10.30 AM
Tomorrow	06-OCT-2006 10.30 AM
Yesterday	04-OCT-2006 10.30 AM
Noon	05-OCT-2006 12.00 PM
Plus 1 Hour	05-OCT-2006 11.30 AM
Minus 1 Hour	05-OCT-2006 09.30 AM
Midnight	05-OCT-2006 12.00 AM

## Row-Level Security

### Introduction

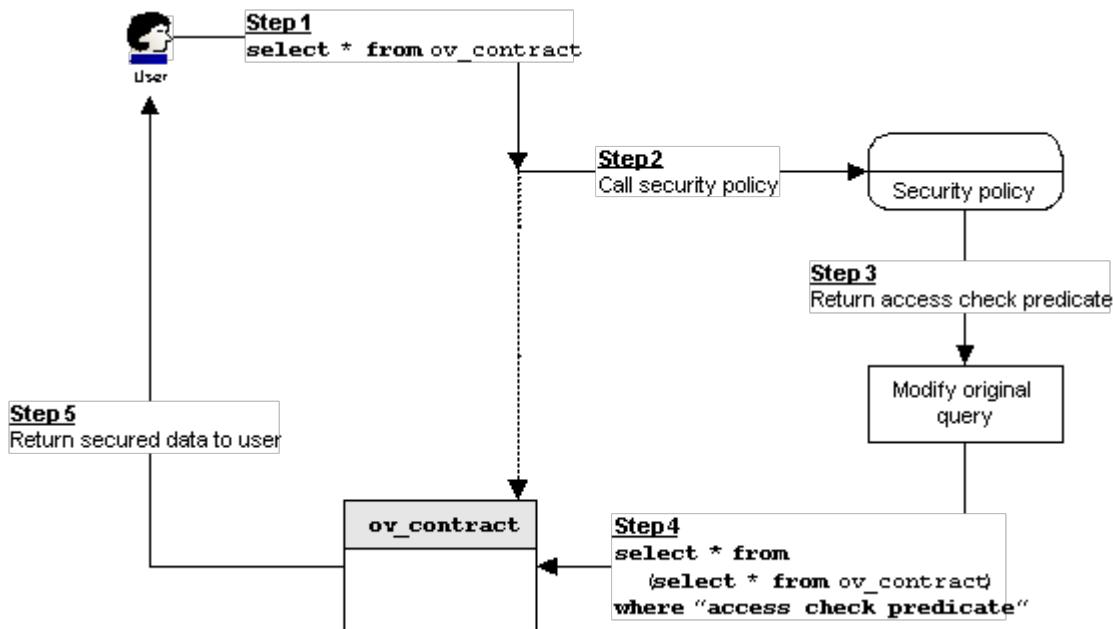
This document describes the Row-Level Security (RLS) functionality in Energy Components (EC).

### Functional overview

RLS in EC is implemented using Oracle's RLS functionality. Such an implementation is also known as a Virtual Private Database (VPD). This section gives an overview of how the mechanism works and describes its main building blocks.

At a high level, VPD security adds a WHERE clause predicate to every SQL statement that is issued on behalf of an individual end user. Depending upon the end user's access privileges, the WHERE clause constrains information to specific rows within the table, hence the name row-level security.

One of the main benefits of VPD is that it is built into the database server instead of the application logic. Consequently the security cannot be bypassed by users that access the database using ad-hoc query and reporting tools.



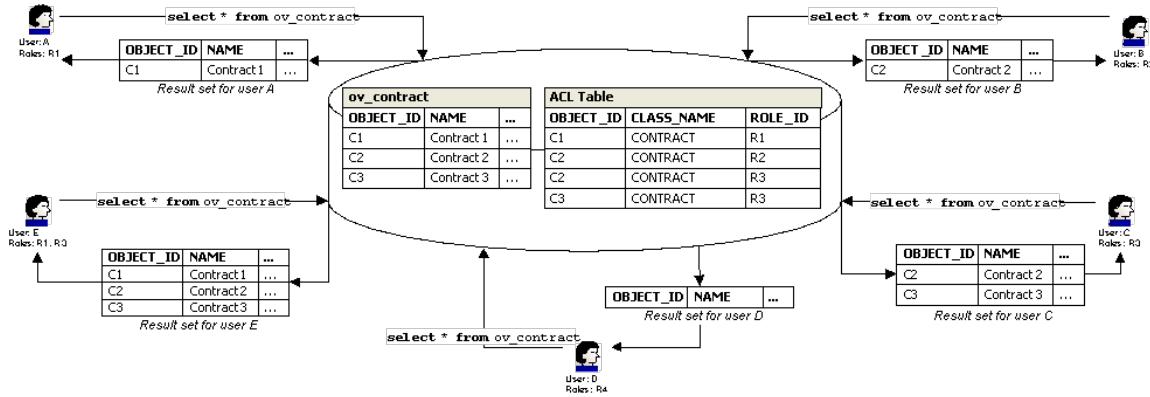
This figure illustrates how the Oracle database processes an end-user query on an access controlled table/view:

1. A user sends SQL to the database server. Either indirectly, via the EC application or a reporting tool, or directly, via an ad-hoc query tool.
2. The associated table/view triggers a pre-defined security policy.
3. The security policy returns an access check predicate.
4. Oracle modifies the original SQL statement by appending the predicate.
5. Oracle executes the modified query and returns secured data to the user.

### The Access Control List

The Access Control List is a database table that identifies the access controlled objects and the application roles that are granted access to those objects. Each ACL record identifies an object and a role. Objects are identified by `object_id` and `class_name`. Roles are identified by `role_id`.

The access check predicates returned by the security policy functions perform ACL table lookups to check that the current user has at least one role with access to the object. If none of the user roles have access to a given object, the records for that object will be removed from the result set.



This figure illustrates result set filtering against the ACL table using user roles and the object identifiers. Note that the result set for user D is empty, since the ACL table contains no contract records for role R4. Also note that the result set for user E contains all contract records. C1 is included in the result set since user E has role R1. Contracts C2 and C3 are included because user E is also assigned the role R3.

### The Application Contexts

The EC application server maintains a pool of database connections. All connections are made as the energyx database user. When the EC application issues a query or a DML statement to the database, it does so on behalf of an application user using an available connection from the pool. Access privileges in EC are associated with application users and roles, not with database users.

The security policy must return a predicate that checks the access for the current application user, not for the energyx database user. Thus, the current application user must be communicated to the database connection, before that connection is used to issue queries and DML statements on his/her behalf. The access control mechanism uses Oracle application contexts for that purpose.

An application context is a secure data cache stored in Oracle's User Global Area or Shared Global Area. Global application contexts, stored in the SGA, are shared across all sessions. A global application context is used to cache information in the t\_basis\_\* tables. The global context is initialised at the first access to an access controlled database object, and is refreshed whenever a user's configuration is changed.

Secure session-based application contexts, stored in the UGA, are session private. The session-based application context is used to hold the application user id together with his/her associated roles. The security policy predicates get the current user's roles from the session-based application context.

Whenever the EC application gets a database connection from the pool, that connection's session context is initialised with information about the current application user and roles. Thus, when the application issues queries and DML statements using that connection, the predicates pick up the current user roles from the session context. For ad-hoc query and reporting tools, the session context can be set up using installation specific login triggers.

### The Security Policies

Security policies are added to the generated views of any access controlled OBJECT and DATA class. Note that INTERFACE classes can also be access controlled. INTERFACE views are composed as a union of OBJECT views. Policies added to the OBJECT views will therefore be invoked by queries against the INTERFACE views. .

OBJECT class views have a single security policy. The policy checks that the ACL table contains at least one record for the object and user roles in question.

DATA class views may have multiple security policies. If the owner class is access controlled, one policy will check that the user has access to the owner object. In addition, policies will be added for each of the DATA class relations that are subject to access control. For DATA class views with multiple security policies, all predicates must evaluate to true.

These are rules that apply:

1. If no limitation is defined on an object class, all roles will have full access to all objects in that class.
2. If a role has no limitations for a given class, then the default is **full access**.
3. If there are limitations on levels in an object group model, the limitation on the lowest level will take precedence over limitations on a higher level.
4. If a user has access to more than one role, then the following applies:
  - If there is a limitation on one of the roles while other roles have full access to the same class, the user will have full access to all objects in the class.
  - To limit access to a class for a user, all roles for the user must have limitations related to the given class.

		Delivery points					
		DP1	DP2	DP3	DP4	DP5	DP6
Contracts	C1						
	C2						
	C3						
	C4						
	C5						
	C6						

Role: R1

Role: R2

This figure illustrates a combination of CONTRACT and DELIVERY POINT.

Role R1 is limited to see contract C4 only.

Role R2 is limited to see delivery point DP4 only.

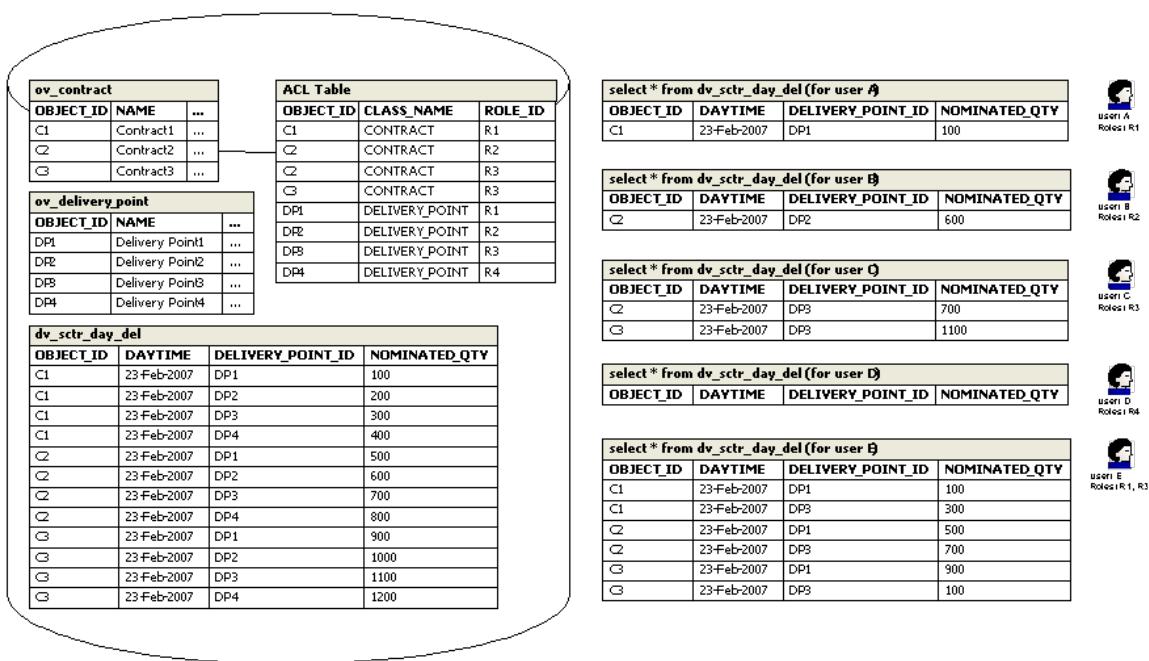
#### Example 1:

If the user get role R1 only, all delivery points for contract C4 could be seen, but none for the other contracts.

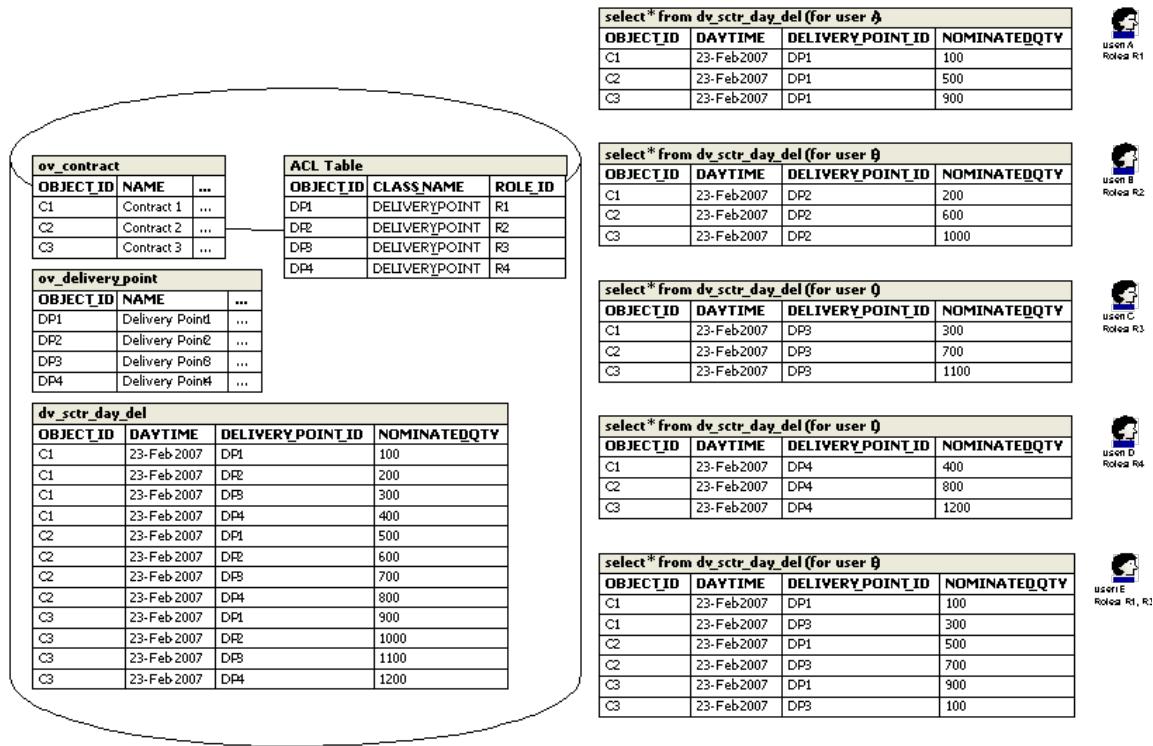
#### Example 2:

If the user get both roles R1 and R2, all delivery points and all contracts could be seen. This is because all roles by default will have full access to all objects in these classes. When role R1 is limited to contract C4, role R2 will have full access to all contracts. And when role R2 is limited to delivery point DP4, role R1 will have full access to all delivery points.

This can be handled by setting no access for role R2 in CONTRACT and no access for role R1 in DELIVERY POINT. Then the user will be limited to see delivery point DP4 and contract C4 only.



This figure illustrates a configuration where the CONTRACT and DELIVERY\_POINT classes, and the DELIVERY\_POINT relation in the SCTR\_DAY\_DEL class, are subject to access control. Two security policies will be added to the generated **dv\_sctr\_day\_del** view: one for the CONTRACT owner class and one for the DELIVERY\_POINT relation. A user will get access to a record if he/she has access to both the CONTRACT and the DELIVERY\_POINT.



This figure illustrates a configuration where the DELIVERY\_POINT class and the DELIVERY\_POINT relation in SCTR\_DAY\_DEL are subject to access control. In this configuration, a single security policy will be applied. The policy will check that the current user has access to the DELIVERY\_POINT in question. No security policy will be added for the owner class since CONTRACT is not subject to access control.

## How to Setup SSL/HTTPS

### Abstract

The Wildfly application server can be setup to communicate over an encrypted network to the clients. The document describes how to generate and use a self-signed certificate. It has been updated with instructions for how to set-up a certificate signed by Verisign. The information regarding the latter is based on information found at the following website: <http://www.verisign.com/support/ssl-certificates-support/security-solutions.html>. One should carefully consider reading through the information issued by Verisign, both for mistyping in this document and possibly new updates from Verisign.

### Configuring Wildfly to use SSL/HTTPS encryption

Follow the steps below to configure SSL/HTTPS.

#### Step 1 - Create/Install the Server Certificate

The EC app distribution comes with a script to create a self-signed certificate. This can be used for testing purposes, but will not be officially trusted so it will create warnings in the users browser.

Prerequisite is an installed and running EC application server. The EC distribution should be unzipped to *C:\EnergyComponents\ec-app-install*, or adjust the mentioned paths accordingly.

The certificate will be created and stored in *server.keystore* and put in ..\domain\configuration of the WildflyDir. A client certificate will be imported into client.truststore, that will be used by the Java truststore so EC trusts itself.

Open a command prompt and navigate to the folder *C:\EnergyComponents\ec-app-install*. Then issue this command (Run commands as admin):

Run without parameters for usage instructions:

```
C:\EnergyComponents\ec-app-install>GenSelfSignedSSLCert.bat
This utility will create a self signed SSL certificate which can be
used for testing purposes.
The certificate will be created as server.keystore and put in
..\domain\configuration of WildFly
Usage:
  GenSelfSignedSSLCert.bat [-help] [-jbossDir=path] [-hostname=name]
  [-ip=ipadress]
    -help (-?)           : Prints this description of the command line
    options.
    -jbossDir (-j)       : The folder where the WildFly is installed
    -hostname (-h)       : The hostname to generate certificate for (Can
    be the same as IP)
    -ip      (-i)        : The IP that the server is listening on
  Example :
  GenSelfSignedSSLCert.bat -j=C:\EnergyComponents\wildfly-8.2.1.Final
  -h=WL2017153 -i=10.52.64.50
```

A new server certificate will be created in ..\<WildFlyDir>\domain\configuration\server.keystore.

Note: If you have purchased a certificate from an external source (so it will be validated by a third-party CA centre) then you must add (or replace) it to the server.keystore file by using the keytool.exe tool. Step 1a and 1b describes the procedure for Verisign.

#### Step 1a – Generate a public/private key-pair to be signed by Verisign

This step should be used to generate a new public/private key pair. The public key can later be signed by Verisign.

In a command prompt navigate to the folder: <WildFly>\domain\configuration\|. Then issue the following command:

```
%JAVA_HOME%\bin\keytool -genkey -keyalg RSA -alias JBoss-SSL -keystore server.keystore
```

Enter the information request by the keytool command. Verisign recommends that the first and last name to be the url.

### **Step 1b – Generate a CSR (Certificate Signing Request) for Verisign**

When the key pair has been generated, a signing request file should be generated.

In a command prompt navigate to the folder: <WildFly>\domain\configuration). Then issue the following command:

```
%JAVA_HOME%\bin\ keytool -certreq -keyalg RSA -alias JBoss-SSL -file certreq.csr -keystore server.keystore
```

The certreq.csr file can later be used to in the Verisign enrollment process.

### **Step 2 - Configure WildFly**

The EC app distribution comes with a script to configure a default SSL setup for EC.

This utility will run the WildFly CLI script **defaultSSL.cli** in the current directory and will do the following configuration :

- Stop servers in ec-server-group.
- Add a SSL security realm *SSLRealm*. And configure it with:
  - keystore name: server.keystore
  - location : jboss.domain.config.dir
  - password : energy
  - alias : JBoss-ssl
- Setup a HTTPS listener for undertow. Listening on the IP defined in the system property jboss.bind.address or if undefined, localhost. It will listen on the *https* socket, default 8443.
- Set the system property *com.ec.eccore.transport.guarantee* to CONFIDENTIAL. This will force a redirect from normal http connections to secure https connections.
- Set the system property *javax.net.ssl.trustStore* to <WildFlyDir>\domain\configuration\client.truststore.
- Start the servers in ec-server-group.

Important: a certificate has to be present in WildFlyDir\domain\configuration with the name server.keystore and a client certificate in WildFlyDir\domain\configuration\client.truststore before running this.

Both of these can be created with the **GenSelfSignedSSLCert.bat** script.

Open a command prompt and navigate to the folder C:\EnergyComponents\ec-app-install. Then issue this command (Run commands as admin):

Run without parameters for usage instructions :

```
C:\EnergyComponents\ec-app-install>defaultSSL.bat -?
This utility will run the WildFly CLI script defaultSSL.cli in the
current
directory. This will setup a HTTPS listener for undertow.
Important : a certificate has to be present in
JBOSS_DIR/domain/configuration
with the name server.keystore before running this.
A self-signed certificate can be created with the
GenSelfSignedSSLCert.bat utility.
After running this JBOSS will listen to either 127.0.0.1 or the ip
defined in jboss.bind.address
"So if you are testing this locally. Add
-Djboss.bind.address=<your.ip.x.x> to JAVA_OPTS in
bin\domain.conf.bat"
Usage:
defaultSSL.bat [-help] [-jbossDir=path]
-help (-?)           : Prints this description of the command line
options.
-jbossDir (-j)       : The folder where the JBoss is installed and
running.
```

### Step 3 – Import the certificate issued by Verisign.

Verisign states that: "If you are installing an SSL Certificate with SGC, you need to copy an [Intermediate CA Certificate](#) and install it before you install the SSL Certificate."

In a command prompt navigate to the folder: <WildflyDir>\domain\configuration. Then issue the following command:

```
%JAVA_HOME%\bin\keytool -import -keystore server.keystore -alias root
-trustcacerts -file verisign.ca
```

The verisign.ca being the file with the Intermediate CA certificate, at the time of writing the file can be obtained by following the link above.

The certificate received from verisign is stored in a file called certificate.cer. To import the certificate for your host use the following command:

```
%JAVA_HOME%\bin\keytool -import -alias JBoss-ssl -file certificate.cer
-trustcacerts -keystore server.keystore
```

### Step 4 - Configure forced HTTPS

To make the server redirect all connections to https change the com.ec.eccore.transport.guarantee property to CONFIDENTIAL.

Energy Components requires connections to localhost and when **com.ec.eccore.transport.guarantee** is set to CONFIDENTIAL a certificate for localhost has to be installed into WildFly's trusted keystore. This is handled by the **defaultSSL.bat** script.

Make sure that the system property **javax.net.ssl.trustStore** is set to a valid client.truststore.

### Step 5 - Import certificates for external services (Optional)

For external web services like Signicat PKI the corresponding certificates have to be imported into the client.truststore. As an example we import Signicat's client certificate

To use Signicat PKI signing over https and **com.ec.eccore.transport.guarantee=CONFIDENTIAL**, Signicat's client

certificate has to be imported into JBoss's client.truststore.

How to:

1. Go to <https://id.signicat.com/signatureservice/services/signatureservice> with Internet Explorer.
2. Click on the padlock in the address bar.
3. Click View Certificates.
4. Click Details tab.
5. Copy To File.
6. Export the certificate as DER encoded binary X.509 (.CER).
7. Save it somewhere.

Import it into the client.truststore with

- cd ..\WildFlyDir\domain\configuration
- keytool -import -v -keystore client.truststore -storepass energy -file C:\EnergyComponents\signicat.cer -alias signicat

### Step 6 - Configure Firewall

Energy Components requires that the normal HTTP/1.1 connector must be open for localhost access. You may configure your firewall to block this port (80, 8080, 8081 and so on) for *external* access.

## Smart Journaling

### Abstract

The EC application has for many years supported journaling of old data values when changes are made to a row in the database.

The challenge with this solution has been that it also copies and store rows that can be considered to be "noise" because it gives no additional information. Typically empty instantiated records, data capture updates to empty columns, etc.

The Smart journaling tries to minimize the journaling by excluding some of the cases where journaling is not needed. It also addresses the need for recording revision text on deleted rows.

### Requirements addressed

1. Avoid journaling for records created by system users (instantiated rows), based on user id stamp on the old record. There is a defined list of these users (1).
2. Avoid journaling when a record is updated by defined system users (i.e. Transfer) defined in another system user list, (2), and the old record is also last updated by the same system user.
3. The lists of system users under 1 and 2 are independent lists, and can contain 1 or more defined system users each.
4. When a record is deleted, this should always result in a new journal entry including the revision text entered for the deleted record.
5. The journal entries created as a result of a delete in the EC client should have JN\_OPERATION=DEL.

### Configuration

The behaviour and requirements above can be achieved in EC by configuring the following system attributes in CTRL\_SYSTEM\_ATTRIBUTE.

Attribute Name	Default Values	Comments
JOUR_USER_EXCL_OLD	'SYSTEM,INSTANSIATE'	List (1) of user where journaling will not be done ref req 1 (above).
JOUR_USER_EXCL_NEW	'TRANSFER'	List (2) of user where journaling will not be done ref req 2 (above).
JOUR_USER_EXCL_OLD_IND	N	Flag that determines if JOUR_USER_EXCL_OLD list is used or not.
JOUR_USER_EXCL_NEW_IND	N	Flag that determines if JOUR_USER_EXCL_NEW list is used or not.

In the default configuration journaling will behave as for previous versions of EC. **The projects will actively have to turn on and configure these to get the new Smart Journaling functionality.**

The exception is the journal entry for deleted records (ref. requirement 4). A delete in the client will make the DAO trigger an update to set rev\_text before it does the delete against the class.

**Note that when changes are done to the 4 attributes listed here, the View layer need to be regenerated before the new configuration applies.**

Since the lists of system users are generated into the Instead of triggers, you should try to keep the number of system users in each list down to a reasonable level (<10).

### Example 1 - Data class

Consider the following operations done to a given row in DV\_PWEL\_DAY\_STATUS.

Operation	Change	Who	Comment
Insert	Empty row	INSTANSIATE	System user (list 1)
Update	ON_STREAM_HRS = 12	BILL	Normal user
Update	ON_STREAM_HRS = 14	TRANSFER	System user (list 2)
Update	ON_STREAM_HRS = 16	TRANSFER	System user (list 2)
Update	ON_STREAM_HRS = 24	BILL	Normal user
Delete	Set Rev_text and delete row	BILL	Normal user

After these operations the row will no longer exist in the normal view DV\_PWEL\_DAY\_STATUS, but the history of the row can be found in the journal table.

Without Smart Journaling turned on (the default setting), the entries in DV\_PWEL\_DAY\_STATUS\_JN will look like this:

JN_OPERATION	REV_NO	ON_STREAM_HRS	CREATED_BY	CREATED_DATE	LAST_UPDATED_BY	LAST_UPDATED_DATE	LAST_UPDATED_BY	LAST_UPDATED_DATE
1 DEL	5	24	INSTANSIATE	... 26.03.2010 13:45:36	• BILL	... 26.03.2010 13:46:24	• 6 • BILL	... 26.03.2010 13:46:24
2 UPD	4	24	INSTANSIATE	... 26.03.2010 13:45:36	• BILL	... 26.03.2010 13:46:10	• 6 • BILL	... 26.03.2010 13:46:10
3 UPD	3	16	INSTANSIATE	... 26.03.2010 13:45:36	• TRANSFER	... 26.03.2010 13:45:58	• 6 • TRANSFER	... 26.03.2010 13:45:58
4 UPD	2	14	INSTANSIATE	... 26.03.2010 13:45:36	• TRANSFER	... 26.03.2010 13:45:49	• 6 • TRANSFER	... 26.03.2010 13:45:49
5 UPD	1	12	INSTANSIATE	... 26.03.2010 13:45:36	• BILL	... 26.03.2010 13:45:37	• 6 • BILL	... 26.03.2010 13:45:37
6 UPD	0	INSTANSIATE	... 26.03.2010 13:45:36	•	...	...	• 6 •	...

Note that the revisions are sorted descending, and that you find traces of all the operations as separate rows in the journal table.

With Smart Journaling activated, some of these operations are not journaled, because of the filtering related to system users. DV\_PWEL\_DAY\_STATUS\_JN will look like this:

JN_OPERATION	REV_NO	ON_STREAM_HRS	CREATED_BY	CREATED_DATE	LAST_UPDATED_BY	LAST_UPDATED_DATE	LAST_UPDATED_BY	LAST_UPDATED_DATE
1 DEL	3	24	INSTANSIATE	... 26.03.2010 13:54:04	• BILL	... 26.03.2010 13:54:50	• 4 • BILL	... 26.03.2010 13:54:50
2 UPD	2	24	INSTANSIATE	... 26.03.2010 13:54:04	• BILL	... 26.03.2010 13:54:38	• 4 • BILL	... 26.03.2010 13:54:38
3 UPD	1	16	INSTANSIATE	... 26.03.2010 13:54:04	• TRANSFER	... 26.03.2010 13:54:28	• 4 • TRANSFER	... 26.03.2010 13:54:28
4 UPD	0	12	INSTANSIATE	... 26.03.2010 13:54:04	• BILL	... 26.03.2010 13:54:04	• 4 • BILL	... 26.03.2010 13:54:04

If we inspect the entries here, we will see that there are 2 operations that have not been journaled:

1. When BILL updated the empty row created by INSTANSIATION (ref, requirement 1).
2. When TRANSFER updated ON\_STRM\_HRS from 14 -> 16 (ref. requirement 2). TRANSFER as a system user (list 2) can update his own changes several times without triggering a new journal entry.

The journal entries can be used when generating audit logging reports for changes. It can be useful to also illustrate the difference the use of Smart Journaling will have on the queries for this. Using the following query (Ctrl\_system\_attribute.master\_data\_report\_on = 'Y'):

```
select * from table (ecdp_change_logging.getlogging('PWEL_DAY_STATUS',
sysdate - 3, sysdate + 3));
```

Without Smart Journaling:

CLASS_NAME	OBJECT_ID	ROW_ID	ATTRIBUTE	OLD_VALUE	NEW_VALUE	LAST_UPDATED_BY	LAST_UPDATED_DATE	REV_NO	REV_TEXT	REC_ID
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	OBJECT_ID	6D71E08E360	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	DAYTIME	... 26.03.2010	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	ON_STREAM_HRS	... 24	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_CHOKE_SIZE	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_WH_TEMP	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_WH_PRESS	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_BH_TEMP	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_BH_PRESS	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_DH_PUMP_SPEED	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_OIL_RATE	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GAS_RATE	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_LIQ_VOL	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GL_RATE	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GAS_ENERGY	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GAS_GCV	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GAS_GOV	...	Deleted Row	... BILL	... 26.03.2010 13:46:24	✓ 5	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	ON_STREAM_HRS	... 14	Deleted Row	... TRANSFER	... 26.03.2010 13:45:58	✓ 3	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	ON_STREAM_HRS	... 12	Deleted Row	... TRANSFER	... 26.03.2010 13:45:49	✓ 2	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	ON_STREAM_HRS	... 12	Deleted Row	... BILL	... 26.03.2010 13:45:37	✓ 1	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	OBJECT_ID	... New Row	6D71E08E360	INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	DAYTIME	... New Row	26.03.2010	INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	ON_STREAM_HRS	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_CHOKE_SIZE	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_WH_TEMP	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_WH_PRESS	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_BH_TEMP	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_BH_PRESS	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_DH_PUMP_SPEED	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_OIL_RATE	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GAS_RATE	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_WATER_RATE	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_LIQ_VOL	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GL_RATE	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GAS_ENERGY	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GAS_GCV	... New Row	... INSTANSIATE	... INSTANSIATE	... 26.03.2010 13:45:36	✓ 0	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	ON_STREAM_HRS	... 16	... 24	... BILL	... 26.03.2010 13:45:38	✓ 2	Compete the test by deleir	4C637B8
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	ON_STREAM_HRS	... 12	... 16	... TRANSFER	... 26.03.2010 13:45:28	✓ 1	Compete the test by deleir	4C637B8

In this listing the insert and delete operations result in a line for each attribute to show the value of all the class attributes. For the updates only changed attributes are listed.

With Smart Journaling:

CLASS_NAME	OBJECT_ID	ROW_ID	ATTRIBUTE	OLD_VALUE	NEW_VALUE	LAST_UPDATED_BY	LAST_UPDATED_DATE	REV_NO	REV_TEXT	REC_ID
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	OBJECT_ID	6D71E08E360	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	DAYTIME	... 26.03.2010	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	ON_STREAM_HRS	... 24	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_CHOKE_SIZE	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_WH_TEMP	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_WH_PRESS	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_BH_TEMP	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_BH_PRESS	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_DH_PUMP_SPEED	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_OIL_RATE	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GAS_RATE	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_WATER_RATE	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_LIQ_VOL	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GL_RATE	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GAS_ENERGY	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	AVG_GAS_GCV	...	Deleted Row	... BILL	... 26.03.2010 13:54:50	✓ 3	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	ON_STREAM_HRS	... 16	... 24	... BILL	... 26.03.2010 13:54:38	✓ 2	Compete the test by deleir	900135C
PWEL_DAY_S	6D71E08E3	AS1_Well_001 2010-03-2	ON_STREAM_HRS	... 12	... 16	... TRANSFER	... 26.03.2010 13:54:28	✓ 1	Compete the test by deleir	900135C

Note again that the main difference is that the record of the empty INSTANSIATION row is not there, and that only the last of TRANSFER's 2 changes to the row are recorded.

## Example 2 - Object class with 4eyes approval and mandatory revision text

Consider the following operations done to a given row in OV\_AREA



Operation	Change	Who	Comment
Insert	New area name = 'Test21'	INSTANSIATE	System user (list 1)
Approval	Make record official	SYSADMIN	Normal user
Update	name = 'name2'	BILL	Normal user
Approval	Make record official	SYSADMIN	Normal user
Update	name = 'name3'	TRANSFER	System user (list 2)
Approval	Make record official	SYSADMIN	Normal user
Update	name = 'name4'	TRANSFER	System user (list 2)
Approval	Make record official	SYSADMIN	Normal user
Update	name = 'name6'	BILL	Normal user
Approval	Make record official	SYSADMIN	Normal user
Delete	Set OED = OSD and rev_text	BILL	Normal user

After these operations the row will no longer exist in the normal view OV\_AREA, but the history of the row can be found in the journal table.

Without Smart Journaling turned on (the default setting), the entries in OV\_AREA\_JN will look like this:

JN_OPERATION	REV_NO	NAME	CREATED_BY	CREATED_DATE	LAST_UPDATED_BY	LAST_UPDATED_DATE	REV_TEXT	APPROVAL_BY
1 DEL	0.5	... name6 ... INSTANSIATE	... JOE	26.03.2010 13:31:40	... BILL	26.03.2010 13:34:12	... so sorry to see you go ...	BILL
2 UPD	0.4	... name6 ... INSTANSIATE	... BILL	26.03.2010 13:31:40	... sysadmin	26.03.2010 13:33:51	... changed name4 to name6 ...	sysadmin
3 UPD	0.3	... name4 ... INSTANSIATE	... TRANSFER	26.03.2010 13:31:40	... sysadmin	26.03.2010 13:32:05	... changed name to name4 ...	sysadmin
4 UPD	0.2	... name3 ... INSTANSIATE	... TRANSFER	26.03.2010 13:31:40	... sysadmin	26.03.2010 13:31:51	... changed name to name3 ...	sysadmin
5 UPD	0.1	... name2 ... INSTANSIATE	... BILL	26.03.2010 13:31:40	... sysadmin	26.03.2010 13:31:40	... changed name to name2 ...	sysadmin
6 UPD	0.0	Test21 ... INSTANSIATE	... sysadmin	26.03.2010 13:31:40	... sysadmin	...	...	... sysadmin

Note that in this setting also a system user like TRANSFER has to set a rev\_text to be able to update the row.

With Smart Journaling turned on, the entries in OV\_AREA\_JN will look like this:

JN_OPERATION	NAME	CREATED_BY	CREATED_DATE	LAST_UPDATED_BY	LAST_UPDATED_DATE	REV_NO	REV_TEXT	APPROVAL_BY
1 DEL	name6 ... INSTANSIATE	... JOE	26.03.2010 13:21:09	... BILL	26.03.2010 13:22:01	0.3	... so sorry to see you go ...	BILL
2 UPD	name6 ... INSTANSIATE	... BILL	26.03.2010 13:21:09	... sysadmin	26.03.2010 13:21:46	0.2	... rev_text is required here 2 ...	sysadmin
3 UPD	name4 ... INSTANSIATE	... TRANSFER	26.03.2010 13:21:09	... sysadmin	26.03.2010 13:21:24	0.1	...	sysadmin
4 UPD	name2 ... INSTANSIATE	... BILL	26.03.2010 13:21:09	... sysadmin	26.03.2010 13:21:09	0.0	...	sysadmin

Note here the following:

- Even if rev\_text is mandatory, BILL is still allowed to update the empty INSTANSIATION record without putting in a revision text, since this does not trigger a new journal entry.
- The system user TRANSFER doesn't have to provide a "dummy" revision text, because the system recognises this as a system driven operation.
- Normal users must always provide a rev\_text that is different from the previous text.
- All the approval operations are recorded on the same row in the journal table as they belong to.

## How to create, install and configure a Jasper Report

### Introduction

JasperReports is a popular open source reporting engine. It is able to produce pixel-perfect documents that can be exported to different formats. Currently we support PDF, Excel and Xml, and we support database as input source. We now support Jasper Reports version 6.0. The old version of Jasper (3.1.0) is still available, but will not be maintained any further.

Jasper reports are based on Jasper Definition files. These files can be created using a third party tool. JasperSoft now provides a new tool called Jasper Studio. It will be the customers responsibility to provide users with tooling for this. See <https://community.jaspersoft.com/>. There is both a community- and a professional edition of Jasper Studio.

Note: EC 10.4 and older versions of EC will only support Jasper Report 3.1, and will be based on 'EC Internal' as report system, as before. For EC 11, the old Jasper version will still be supported. The configuration will be like before, using EC Internal as report system. Therefore, when migrating to EC 11, reports will work as before and will not require any rewrite/recompile to the new version of Jasper.

This How-To will give step-by-step instructions for how to create, configure and install a Jasper Report.

## Step-by-Step Instructions

### *Create and Install a Report*

1. Create and compile the jasper report using a third-party tool. EC will use the compiled jasper file (.jasper). Note the following:
  - a. The report parameters configured in EC are referenced as parameters in the jrxml. The values will be injected before the report is generated.
  - b. To be able to reference images and subreports within a jasper report, you can create a parameter 'BASE\_URL' (with java.lang.String as class) and use it in the jrml expression, e.g. \${P{BASE\_URL}} + "energy\_components.png". The BASE\_URL parameter will be populated automatically by the EC system. It will be the full url to where the main jasper report file is located. You don't have to populate this parameter yourself.
  - c. If you want to use an image file located under <wildfly-home-dir>/web-resources/images, you can create a parameter 'HOST' (with java.lang.String as class) and use it in the jrml expression, e.g \${P{HOST}} + "web-resources/images/ec-logo.png". The HOST parameter will be populated automatically, like the BASE\_URL parameter.
  - d. Example:
    - i. Assume EC is running at this location: <http://10.52.64.71:8080>
    - ii. Assume the Wildfly folder is located at C:\JBOSS\wildfly-8.2.1.Final on the server.
    - iii. Assume the **Jasper report file** and the **image** (tietologo.png) are located at C:\JBOSS\wildfly-8.2.1.Final\web-resources\reports on the server.
    - iv. The BASE\_URL parameter coming into the report will look like this: <http://10.52.64.71:8080/web-resources/reports/>.  
(Enter this as the **default value expression** for the BASE\_URL parameter. Then you will be able to see the image when previewing the report).
    - v. The Image expression in the Jasper report will look like this: \${P{BASE\_URL}} + "tietologo.png".  
Note that the reference to the image is picky when it comes to uppercase/lowercase. The reference to the image file name must match case-wise!
    - vi. If everything is correctly defined the image will show in the Jasper Studio development environment.
2. Install the report on the EC server. This can be done in two alternative ways:
  - a. Copy all report resource files for the report (including compiled jasper files and images) to the standard web-resources folder for reports given by <wildfly-home>/web-resources/reports. You can create your own subfolders under 'reports'. NOTE: The web-resources folder is not password-protected.
  - b. Include all report resource files in a custom web-application, and build and deploy this to the EC server the standard way. The web application does not need to be included in ec-app.ear. Instructions for how to do this is given in a section called 'How to add/modify custom jasper reports in an operational context'.

### *Configure a Report*

1. In the EC treeview, select Reporting and Report Template. Add a new Report Template with EC Jasper Report as report system.



2. The template will have a system parameter called Jasper Definition Url. If you copied the report to the web-resources folder, the url should be '/web-resources/reports/<any sub-folders>/<report definition file> (e.g. /web-resources/reports/myReports/myReport.jasper). If you added it to a custom web application, you need to use the context root for that application. Note that the url must start with '/'.
3. Add a report parameter for report format. Note: Only pdf, xml and xls is currently supported. Pdf will be generated by default if format is not provided.
  - a. Name: FORMAT
  - b. Parameter Type 'EC Code Type'
  - c. Parameter Sub Type: REPORT\_FORMAT
  - d. Value: Select the appropriate format
4. Add additional parameters that you need for your report. The parameters will be forwarded and made available as 'parameters' for the Jasper report.
5. In the Report Definition screen, add a new definition and add the new report template.
6. In the Report Administration screen, add a new report for the new report definition.
7. Provide data for the report parameters and click on Generate.
8. Click on Refresh.
9. The generated report should be available under 'Generated Reports'.

# Messaging Technical Documentation

## Introduction

This section describes the Message Handling Module in the Energy Components (EC). It describes the module in a sufficient level of detail to gain understanding of the actual implementation of the system.

The flow of messages and their statuses seen from EC are presented.

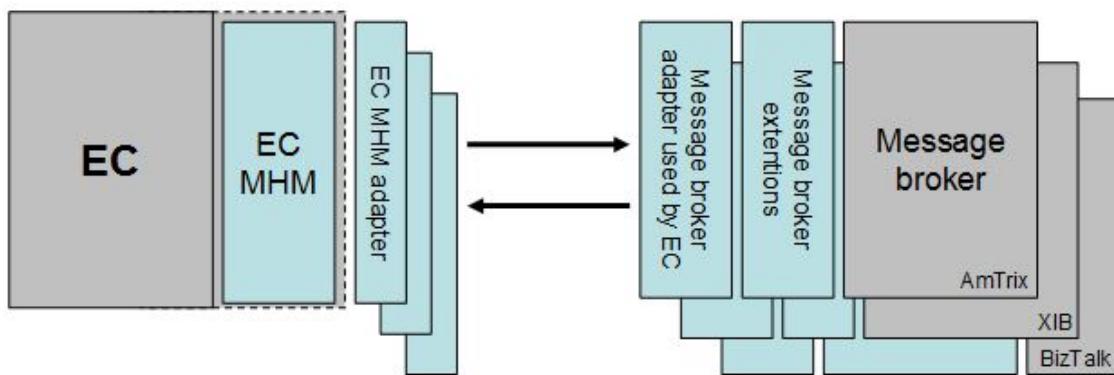
The interface between EC and the message broker have been tightly coupled in the past. Now this coupling is loose but the implementing projects are responsible for the message broker.

## Business Context

This chapter contains an overview of the Message Handling Module (EC-MHM) and the interface to the message broker.

### MHM - architecture overview

The following figure illustrates the functionality in the messaging software:



e

**Figure 3 1: Overview of the interface between EC and the message broker.**

The parts that constitute the messaging are:

- The message broker.
- The EC specific extensions to the message broker.
- The EC-MHM adapter handling all interaction with the message broker.
- A set of screens in EC covering required end-user functionality using data from both EC and the message broker with extensions.

All messages sent from EC to the message broker are routed through the EC-MHM adapter which transfers the messages to the message broker. Incoming messages are fetched by the EC-MHM module through the EC-MHM adapter and routed to the destination module in EC. Due to security reasons firewalls will typically block requests initiated from the message broker. Hence, the EC-MHM pushes and pulls messages to and from the message broker. There is no actual EC-MHM adapter delivered with EC. The projects have to implement a class implementing an interface and configure EC-MHM to use this class. This configuration is done by the system settings feature in EC.

The interface between the EC-MHM adapter and message broker would typically be based on JDBC or web services.

The message broker or the message broker extensions are responsible for the message archive. This should have the original and converted messages and their status and log.

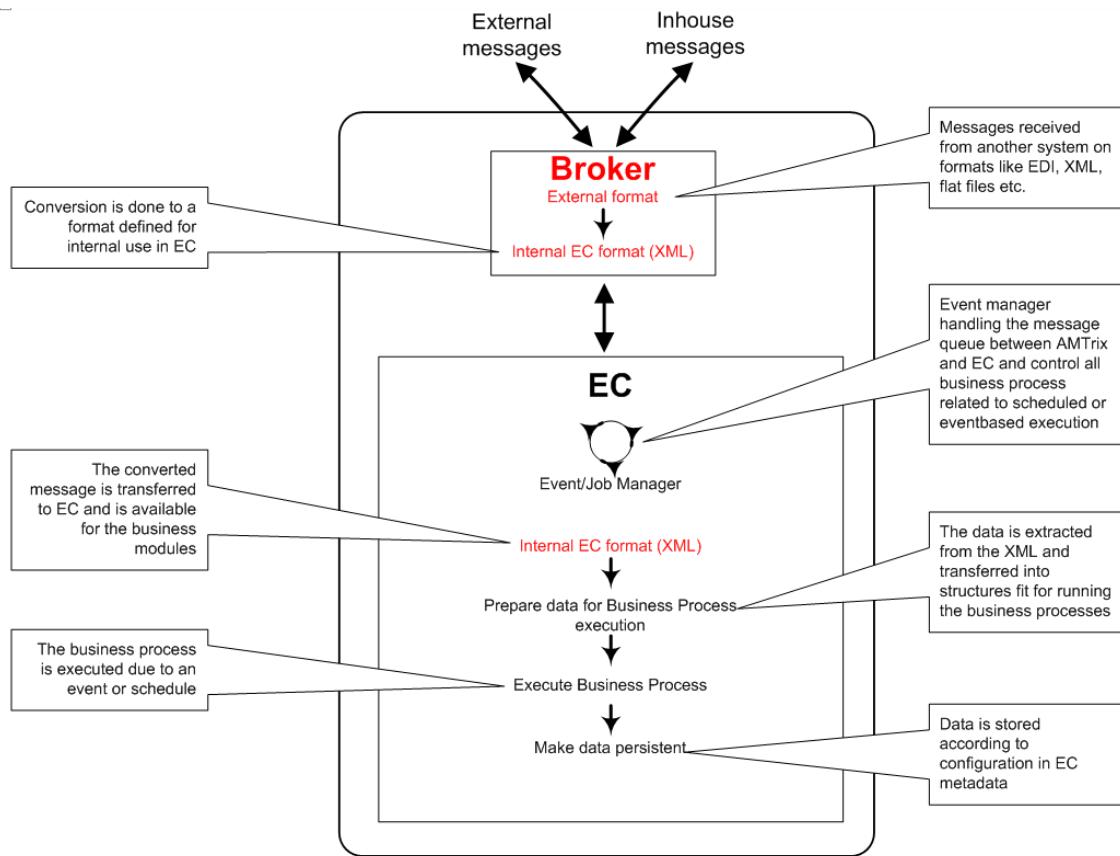
The message broker extensions make it possible to provide the services that EC needs without modifying the message broker used. Then it is more straight forward to upgrade the message broker in use.

## Message Handling Module

### General scope

The MHM strategy is to have whatever required end-user functionality to handle messaging in a standard EC user interface. The MHM is the interface for interchange of messages in EC. All EC sub-systems are connected to the message broker via the EC-MHM module. The message broker handles the logistics side of this process. To support this part, there are functions in the message broker that technically secure the interchange of messages with internal

functions that supports all the demands of track and trace. The message broker can be used for handling external and on-site interface integration with EC.



**Figure 3 2: Main steps - incoming messages**

It is important to get a clear idea of what the role of the messaging module is in EC. The purpose of the messaging module is to be the infrastructure required in order to support the business requirements of the different business areas in EC with respect to messaging. The messaging module does not contain business logic and does not interpret or validate messages from a business perspective. The rules for each business process are maintained by the business modules responsible and are not within the scope of the messaging module.

As shown in the total solution, EC is integrated with a message broker. The incoming messages are converted to an internal EC format in the message broker and made available via EC-MHM. Further steps for data preparation and business process execution is a part of the general technical EC framework infrastructure scope, but from an end user point of view the steps displayed are necessary in order to do all steps necessary in order to cover requirements for an incoming message. Thus in this way EC covers the required processing steps.

The steps for outbound messages are in Figure 3 3. Note that the steps in red color are within the scope of the message broker. Executing business processes is not within the EC-MHM scope.

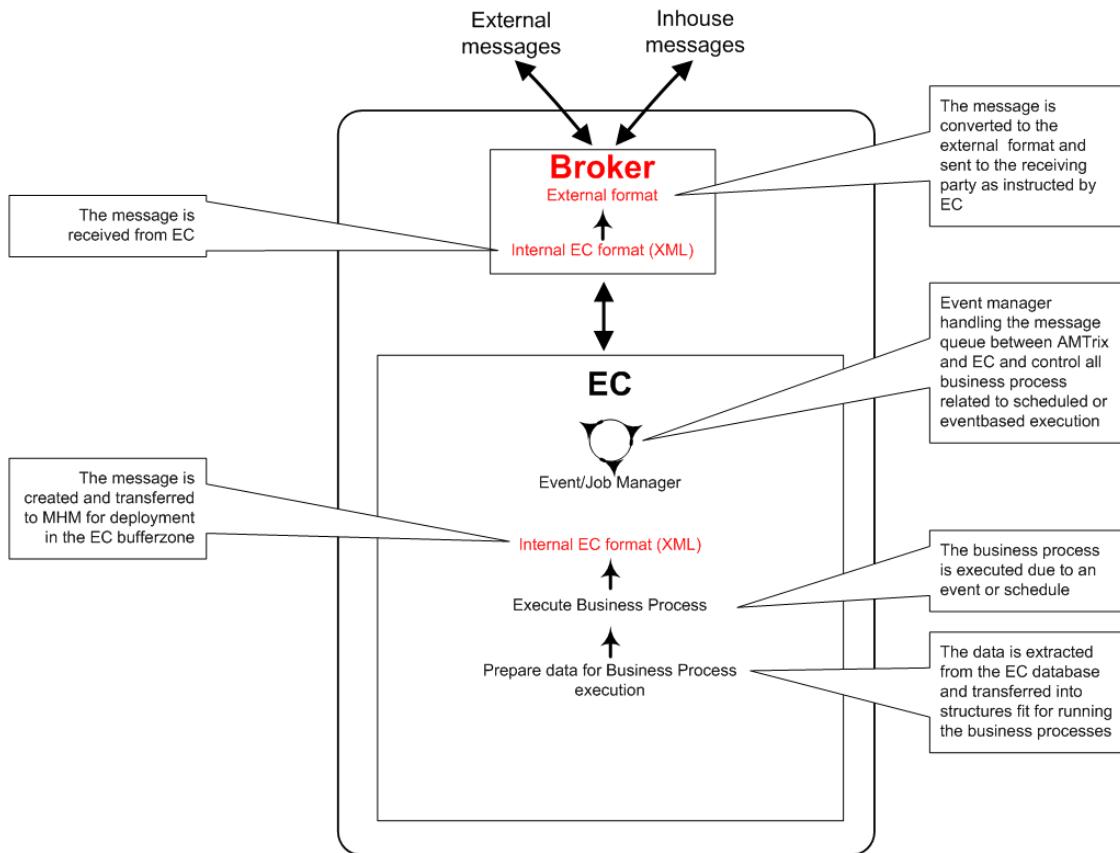


Figure 3 3: Main steps - outbound messages

#### High level description

All message generation and data loading is controlled and executed in EC. The message related master data is located in EC, except master data for message detection used for recognizing messages is in the message broker. The message broker's open architecture enables EC to connect to and interact with different applications on different platforms with varying technical interfaces. The message broker can be used for both internal and external integration purposes.

The message broker handles the logistics of an interchange of information between the external recipients and EC-MHM. All other subsystems do not have an interface with the message broker as they only communicate with EC-MHM as shown in Figure 3 1. When a message enters the message broker a predefined chain of events for specific interchanges are executed.

The message broker handles message conversion/translation between external and internal message formats. EC prefers XML messages tightly connected to the business processes and the message broker converts messages to internal EC formats and formats preferred by external parties. Tools in the message broker for creation of conversion programs are typically graphical and support drag-and-drop technique. The conversion function is able to validate data and make decisions based on the content of a message and what event to perform in the next step of the integration process. The process of converting a message is supervised and logged. It is possible to track and trace a message from the point it enters the MHM sub-system and up to the point where it leaves the sub-system, regardless of what techniques that are used in the process.

There is typically an extensive logging of the activity in the message broker. It is a close interaction between all events and every interface essential for the integration and for the log module used to track and trace the interchange. Every step in the process should be logged in different parts of the log module depending on what interface that is used in the process (see also Logging subsection).

Extensions may be required to the message broker in order to support message archiving functionality. Archived message is available via EC screens (see also Archiving subsection).

The message broker usually provides reporting functions regarding transactions and volume information of the interchanges made.

A message broker usually supports Secure Messaging. Secure messaging is not implemented in the current version of

EC-MHM, but is a part of the available infrastructure. The preferred method is often S/Mime which is a technique used in many market segments and has been adapted widely in the business community, which also supports the usage of PKI among other things.

Within the message broker the search and logging functionality is physically oriented and on a message level. Within EC the messages must be connected to business data with a configurable mechanism to extract and store business related data i.e. network points and shippers in order to enable cross search mechanism on message-business related issues. The connection between the original message and the data loaded into EC should be taken care of by the project specific load jobs or the implementation of the message archive (message broker extension). There is no such mapping within EC-MHM.

### **Functional overview**

The following requirements must be covered by the MHM:

- The message system must offer a set of functions, supplying following functionality to be used by EC's different sub-systems:
  - Send a message.
  - Receive a message / the data in the message.
  - Messages (both sent and received) should be available to the EC sub-systems either on a disk-structure, in an Oracle database table-structure or a (public) java structure.
  - Fetch status of a message.
  - When an EC sub-system sends a message, the message must be regarded as sent by the sub-system itself, meaning that the MHM should do the checking and validation of the message to ensure that the message has the correct parameters set, and then send the message in the correct format/protocol.
  - If message sending fails on both alternative methods a number of retries can be configured before the sending has failed.
- Different options to automate are supported for both incoming and outgoing messages.
  - Manual: Messages are kept in a "Wait position" until manually acknowledged/rejected by end users.
  - Semi-automatic: The first message instance is handled automatically, but all revisions are handled in the same way as "Manual" messages.
  - Automatic: Messages flow through the system without any end user acknowledgment.
- Priority mechanisms controlling message processing both in EC and the message broker.
- The message system must be able to exchange messages of XML-format with EC's different sub-systems and make data available to the business processes.
- It must be possible to re-route messages in the message broker to other satellite systems.

### **Performance**

One of the challenges of having a common system handling all messages is to have service availability to all business modules and minimize side effects for the business processes due to activity in other parts of the system. The message broker may support several message queues with different priorities. The EC-MHM adaptor should fetch and push the messages with respect to these priorities.

Between EC-MHM and the message broker there is a single queue. This queue is only used to transfer messages having the EC internal XML format to the EC database. No processing is done at this stage. However, the order of the messages in the queue is frequently refreshed and the messages having higher priority will progress through the queue faster than the ones with lower priority.

### **Scheduling and event handling**

In order to execute business processes and exchange messages in the EC message broker interface, some process management is needed.

Two business actions are defined in EC to be scheduled on given intervals by the EC scheduler. These are:

- MessagesSend – which transfers the outbound messages to the message broker.
- MessagesRetrieve – which transfers the inbound messages from the message broker to EC.

The messages will be transferred according to priority settings.

The message broker should also process the messages according to priority settings.

### **System scope and interfaces**

The messaging can be used by several EC sub-systems to send and receive messages to/from other parties and applications. The following sections describe the common functionality and interfaces.

### **MHM basic functionality/processes**

MHM provides the technology for exchanging business documents. This function is important in cases where the integration should be event-driven depending on the information in the message.

Business driven integration is the way to handle messages containing information that controls which events should be performed to send the information to the right application or sub-system.

The process for message integration is able to handle events and to look into the messages and from the information point of view decide which events should be triggered.

The level of automation in the message flow in EC is configurable in EC-MHM. Messages can be configured to be manually acknowledged or flow automatically without user intervention.

### Message flow

#### Overview of Message flow and Message statuses

The interaction between the message broker and the EC subsystem is done by the EC-MHM adapter. Both the message broker and EC are fetching and storing messages to the message archive. The message archive will reside in the message broker extension (or the message broker itself if convenient). The message broker will not connect directly to EC. This eliminates a possible security risk and enables architectures with deployment of EC and the message broker in different security zones.

The figure below illustrates the different states a message can have during transport through the system. The messages are available in the EC-MHM end-user screen "Message Journal" in all stages.

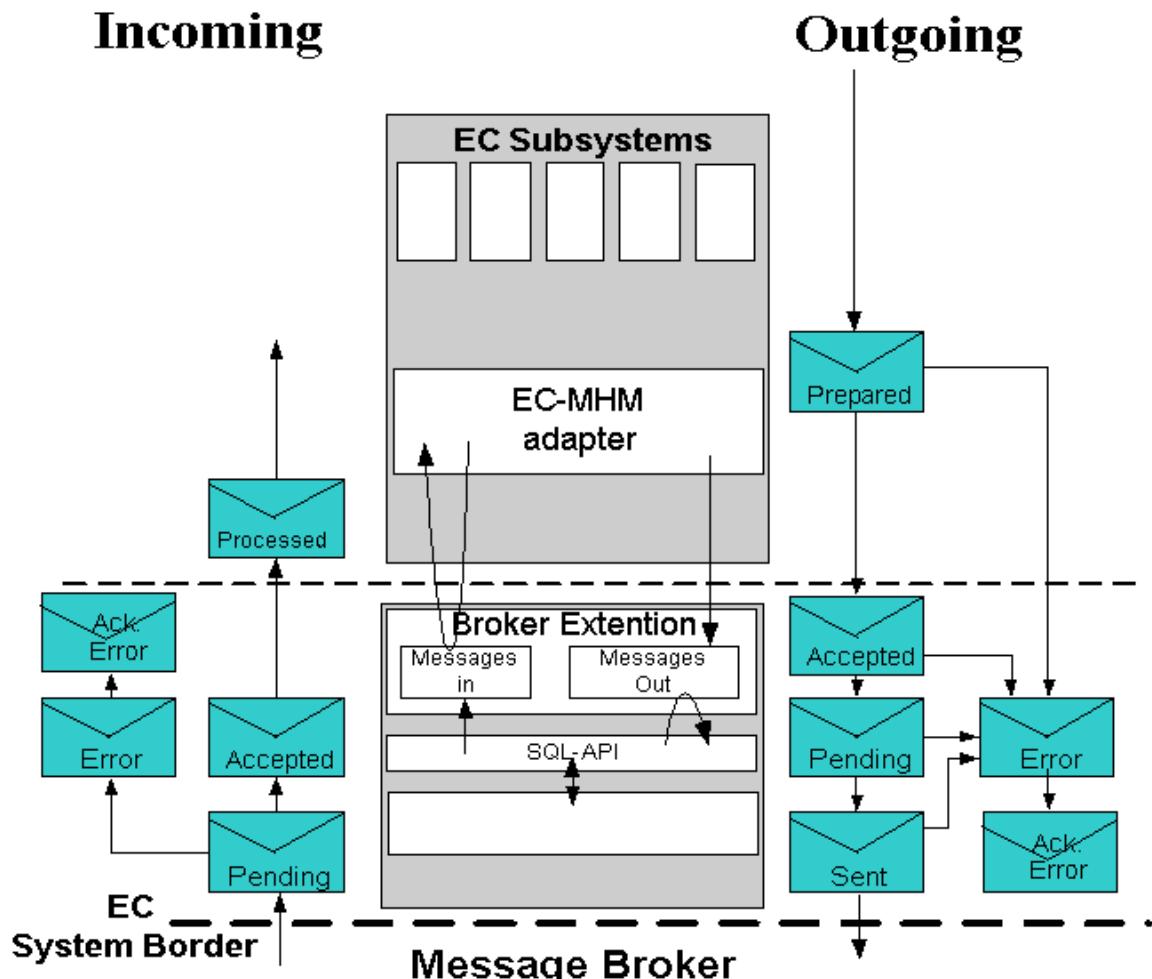


Figure 4 1 Message statuses during transfer

#### Different statuses for outbound messages

- *Prepared*, message is transferred from the EC-MHM queue for outgoing messages and is flagged *prepared*. The message is ready for further processing by the message broker. The EC-MHM queue for

- outgoing messages is described in the "Message Type" section.
- *Accepted*, message is fetched by the message broker. Message processing in broker commences.
  - *Pending*, message is converted correctly and logged. The message is now in queue for transmission on the specified communication protocol. For protocols that support acknowledgement, the pending status might be seen as a state where the message broker is waiting for acknowledgement.
  - *Sent*, message is transferred successfully. Be aware that delivery is only guaranteed for protocols supporting acknowledgments, i.e. X.400. With SMTP as the protocol, "Sent" just means that the message has reached the first SMTP server. It does not guarantee that the message has reached its destination because the SMTP protocol does not support that.
  - *Error*, a problem has occurred in the message broker event chain during message processing or the physical message transferral to the receiver failed. The message log contains information about the problem at hand.
  - *Error Acknowledged* indicates that the message no longer takes any part of EC's interaction with other systems and the message has been manually handled by an end-user.

#### Different statuses for inbound messages

- *Pending*, the message broker has received the message and it's logged in the message archive and in queue for conversion.
- *Accepted*, message is converted correctly and the message was stored correctly in the database table for incoming messages in the message archive. When the message is in status "Accepted" the message broker will release control of the message which means that the message is ready to be loaded into EC. A message in this state is displayed in the "Incoming Messages" screen.
- *Processed*, message is loaded into destination sub-system of EC without errors.
- *Error*, a sub process like e.g. the conversion of the message has failed.
- *Error Acknowledged*, indicates that the message is no longer taking any part in EC's interaction with other systems and the message has been manually handled by an end-user.
- *Not Loaded*, the loading into EC message has failed.
- *Not Loaded Acknowledged*, indicates that the message is no longer taking any part in EC's interaction with other systems and the message has been manually handled by an end-user.

When the EC system has processed the message it will change the status to either "Processed" or "Not Loaded".

#### **EC-MHM queue for outgoing messages**

In order to support the requested functionality in EC a queue for outgoing messages is established in the EC database. This means that the EC modules interact only with EC-MHM which is the layer between EC and the message broker. The purpose is to have a connection into the message broker available for the business modules. The external reference of a message transferred to the message broker is the message id in the queue for outbound messages in EC-MHM. Keeping this reference unbroken from the business module all the way into the message broker enables EC functionality for messaging follow up from screens and business processes.

In addition to containing message references there is support for a simple workflow with possibilities to acknowledge/delete messages, maintain recipients (add/delete) and remarks before transmission of outgoing messages. In order to support this workflow the generated message ids are deployed in the outbound message queue in EC-MHM along with the required information for further processing. These messages are not final and when receiving a reference in the message broker all requests on these messages will be forwarded to the message broker where the messages reside in their official state.

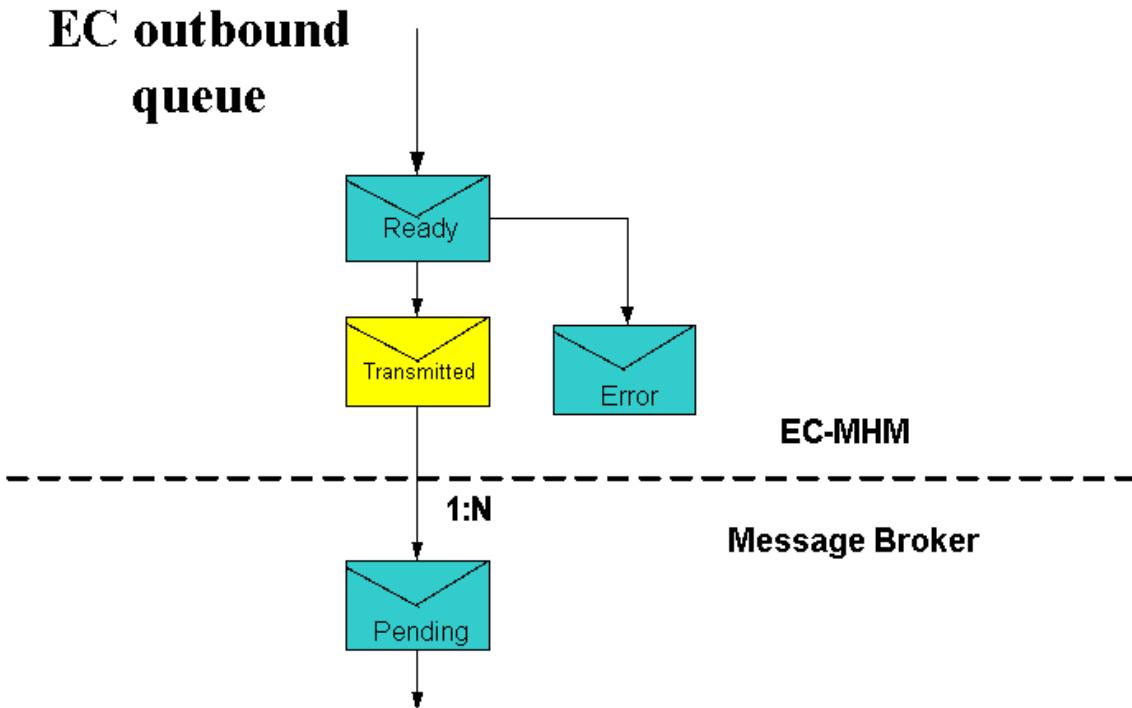
#### References to primary keys in the message broker

There are always issues when coordinating the use of primary keys in a cross-application landscape. Being at the end of a chain isn't very problematic. Therefore referring to an inbound message is quite simple, as a firm non-changing message reference is given to EC from the message broker. Keeping track of outbound messages, however, requires a few more steps. The message id in the message archive should be traceable so the messages can be traced within the message broker.

#### Message status in the EC-MHM queue for outbound messages

Messages approved for transmission are transferred into the message broker. Note that messages in the EC-MHM queue for outbound messages are not defined as sent before they have status "Transmitted". Until that point they still reside in EC.

The figure below illustrates the possible states a message can have in the buffer zone. Note that the end-user can physically delete the message when having status Ready or Error with authorized access to the appropriate screen. Blue colour (darker grey in B/W-printout) means statuses that can be shown in the end-user screen "Outgoing Messages", while yellow means a status that is normally not shown.



**Figure 4 2 Message statuses in the EC-MHM queue for outbound messages**

#### ***EC-MHM outbound messages status description***

Statuses:

- *Ready*, an EC sub-system has generated the message. The message is waiting for a manual accept. In this state it is possible to physically delete the message. If a message is configured for automatic handling it will not appear in any EC-MHM screen and the transfer to the message broker will start automatically.
- *Transmitted*, the message is successfully transferred into the message queue in the message broker. Transmitted messages are defined as sent, and cannot be deleted.
- *Error*, a transferral into the message queue in the message broker has failed. In this state it is possible to physically delete the message

As shown in the figure above the message is initiated in the message broker with status "Prepared".

#### ***Format conversion/translation***

Conversion/translation of messages is an important function in the message broker. This function should preferably support most of the format standards there are on the market. All messages are handled in XML format in EC. This means the message broker converts whatever format is received to an internal XML format protecting EC for complexity and changes in messages.

There are a number of issues that are important for this module, like:

Industry standards: Support of the following standards: EDI (EdiFact, EdiGas), XML, HTML.

Event-driven messaging: Possibility to act, depending on the information in the message, according to predefined rules in the next step of the process of the message handling, i.e. priority issues.

The message broker handles conversion between different formats and to and from these formats:

- XML: Extended Markup Language, for both external and internal use.
- EdiFact/EdiGas: UN/EDIFACT (United Nations Rules for Electronic Data Interchange for Administration, Commerce and Transport).
- Formatted text.

#### ***The message broker process description***

The message broker must be able to handle different processes depending on the different procedures configured to happen in every step in processing a message. The purpose of the message broker is to act as an interface between the

EC application sub-systems and the external actors. Logistics regarding the messages that are sent out and received are to be handled with these internal message broker processes. The major processes that are pointed out as important for the handling of messages are described below.

#### **Logic for outbound messages**

Logic for outbound messages covers the process (interchange) from the point where a message has been received from an application to the point where the message is sent to the receiving part. Receiving of a message from the application will be done as a database entry.

The message is captured in a handshake operation with EC. When loaded into the message broker, the process commences and the receiver of the message is identified and sufficient address information is added to the process.

If there is a need for conversion this step will be processed. The three formats that are usually supported: EdiFact, Formatted Text and XML.

In the conversion step there must be a function for syntax validation of the outbound message. This is a quality issue and proves in a technical way the converted message is correct in relation to the implementation guidelines and can be technically correctly handled by the receiving part in their conversion module.

Next step will be enveloping and sending the message to the chosen part. The message broker must be able to keep track of where in the interchange the message has reached by use of a status.

The whole "outbound message" process will be logged for the possibility of tracking and tracing the interchange. The possibility to monitor the interchange from beginning to end with tools for looking into each message and to store the information sources must be covered.

All messages may be configured with primary and secondary addresses. If problems occur during transmission the secondary address is applied. If none of the addresses work the transmission is logged as an error. The user may then manually try to retransmit the message and choose other alternative addresses.

It is important that one message may be converted into several formats and sent to different companies using different protocols and connection types.

#### **Logic when using attachments for outbound messages**

Attachments are used for distributing reports. When the original message is converted and is in the SMTP part of the event chain, the message broker will add the original message as the body part in the mail message and add any attachments. When adding attachments to outbound messages these will be transferred along with the message. Multiple attachments may be supported by the message archive. Attachments are not supported for inbound messages.

#### **Message Archive**

The message archive in the message broker extension supplies a way to store the messages and log the transfer, to give possibility to search for or view/edit message(s). Both original and converted messages will be stored.

How to purge the message archive has to be implemented by the projects (see also Logging and Archiving).

#### **Communication Control**

The message broker handles the communication layer and have the responsibility to do the actual message transfer to the receiving part/server using different kinds of formats and protocols (e.g. initiate a call to the receiving server on a designated line – preparing a connection to a part via an ADMD service, directly over leased lines to a part or via internet SMTP server). It is also convenient to configure the message broker to use an alternative method if the primary choice has failed, which is automatically used if the primary method fails. As an example the MHM system will primary try to send it by SMTP to a certain address and if that fails it will try to send it over X.400 to an X.400 address.

When posting a message to the message broker EC will supply delivery methods and addresses to the message broker. The list of delivery methods are sorted by preference. The message broker tries the first delivery method and tries the next in the list if it fails. Note that not all delivery methods acknowledge successful delivery. The message broker should be able to handle every level of acknowledgement that comes back on different protocols.

#### **Logic for Inbound messages**

Logic for the inbound messages covers the process from the point when the message broker is contacted via one of the communication links and the message has been made accessible to EC.

The communication interface is responsible for handling messages as they arrive and detect which process is to be started. In this detection process the message format and receiving sub-system is identified. The processing of the different messages must therefore contain rules to make it possible to route each message in the correct direction towards the sub-system.

In the process there is a chain of events to handle the message the proper way and deliver the message to the right application. The messages received must be converted from the external format (typically the formats EDI, Formatted Text and XML should be supported) to the internal EC XML-format. The whole process is logged and the message is stored in the log module.

### **Confirmed messaging**

In cases where communication protocol acknowledgement is insufficient confirmed messaging is required. As we have chosen to minimize the logic in the message broker this kind of functional confirmation must be handled in EC.

Typically a business process is started up to generate a message. A business process in EC should be applied to keep track of the confirmations. Timeline, events and actions required to handle the confirmation process is specified and handled at this level. This covers the point-to-point channel scenario. Thus, only looking at the information within the message broker will not cover this. In cases with these requirements an alert screen in EC should be available to users for handling the issues that might arise, and extensions to the EC buffer zone tables in order to keep track of confirmations. Note that the EC standard XML header for internal messages contains a tag for reference information.

### **Logging**

Tracing a message transfer through the MHM means that you will follow the transfer from the point where the message enters MHM and up to the point where it leaves MHM. The EC system should be able to view the log of a message through the EC-MHM adapter.

The configuration of the log modules in a message broker and what is to be logged gives you a possibility to choose the level of information you desire to trace the message transfer. This means that you can, as an alternative, log every single step of the process of a message transfer. A lot of log items will then be created per transfer and it is not recommended in a production environment to log every message transfers in this manner. Message transfers that are stable and has been in production for a period should be configured to log information that is necessary to handle a day-to-day production in a safe manner.

Big log files affect the performance of viewing the logs in a negative way. To handle this it is necessary to keep the log files in a day-to-day operation suitable size by storing log files in an automated way.

### **Viewing Log Files**

The message broker can have a graphical user interface for viewing different logs within the message broker. Several log files are continuously receiving information from different parts of the message broker. The information is primarily error information from communication software and other running processes.

When you view the different log files you can configure what information you will access. You can select what information to look at by different filters methods.

Log information related to issues that are interesting for end-users, like conversion problems or violated business rules when validating/loading the data into EC, is available via EC screens. The message broker is responsible for adding this information to the log made available in the message broker extension.

### **Archiving**

Since the messages may be purged from EC and the message broker, there should be a policy for each project for purging the message archive implemented in the message broker extension. Then important messages can be inspected later if requested.

The message archive can be view by using screens in EC.

There will also be possibilities to view earlier versions of the same message, which is an earlier message from which the actual message is created (resent). This is possible due to the fact that the message broker is logging/archiving that info.

## **Configuration**

### **EC master data**

The main strategy is to keep the majority of master data in EC making it available in EC end-user screens. In this section the parameters controlling the message flow is explained. Further control of master data related to e.g. contact information and distribution list, is documented in specific MHM business functions.

### **Functional Area**

The functional areas make it possible for user roles to have individual access to contact group set, contact group, message contact, etc.

## Message Type

We have a set of parameters which defines message behaviour. These are related to what is called a message type and a message sub type. A message type in this context is a unique internal XML definition in EC. The message sub type is present only in the XML and the load job can make different use of this, e.g. nomination and re-nomination.

### Message type configuration setting

Parameter	Comment
FUNCTIONAL_AREA	The EC module which owns this message, alternatively a reference to the system as such.
TYPE	The identification of this message.
NAME	The full name of the message.
SUBJECT	A subject for the message displayed in the message journal where the system can dynamically replace "variables" in the subject by actual business values i.e. "Confirmation for \$shipper\$" is displayed as "Confirmation for BP".
HANDLING	Specification of the automation of the message (Manual, Semi-Automatic, Automatic).
LOADING_JOB	The full name of the java class used for loading the inbound message. All message types of inbound messages should have a loading job defined. For messages which are meant to be read only and manually acknowledged this class should be used: com.ec.frmw.mhm.model.ejb.inbound.LoadReadOnlyMessage
GENERATING_JOB	The full name of the java class used for generating the message.
INTERNAL_FORMAT	Internal format of the message.
DIRECTION	Indicates the direction of this message (In, Out, Both).
FREQUENCY	The frequency of the message (Daily, Weekly, Monthly, Event based).
SCHEMA_URL	Refers to the location of the schema which define this specific internal XML.
FORMAT_TYPE	Tells whether these messages have XML or text in the body tag or if the message actually is some kind of attachment to be distributed.
DEFAULT_SENDER	Default sender of this message type.

### EC XML Standard message header

A common message header is applied for all internal XML messages exchanged between EC and AMTrix. The header fields are described below and the XSD for the header is available as well.

Element	Attribute	Comment
Msg	Type	This is the identification of the message type.
	SubType	Further specification of the message type.

	System	This is the system generating the message which might be EC or some satellite system if sharing the message broker.
	Revision	Identify the message revision and is a business oriented value.
	Priority	This indicates a message priority category and is applied when setting the execution sequence of business processes in EC.
	TestFlag	Indicates if it is a test or production message. Valid values are (Y,N).
	ReplyOnReference	If this message refers to another message this is the technical way of keeping integrity within the message.
RecipientList	TO	A semicolon separated list of receivers.
	CC	A semicolon separated list of receivers getting a carbon copy.
Sender	CompanyNumber	The value of the sender company code as configured in EC master data.
	CompanyName	The value of the sender company name as configured in EC master data.
	ContactCode	The value of the sender contact code as configured in EC master data.
	ContactName	The value of the sender contact name as configured in EC master data.
	Ident	The value of the sender main EDI address as configured in EC master data.
	SubIdent	The value of the sender sub-EDI address as configured in EC master data.
Receiver	CompanyNumber	The value of the receiver company code as configured in EC master data.
	CompanyName	The value of the receiver company name as configured in EC master data.
	ContactCode	The value of the receiver contact code as configured in EC master data.
	ContactName	The value of the receiver contact name as configured in EC master data.
	Ident	The value of the receiver main EDI address as configured in EC master data.
	SubIdent	The value of the receiver sub-EDI address as configured in EC master data.

Generated	Date	The day and time of the generation.
	GeneratedBy	The user generating the message. If system generated it refers to the system.
Subject		An appropriate subject in order to have a user friendly string in the message journal.
Period	FromDate	Start of the given period. This value relates to the business content of the message.
	ToDate	End of the given period. This value relates to the business content of the message.

```

<?xml version="1.0" encoding="ISO8859-1"?>
<!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com) by ASA
(TE) -->
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Lars Petter
Bjørgen (Energy Components) -->
<!--W3C Schema generated by XML Spy v4.2 U (http://www.xmlspy.com)-->
<xsschema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xss:element name="MsgList">
<xss:complexType>
<xss:sequence>
<xss:element name="Msg">
<xss:complexType>
<xss:sequence>
<xss:element name="Header">
<xss:complexType>
<xss:sequence>
<xss:element name="RecipientList">
<xss:complexType>
<xss:attribute name="TO" type="xs:string" use="required"/>
<xss:attribute name="CC" type="xs:string" use="optional"/>
</xss:complexType>
</xss:element>
<xss:element name="Sender">
<xss:complexType>
<xss:attribute name="CompanyNumber" type="xs:string"
use="optional"/>
<xss:attribute name="CompanyName" type="xs:string"
use="required"/>
<xss:attribute name="ContactCode" type="xs:string"
use="optional"/>
<xss:attribute name="ContactName" type="xs:string"
use="required"/>
<xss:attribute name="Ident" type="xs:string"
use="optional"/>
<xss:attribute name="SubIdent" type="xs:string"
use="optional"/>
</xss:complexType>
</xss:element>
<xss:element name="Receiver">
<xss:complexType>
```

```
        <xs:attribute name="CompanyNumber" type="xs:string"
use="optional"/>
        <xs:attribute name="CompanyName" type="xs:string"
use="required"/>
        <xs:attribute name="ContactCode" type="xs:string"
use="optional"/>
        <xs:attribute name="ContactName" type="xs:string"
use="required"/>
        <xs:attribute name="Ident" type="xs:string"
use="optional"/>
        <xs:attribute name="SubIdent" type="xs:string"
use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="Generated">
    <xs:complexType>
        <xs:attribute name="Date" type="xs:dateTime"
use="required"/>
        <xs:attribute name="GeneratedBy" type="xs:string"
use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="Subject" type="xs:string" minOccurs="0"/>
<xs:element name="Period">
    <xs:complexType>
        <xs:attribute name="FromDate" type="xs:dateTime"
use="required"/>
        <xs:attribute name="ToDate" type="xs:dateTime"
use="required"/>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="Type" type="xs:string" use="required"/>
<xs:attribute name="SubType" type="xs:string" use="optional"/>
<xs:attribute name="System" type="xs:string" use="required"/>
<xs:attribute name="Revision" type="xs:nonNegativeInteger"
use="required"/>
        <xs:attribute name="Priority" type="xs:nonNegativeInteger"
use="required"/>
        <xs:attribute name="TestFlag" type="xs:string" use="required"/>
        <xs:attribute name="ReplyOnReference" type="xs:string"
use="optional"/>
    </xs:complexType>
</xs:element>
</xs:sequence>
```

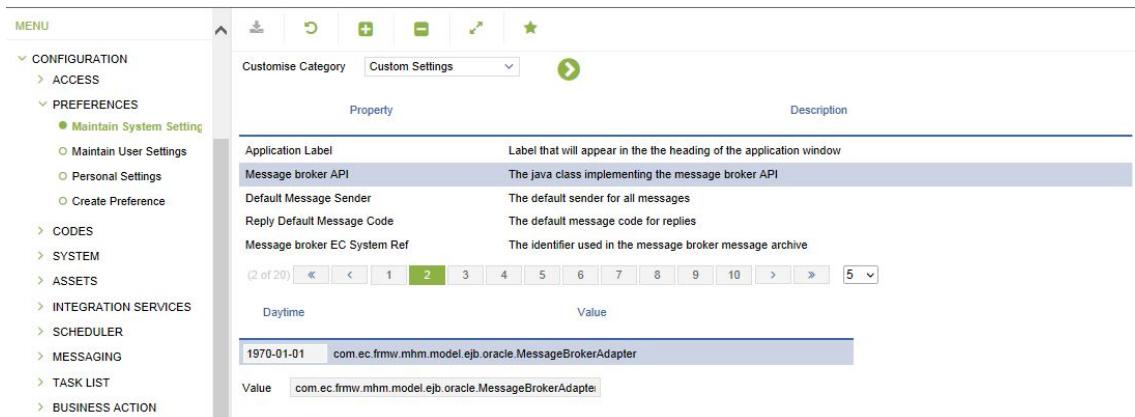
```
</xs:complexType>
</xs:element>
</xs:schema>
```

### **EC XML Body definition**

The body of the message will vary with each message type. Each message type has its own format, which holds all of the data from the original message. Each message type can of course be translated to different outgoing formats like XML, EDI and formatted text. And the other way; the same message type can have 20 different layouts as formatted text messages from 20 different companies, but the MHM will translate all these messages to one single internal XML format. The use of only one format makes it much easier to implement the program handling that specific message type. Only one kind of XML-format needs to be interpreted.

### **The EC-MHM adapter**

EC-MHM does not come with any adapter for connecting to a message broker. When a project has implemented an adapter this has to be installed on the EC Wildfly application server. Then you must configure the system property telling EC-MHM which adapter to use.



**Figure 5 1 Setting the system property telling which EC-MHM adapter to use**

### **The scheduled jobs for posting and retrieving messages**

This section describes step by step how to configure the jobs posting messages to and retrieving the messages from the message broker.

#### **Create a functional area for the EC-MHM administration**

Create a functional area for the EC-MHM administration by using the screen: Configuration \ Assets \ Basic Objects \ Create Functional Area. Remember to restrict the access by using the screen: Configuration \ Access \ Object Partition.

Area Code	EC-MHM_ADMIN
Area Name	EC-MHM Administration
Start date	1900-01-01
End date	

**Figure 5 2 Creating the functional area for the EC-MHM administration**

#### **Create a scheduled job for transferring inbound and outbound messages**

The EC scheduler is used to periodically exchange messages between EC and the message broker. There are two predefined business actions to use:

- MessageRetrieve – for inbound messages.
- MessageSend – for outbound messages.

These business actions will act on all messages ready to transfer so we should check the option "Ignore Misfires".

Since these actions will be scheduled to run frequently the "Retain Count" should be set to avoid flooding the history log of the scheduler.

The job should also be set to be stateful avoiding more than one job processing a message.

These business actions do not have any parameters.

The following screenshots shows how to create and setup the scheduled job:

The screenshot displays the Tieto Business Application Platform's configuration interface. On the left, a navigation menu is visible under the 'SCHEDULER' section, specifically the 'Business Actions' subsection. In the center, a 'SCHEDULE' table lists various scheduled tasks. One task, 'EC\_MHM Inbound', is highlighted with a red border. This task is named 'Tranferring Inbound Messages', belongs to the 'MHM13\_Production' functional area, and is currently in 'WAITING' status. Its next fire time is set for '2014-12-08 1'. Below the table, a pagination control shows page 2 of 5. At the bottom of the main panel, there are tabs for 'DETAILS', 'BUSINESS ACTION', 'SCHEDULE', and 'MONITOR', with 'BUSINESS ACTION' being the active tab. Under 'BUSINESS ACTION', a table shows a single entry: 'MessagesRetrieve' with the description 'Retrieves the incoming messages available from the broker', sequence number 1, and an unchecked 'Isolate Transaction' checkbox. To the right of this, under 'PARAMETERS' and 'FIXED PARAMETERS', both sections show 'No records found.' The overall interface has a clean, modern design with a light blue header and a white background.

**Figure 5 3: Creating a job for inbound messages**

SEARCH

FAVORITES

MENU

- ✓ CONFIGURATION
  - > ACCESS
  - > PREFERENCES
  - > CODES
  - > SYSTEM
  - > ASSETS
  - > INTEGRATION SERVICES
  - > SCHEDULER
    - Business Actions
    - Schedules
    - Schedule History
    - Manage Scheduler
  - > MESSAGING
  - > TASK LIST
  - > BUSINESS ACTION
- > EC PRODUCTION
- > EC SALES
- > EC REVENUE
- > REPORTING
- > PROCESS AUTOMATION
- > MESSAGING
- > TASK LIST
- > EC INTEGRATION SERVICE

SCHEDULE

Name	Description	Functional Area	Status	Pinned To	Next Fire Time	Enabled
EC_MHM Inbound	Transferring Inbound Messages	MHM13_Production	WAITING		2014-12-08 1	<input type="checkbox"/>
EC_MHM Outbound	Transferring Outbound Messages	MHM13_Production	WAITING		2014-12-08 1	<input checked="" type="checkbox"/>
ERPDocProcess	Processing documents at levels business unit, contract area	Revenue	EXPIRED			<input type="checkbox"/>
EXCEL_IMPORT	Excel Import	EC	EXPIRED	WS003040.U		<input checked="" type="checkbox"/>
ForecastCalculation	Forecast Calculation	Allocation	EXPIRED	WS003040.U		<input checked="" type="checkbox"/>
ForecastDailySaleCalculation	Forecast Daily Sale Calculation	Allocation	EXPIRED	WS002994.g		<input checked="" type="checkbox"/>
GenReverseDocuments	Reversal of accruals: Generate a real reversal document	Revenue	EXPIRED			<input type="checkbox"/>
HUMAN_TASK	One time schedule of JBPM process	EC	EXPIRED	WS003040.U		<input checked="" type="checkbox"/>
HUMAN_TASK	One time schedule of JBPM process	EC	EXPIRED	WL2006560.J		<input checked="" type="checkbox"/>
ImportVOQuantities	Import of VO Quantities	Revenue	EXPIRED			<input type="checkbox"/>
InitiateDay	Initial Day	EC	EXPIRED	WS003040.U		<input checked="" type="checkbox"/>
InstantiateDayVO	Instantiate Daily Volumes	Revenue	EXPIRED	WS003040.U		<input checked="" type="checkbox"/>
InstantiateDaysVO	Instantiate Volumes for a number of days	Revenue	EXPIRED	WS003040.U		<input checked="" type="checkbox"/>
InstantiateMthVO	Instantiate Monthly Volumes	Revenue	EXPIRED	WS003040.U		<input checked="" type="checkbox"/>
InstantiateNextDayVO	Instantiate Next Days Volumes	Revenue	EXPIRED	WS003040.U		<input checked="" type="checkbox"/>

(2 of 5) << < 1 2 3 4 5 > >> 15 ▾

DETAILS BUSINESS ACTION SCHEDULE MONITOR

BUSINESS ACTION

Action	Description	Sequence#	Isolate Transaction
MessagesSend	Transfer the outbound message from EC to the broker	1	<input type="checkbox"/>

PARAMETERS FIXED PARAMETERS

Name	Value	Name	Value
No records found.			

No records found.

Figure 5 4: Creating a job for outbound messages.

✓ CONFIGURATION
 

- > ACCESS
- > PREFERENCES
- > CODES
- > SYSTEM
- > ASSETS
- > INTEGRATION SERVICES
- > SCHEDULER
  - Business Actions
  - Schedules
  - Schedule History
  - Manage Scheduler

NODES

Name	Current State	Startup State	Startup Time	Max Threads	Available Threads	Job Counter	System Time Offset
WS003040.Ur	RUNNING	--	2014-12-04 14 10	10	97	- 00:03:03	

STANDBY RESUME

RUNNING JOBS

Name	Group	Fire Time
No records found.		

Figure 5 5: Checking for nodes serving the scheduled jobs.

Task Name	Description	Status	Duration	Executed by	Last Run Date
DailyDispatchingCalculation	Daily Dispatching Calculation	Allocation	EXPIRED	WS003040 U	2014-12-08 00:00:00
DailyProductionAllocation	Daily Production Allocation	Allocation	EXPIRED	WS003040 U	2014-12-08 00:00:00
DailySaleCalculation	Daily Sale Calculation	Allocation	EXPIRED	WS002994 g	2014-12-08 00:00:00
DocProcess	Processing documents at levels business unit, contract area	Revenue	EXPIRED	WS003040 U	2014-12-08 00:00:00
DocProcessDelLog	Deletes all log related to document processing	Revenue	EXPIRED		2014-12-08 00:00:00
EC_MHM Inbound	Transferring Inbound Messages	MHM13_Production	MISFIRE	WS003040 U	2014-12-08 00:00:00
EC_MHM Outbound	Transferring Outbound Messages	MHM13_Production	WAITING		2014-12-08 00:00:00
ERPDocProcess	Processing documents at levels business unit, contract area	Revenue	EXPIRED		2014-12-08 00:00:00
EXCEL_IMPORT	Excel Import	EC	EXPIRED	WS003040 U	2014-12-08 00:00:00
ForecastCalculation	Forecast Calculation	Allocation	EXPIRED	WS003040 U	2014-12-08 00:00:00

**Figure 5 6: Checking the history log of the scheduler**

### The message broker

With respect to master data the configuration is limited to:

- Maintaining the library of valid message conversions (mappings) implemented.
- Maintaining the detection filter for recognizing messages in order to assign messages to EC message types.

In addition to these there are of course technical settings configuring the way communication protocols should be used, archiving settings, etc.

### Supporting multiple message versions

From reading the description of where configuration data is located it might still be difficult to see how multi message version is supported.

The message type is configured in EC. If different versions of a message require different business processing, and data content differ to such a degree that a new internal XML is required, this will result in a set of message sub types configured in EC.

If the business rules and data content can be united in one message type this is preferred. In most cases this is the result. The issue of handling different ways of presenting this to the other party is handled by the message broker. Thus, in the message broker it may be possible to configure how to apply mappings on a business partner level and EC can make references to business rules on the same level connecting business partners to different message sub-types and business processes.

### Wildfly configuration

EC-MHM inbound message handling uses JMS and requires a few configuration steps on the Wildfly server.

#### **Configure parallel processing of inbound messages**

The MessagesRetrieve business action enqueues all inbound messages on a JMS queue called ECMHMIInboundQueue. Message driven beans handle messages arriving in this queue in parallel, as they arrive. The beans use the EC message broker to load the message, which in turn instantiates and executes the configured loading job for the message type. The maximum number of beans that can work in parallel, effectively the number of messages that can be processed simultaneously, is configurable. In the WildFly configuration file (domain.xml), a system property named com.ec.frmw.mhm.inbound.queue.max.mdb.pool.size is defined to limit the maximum number of messages that can be processed in parallel by the message driven beans. The value of the property is defaulted to 15. To change the value, configure the property using Wildfly CLI, as follows:

- Go to the <wildfly>\bin folder. Run jboss-cli.bat in the command line.
- Type **connect** and hit *Enter*.
- Set the value through running the following command (note that we are now increasing the value to 20):

```
/system-property=com.ec.frmw.mhm.inbound.queue.max.mdb.pool.size:
write-attribute(name=value,value=20)
```

You should have an output that looks similar to the image below:

```
[domain@localhost:9990 /] connect
[domain@localhost:9990 /] /system-property=com.ec.frmw.mhm.inbound.queue.max.mdb.pool.size:write-attribute(name=value,value=20)
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => undefined
}
[domain@localhost:9990 /]
```

## Message Broker Interfaces (AMTrax as an example)

The interfaces to other systems are taken care of by the message broker. In this chapter the message broker *AMTrax* is used as an example.

The message broker provides the other sub-systems in EC with the possibility to exchange data in standardized message formats, regardless of physical distance and hardware platforms of the (external) recipients. In order to enable this application messaging, the message broker must perform several vital functions where interfaces have a central role in the interchange of information between the message broker and the different actors, as well as different internal functions within EC.

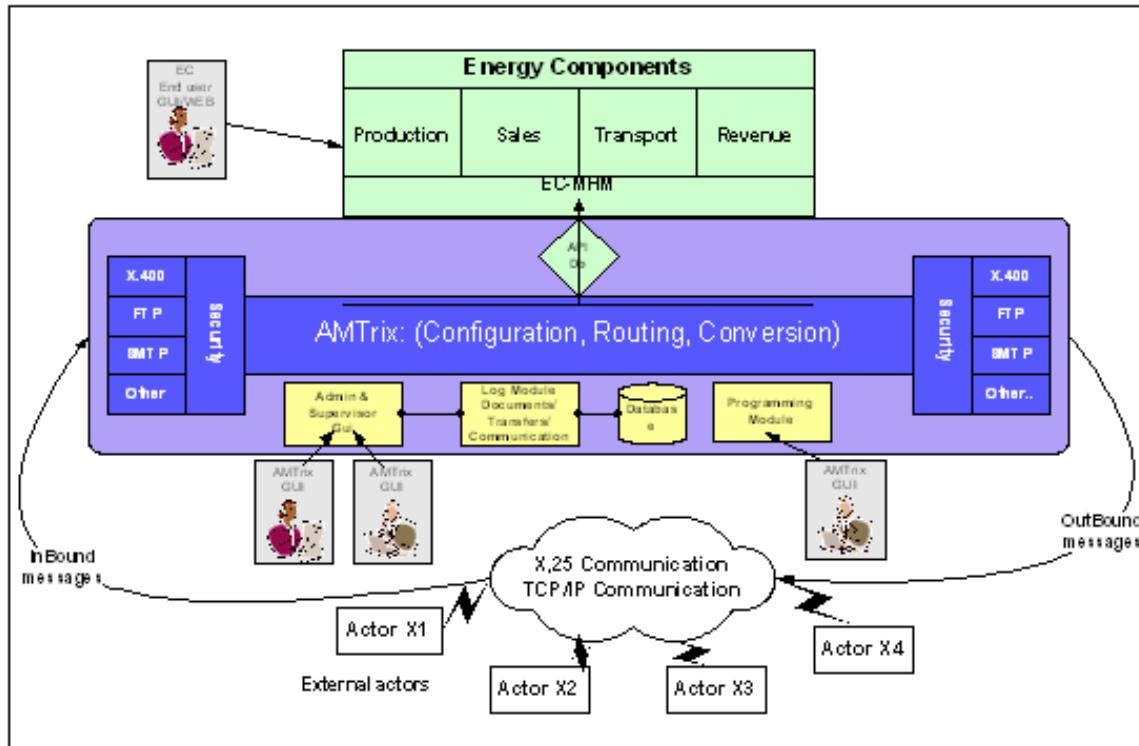


Figure 61 A typical architecture using AMTrax as the message broker.

### **Application interfaces - Connectors**

Application interfaces or connectors are a complete solution package that is used for an efficient implementation and integration of specific applications into a messaging infrastructure. Pre-fabricated connectors are available to some of the most commonly used applications on the market. This makes AMTrax a powerful tool, since it provides a very "tight" and well defined way for AMTrax to interact with those applications with a minimum of programming. The purpose is basically

to facilitate application integration and thus make it possible for internal and external applications to exchange information.

Application integration is really about three things:

1. Data conversion – making it possible for applications to understand each other's data.
2. Access mechanisms – making it possible for applications to access each other's data.
3. Activation and transaction mechanisms – making it possible for applications to activate each other.

### ***Application Programming Interfaces (API)***

If there is no pre-fabricated connector available, but the application itself provides an API which defines how to connect and interact with that application in a very "tight" manner, it's possible to create your own connector. This can be done with the available tool in AMTrix.

### ***File Based Interfaces***

File Interfaces is a very common way of doing application integrations, mostly because the easy implementation of it and the availability of file-structures and file sharing- or file- transfer protocols on both MHM and application side. AMTrix handles this and no modification of the application is needed. It's the way AMTrix interacts with its internal communication modules. This interface is included even if it is not used as the primary interface method between EC sub-systems and AMTrix (see next section). The basis for this technique is normally FTP or disk sharing.

### ***Database interfaces***

Database interfaces give AMTrix access to the most commonly used databases and provides database programming tools like SQL/ODBC. Database interfaces built on SQL supports the standards there are on the market regarding SQL. EC can therefore interact with AMTrix installed on an Oracle database. This technology supports connectivity in a very standardized but also flexible way. EC-MHM uses SQL-APIs to make messaging functions, like send and receive, easily accessible to all Sub-systems.

See the Technical Appendix for more information about this.

### ***Development functions***

AMTrix includes several different functions for development. For example the connector development tool mentioned above. There are drag-and-drop tools available which simplify the management of traditional EDI documents as well as the "anything-to-anything" flow of messages and helps to create conversion programs to move data from one data file format to another.

### ***Communication interfaces***

Communication interfaces provide AMTrix's communication with internal and external parties as well as support interaction with the local EC system. The EC system is usually installed on other hardware components and SQL connections are used for the interaction between AMTrix and EC. AMTrix includes different communication protocols like TCP/IP, X25, SMTP, X.400 and FAX (note: only outgoing and provided via SMTP). Other communication types like HTTPS, HTTP, FTP, OFTP and AS2 can also be supported. The x.400 interface and its configuration makes it possible for an existing organization to keep their existing X.400 addressing.

The table below has some more protocol detailing.

Protocol	Transport	Comment
X.400	X.25	X.420 & X.435
X.400	TCP/IP	X.420 & X.435
SMTP	TCP/IP	
FAX	Fax, only outgoing	Via SMTP-Mail
TELEX	Telex, only outgoing	Via SMTP-Mail
FTP	TCP/IP	FTP using ISDN and for internal integration using in-house infrastructure
OFTP	TCP/IP	FTP using ISDN and for internal integration using in-house infrastructure
HTTP	TCP/IP	
HTTPS	TCP/IP	
AS2	TCP/IP	HTTP and MIME

AMTRix is constructed in a modular way. The communication interfaces that interact with the "core process engine" (central message distribution engine) at an "upper level" and with the communication programs at the lower levels generate logging/transfer reports during the progress of the transfer.

See Technical Appendix for more information regarding Communication interfaces and their usage.

#### **SMTP Communication or E-MAIL function**

The E-Mail communication method enables AMTRix to use Internet email when sending and receiving EDI messages, formatted text or XML. It supports both incoming and outgoing messages as well as MIME extensions (S/MIME via the Security Interface/Module). The EMAIL communication is based on the following two protocols:

- SMTP (Simple Mail Transfer Protocol) to send messages via a Mail Server that provides SMTP.
- POP3 (Post Office Protocol, Version 3) to retrieve messages (poll messages) via a Mail Server that provides polling via POP3.

See Technical Appendix for more information about E-Mail function.

#### **Security interface**

Security interface provides a platform for Secure Messaging, meaning handling the security of messages leaving and coming into AMTRix, preferable by using S/MIME. S/MIME provides among other things encryption, which is needed to be able to keep the privacy of a message transport employed over a wide area network or over the open Internet. S/MIME supports PKI and use of catalogues. AS2 is supported by AMTRix.

#### **User interfaces, administration and supervision**

*End-User* and *Administrative and Supervision* interfaces are two different types of graphical user interfaces which both are easily adapted into modern workstation environments.

##### **End-User interface**

The WEB based interfaces in EC-MHM provides possibility for personnel in day-to-day operations to check their own transfers of documents, as well as status of incoming messages (as they arrive to the MHM). There is a possibility to search for messages, see lists of different views of messages (in/out/pending), each message status, edit (incoming) messages for manual adjustments, re-activate transformation of messages, etc. This part of the day-to-day interface is just another sub-system to EC and is seen as such from the end users view, even though some functionality will reside in another hardware platform.

##### **Security for the End-User interface**

The end user interface for MHM screens is handled the same way as all other screens in EC. Thus, security and access

issues are handled in a consistent manner with EC.

#### **Administrative & Supervision interface**

These interfaces are intended for the AMTrix administrator. They will be used for administration and configuration of MHM and for supervision purposes. This Graphical User Interface is a client-server solution, preferably very "light" client software, where the logic resides on the server side making it easy to manage. It should also make it possible to achieve the necessary security requirements.

#### ***Security for Administration and Supervision Interface***

There's no additional security checking provided in the GUI for administration and supervision from the one included in the system, which is based upon user id/passwords and different levels of administrative rights. There are of course possibilities to add security via the Operating System, such as stricter TCP/IP checking and/or running connection calls via a security (proxy) server.

## **Appendix**

### **List of EC-MHM Business functions**

This section lists the business functions available in EC-MHM. Detailed descriptions can be found in "EC Framework - Business Functions."

- Maintain Contact Group Set
- Maintain Message Type
- Message Format
- Freetext Message Template
- Actor Maintenance
- EDI Address
- Value Mapping Maintenance
- Distribution List
- Send Freetext Message
- Communication Status
- Message Alerts
- Message Distribution
- Outgoing Messages
- Incoming Messages
- Message Journal
- Message Workflow Overview
- Message Generation

### **Implementation of the message archive**

Message archive of the message broker extension is implemented as database tables and has been adopted into EC.

#### **MHM\_MSG**

The MHM\_MSG table is the main table to store messages in. MESSAGE\_ID is the unique key of a message. ATTACHMENT\_ID is the foreign key which refers to the message attachment table. This means one message can only have one attachment.

COLUMN	TYPE	Comment
DIRECTION	VARCHAR2(1)	Not Null, I/O, inbound or outbound message
EC_SYSTEM_REF	VARCHAR2(32)	Not Null, System reference
MESSAGE_ID	NUMBER	Not Null, Unique ID of message
EXTERNAL_REF	VARCHAR2(64)	External reference
MB_REF	VARCHAR2(64)	Message broker reference
PRIORITY	NUMBER	Priority of message
STATUS	VARCHAR2(32)	Not Null, Message status
RECEIVED_DATE	DATE	Not Null, Received date
MSG_TYPE	VARCHAR2(32)	Message type
SUBJECT	VARCHAR2(80)	Subject
SENDER	VARCHAR2(64)	Sender
RECIPIENT	VARCHAR2(64)	Recipient
VALID_FROM	DATE	Not Null, Valid from date
VALID_TO	DATE	Not Null, Valid to date
REVISION	NUMBER	Not Null, Revision
ACKNOWLEDGED	VARCHAR2(1)	Acknowledged flag
TRANSPORT	VARCHAR2(32)	Transport
DERIVED_FROM	NUMBER	Message Id which this message derived from
MESSAGE_ORIGINAL	CLOB	Original message
MESSAGE_CONVERTED	CLOB	Message after converted
ATTACHMENT_ID	NUMBER	Attachment ID refers to MHM_MSG_DM
LOADED_DATE	DATE	Loaded date
LOG	CLOB	Log information

### MHM\_MSG\_ATT

The MHM\_MSG\_ATT table is used to store message attachments.

COLUMN	TYPE	Comment
ATTACHMENT_ID	NUMBER	Not Null, unique attachment ID
ATTACHMENT_NAME	VARCHAR2(1000)	Attachment name
ATTACHMENT	BLOB	Attachment blob content

### MHM\_MSG\_DM

The MHM\_MSG\_DM table is used to store message delivery method. MESSAGE\_ID refers to message in MHM\_MSG.

COLUMN	TYPE	Comment
MESSAGE_ID	NUMBER	Not Null, unique message ID refers to MHM_MSG
PRIORITY	NUMBER	Priority
METHOD	VARCHAR2(32)	Delivery method
ADDRESS	VARCHAR2(1000)	Delivery address

### Implementation of the value map used by the message broker

A message value map is used by the message broker to translate values in the inbound and outbound messages.

COLUMN	TYPE	Comment
EC_SYSTEM_REF	VARCHAR2(32)	Not Null, EC system reference
MAPPING_CATEGORY	VARCHAR2(32)	Not Null, Mapping category
SOURCE_ID	VARCHAR2(32)	Not Null, Source ID
SOURCE_VALUE	VARCHAR2(240)	Not Null, Source value
DAYTIME	DATE	Not Null, Daytime
TARGET_VALUE	VARCHAR2(240)	Target value

### Configuration of context menu for sending messages

This section includes a more technical description of how to configure a context menu used for sending messages.

A context menu is a menu appearing on the screen when the user performs a click on the right hand mouse button. This allows the user to perform actions connected to data in a screenlet.

#### ***The context menu related EC database tables***

##### **BUSINESS\_FUNCTION**

This table should have one entry for each business function having a context menu. Table columns are:

- BF\_CODE – The code identifying the BF (MHM-0010).
- NAME – The name of the BF (this is the name attribute of the screen element in the screen XML).
- URL – The URL to the BF (this is the id attribute of the screen element in the screen XML).

##### **BF\_COMPONENT**

This table should have one entry for each component in business function having a context menu. Table columns are:

- BF\_CODE – The code identifying the BF (MHM-0010).
- COMP\_CODE – The sort name identifying a component (in the screen XMLs this is the name attribute).
- NAME – The long name describing the component (in the screen XMLs this is the fullname attribute).
- URL – The relative URL of the component (in the screen XMLs this is the id attribute).

##### **CNTX\_MENU\_ITEM**

This table should have one entry for each component in business function having a context menu. Table columns are:

- BF\_CODE – The code identifying the BF (MHM-0010).
- COMP\_CODE – The sort name identifying a component (in the screen XMLs this is the name attribute).
- ITEM\_CODE – The code identifying the menu item.
- NAME – The name of the menu item. This will be shown in the context menu.
- ACTION\_CLASS\_NAME – The full class name of the business action sending the message.
- ACTION\_ROW\_SCOPE – This is either 'all' or 'selectedRows'.
- THRESHOLD\_LEVEL – The minimum access level needed to activate this menu item.
- CONFIRM\_MESSAGE – A confirmation string. The value NULL means that the confirmation dialogue is deactivated.

- REMARK\_IND – Legal values are 'Y' and 'N'. If 'Y' then a remark can be added before executing the menu item.
- AUTO\_SAVE\_IND - Legal values are 'Y' and 'N'. If 'Y' then the changed records will be saved before executing the menu item.
- MIN\_SELECTED\_ROW – Legal values NULL or an integer.
- MAX\_SELECTED\_ROW – Legal values NULL or an integer.
- FUNC\_MESSAGE – The message attribute of the function element of the condition in the menu XML.
- FUNC\_VALIDATION - The text of the function element of the condition in the menu XML.
- DIVIDER\_IND - Legal values are 'Y' and 'N'. If 'Y' then a divider is inserted before this menu item.
- SORT\_ORDER – An integer used for sorting the context menu items.
- DESCRIPTION – The tooltip description for the menu item.

#### CNTX\_MENU\_ITEM\_PARAM

This table should have one entry for each component in business function having a context menu. Table columns are:

- BF\_CODE – The code identifying the BF (MHM-0010).
- COMP\_CODE – The sort name identifying a component (in the screen XMLs this is the name attribute).
- ITEM\_CODE – The code identifying the menu item.
- PARAMETER\_NAME – The name attribute of the action element argument.
- PARAMETER\_VALUE – The value attribute of the action element argument.
- SORT\_ORDER – An integer used for sorting the action element argument.

#### ***An example of configuring a context menu sending a message***

Here is an example of configuring a context menu sending a message.

The screen XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<screen id="/com.ec.frmw.mhm.screens/send_freetext_msg" name="Send
Freetext Message">
  :
  :
  <component relativeid="true" id="/message_template"
fullname="Message template" name="template" style="" cssclass="form">
    :
    :
  </component>
  :
  :
</screen>
```

The class implementing the send message business action is: *com.ec.frmw.mhm.screens.model.ejb.SendFreeTextAction*

We will not have any confirmation dialogue or adding remark. We will save changes before sending the message. Then the configuration will be like this:

```

insert into BUSINESS_FUNCTION (BF_CODE, NAME, URL) values ('MHM-0010',
'Send Freetext Message',
'/com.ec.frmw.mhm.screens/send_freetext_msg');

insert into BF_COMPONENT (BF_CODE, COMP_CODE, NAME, URL) values
('MHM-0010', 'template', 'Message template', '/message_template');

insert into CNTX_MENU_ITEM (BF_CODE, COMP_CODE, ITEM_CODE, NAME,
ACTION_CLASS_NAME, ACTION_ROW_SCOPE, THRESHOLD_LEVEL, CONFIRM_MESSAGE,
REMARK_IND, AUTO_SAVE_IND, MIN_SELECTED_ROW, MAX_SELECTED_ROW,
FUNC_MESSAGE, FUNC_VALIDATION, DIVIDER_IND, SORT_ORDER, DESCRIPTION)
values ('MHM-0010', 'template', 'sendmsg', 'Send Message',
'com.ec.frmw.mhm.screens.model.ejb.SendFreeTextAction',
'selectedRows', 20, null, 'N', 'Y', null, null, null, null, 'N', 10,
'The message will be sent to the reconfigured recipient');

```

## Interfaces

This section includes a more technical description of interfaces discussed in the main document.

### ***The API for sending a message***

The business actions deploying messages to the EC-MHM module will make use of the following classes:

- com.ec.frmw.mhm.model.ejb.EjbKeys
- com.ec.frmw.mhm.model.ejb.outbound.DistributionListException
- com.ec.frmw.mhm.model.ejb.outbound.MalformedMessageException
- com.ec.frmw.mhm.model.ejb.outbound.Message
- com.ec.frmw.mhm.model.ejb.outbound.MessageDistribution
- com.ec.frmw.mhm.model.ejb.outbound.MessageHeader
- com.ec.frmw.mhm.model.ejb.outbound.Recipient
- com.ec.frmw.mhm.model.ejb.outbound.RecipientList

The javadoc for these classes is found in the module com.ec.frmw.mhm released with the EC framework component.

Example code for sending a message:

```

Message msg = new Message();
msg.setTypeId(objectId);
msg.setBody(value); // Where value is a string for text
messages and the body element for XML.
msg.addAttachment(attachment); // Where attachment implements
IMessageAttachment or is a collection of these.

MessageHeader mheader = msg.getHeader();
mheader.setSubject(subject);
mheader.setPeriodFromDate(DateHelper.getNow()); //PeriodFromDate is a
string in ISO format.

InitialContext ctx = new InitialContext();
MessageDistribution msgDistr =
(MessageDistribution)ctx.lookup(MessageDistribution.JNDI_APP_NAME);

Map<String, String> params = new HashMap<>();
params.put("SUBJECT", subject);

// Use this for all messages but reports.
msgDistr.addDistributionList(msg, params, objectId);

// Use this for reports.
msgDistr.addDistributionList(msg, params);

// Optional: add remark.
if (remark != null && remark.trim().length() > 0){
    msg.setRemark(remark);
}

// Send the message.
msgDistr.sendMessage(msg);

```

### **The EC-MHM adapter**

The project specific adapter is written in Java and has to implement the interface *com.ec.frmw.mhm.model.ejb.IMessageBroker*.

The actual class implementing the EC-MHM interface is configured by setting the customise property 'Message Broker API' in the Configuration \ Preferences \ Maintain System Settings screen.

Whenever the EC-MHM adapter is implemented by using JDBC the message broker database has to be configured with the JNDI name <ECDB>\_MHM\_BROKER where <ECDB> is the name of the EC database pool.

### **The load jobs for messages**

The project specific load jobs are written in Java and have to extend the abstract class *com.ec.frmw.mhm.model.ejb.inbound.AbstractMessageLoader*. The javadoc for this class is found in the module *com.ec.frmw.mhm* released with the EC framework component.

### **The generate jobs for messages**

The project specific generate jobs are written in Java and have to extend the abstract class *com.ec.frmw.mhm.model.ejb.outbound.AbstractMessageGenerator*. The javadoc for this class is found in the module *com.ec.frmw.mhm* released with the EC framework component.

### **Security interface**

Security interfaces make the message broker able to provide support for "Secure Messaging" which is a very important

part of messaging traffic. Below is a description of how this can be implemented to make the message broker able to provide the flexibility needed when using Secure Messaging.

The following pictures give an overview of the message flow when receiving and sending messages. As described below, the security module is interacting closely with the communication process.

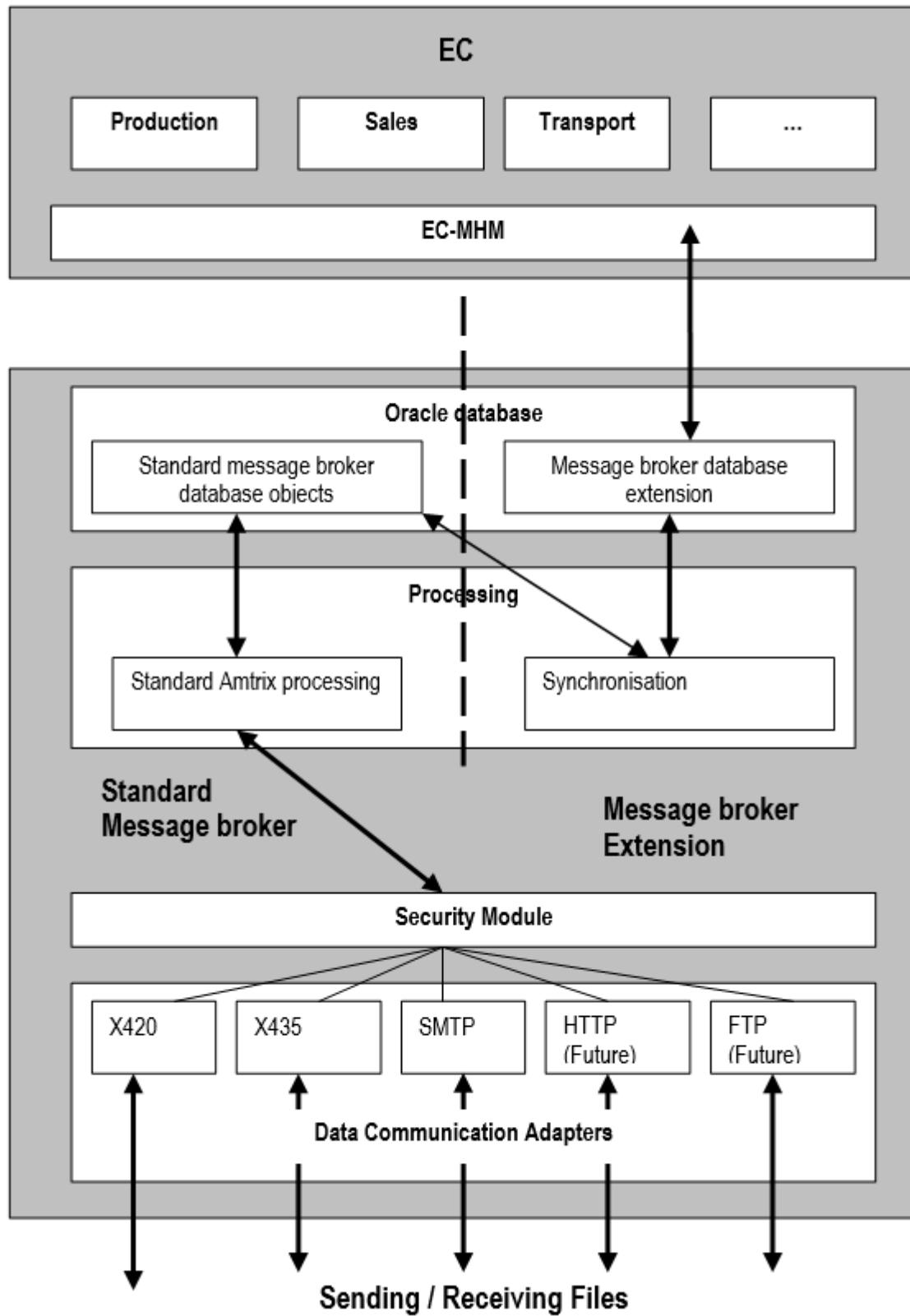


Figure 7 1

## Levels of acknowledgements

Acknowledgements are named as acknowledgements, notifications, delivery status notifications or delivery reports and serves the same function.

Acknowledgement can be handled on different levels in the interchange between parties. The levels that handle acknowledgements are from protocol level up to application level.

The usage of acknowledgements depend on the configuration, this means that in your configuration you have to ask for acknowledgements to be able to get them.

On the chosen protocols there is a difference in how they handle notification. X.400 in general has better defined acknowledgements than SMTP. SMTP supports a simpler function for acknowledgement.

### X.400 level

EDI-applications can exchange acknowledgments, "End-to-end Acknowledgments". Three types of acknowledgments can be combined. You can apply security services to each:

- Positive-responsibility Notification, X.400 has taken responsibility for the EDI transmission.
- Negative-responsibility Notification, X.400 has not taken responsibility for the EDI transmission.
- Forwarded-responsibility Notification, X.400 has passed on the responsibility for the EDI transmission.

The default behaviour of the X.420 and X.435 levels is to automatically initiate generation of notifications of received messages if the originator has requested this. You can also let an application control the sending of notifications.

Sending notifications can be controlled from the application, or be performed automatically by the communication module.

Initiation of the sending of a positive acknowledgment can be done from an application by sending a signal to the X.420 or X.435 communication module.

Sending a negative acknowledgment is done by sending a signal to the X.420 or X.435 communication module.

The specified identity must be the same as the one for the received message to acknowledge.

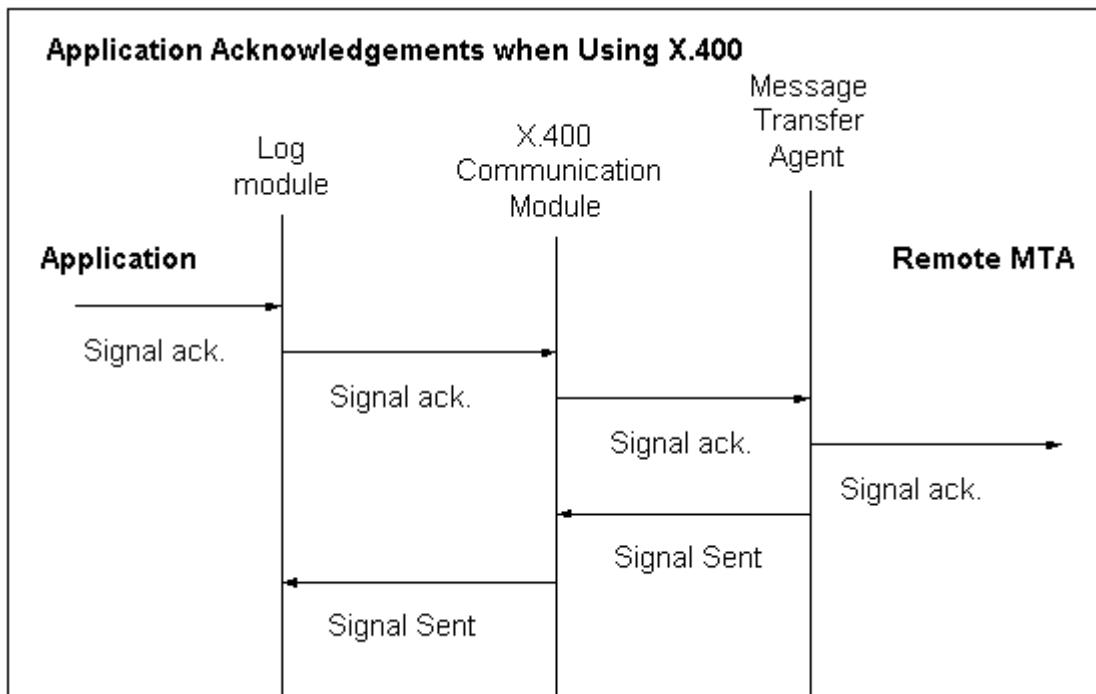


Figure 7 2

### SMTP level

The RFC 1891 specifies a protocol as an extension to SMTP that enables an SMTP client to specify that Delivery Status Notifications (DSNs) should be generated under certain conditions. The protocol is a proposed standard.

The EMAIL communication module implementation uses this protocol to enable the sender to specify whether delivery notifications should be sent or not.

The level of delivery reports can be configured using the administration graphical interface. There are three levels of delivery reports that can be requested:

- None, no delivery notification is requested.
- Failure, only report on failed message transfer
- Success and failure, reports on both of the requested statuses.

If nothing is specified, the default is to request FAILURE notifications.

The log module is used to supervise the progress of an EMAIL file transfer.

When the message has been successfully delivered to the local SMTP server, a Transfer Report of type SENT is generated. The transfer file located in the output directory is removed.

If the message cannot be successfully delivered to the local SMTP server, a Transfer Report of type FAILED is generated. This report contains information about the reason for the failure. If the configuration is done so that there is a fail directory the transfer file is moved from the output directory to the fail directory. Otherwise, the transfer file is removed.

## How to Encrypt Passwords in EC 11

### Introduction

In order to setup JBoss properly a database pool has to be defined for each database that JBoss should be able to talk with. The database pool configurations are stored in the **domain.xml** files. The database pool configuration files normally contain the database username and password as clear text. However, JBoss provides a way to configure datasource using encrypted password. This document describes how you can encrypt your clear text passwords.

### Encrypt the Password

There is a script file (**encrypt\_pw.bat**) provided with the EC distribution that encrypts passwords. This script file receives WildFly location and the password in plain text and returns an encrypted value of the password. Below is given how you encrypt the password:

```
encrypt_pw.bat -j=<WildFly_location_path> -p=<password_in_clear_Text>
```

For example:

```
encrypt_pw.bat -j=C:\wildfly-8.2.1.Final -p=energy
```

```
C:\TEMP\ec-app-install>encrypt_pw.bat -j=C:\TEMP\wildfly-8.1.0.Final -p=energy
Given parameters are :
JBoss_HOME      =C:\TEMP\wildfly-8.1.0.Final
PASSWORD        =energy

Encrypting...
Encoded password: -66d6bf0f7ee54805
C:\TEMP\ec-app-install>
```

### Encrypt Password when Installing EC the First Time

The EC installation script provides an option to encrypt your password during the installation of EC. When executing the batch, pass either **-encrypt=true** or **-e=true** to set the password encryption mode ON. Remember that you have to supply the password in an encrypted form. See the example given below:

```
configureASForEC.bat -j=C:\wildfly-8.2.1.Final
-h=WL2013720.eu.tieto.com -p=1522 -s=smrl12ee -u=ECKERNEL_USER -d=true
-pw=-66d6bf0f7ee54805 -e=true
```

This will create the data-sources all pointing to a security-domain. A sample snapshot is given below:

```

<subsystem xmlns="urn:jboss:domain:datasources:2.0">
  <datasources>
    <datasource jndi-name="java:jboss/datasources/ECDS_UNMANAGED" pool-name="ECDS_UNMANAGED">
      <connection-url>jdbc:oracle:thin:@WL2013720.eu.tieto.com:1522:smr12ee</connection-url>
      <driver>oracle-jdbc</driver>
      <security>
        <security-domain>EncryptedPassword</security-domain>
      </security>
      <validation>
        <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"/>
        <validate-on-match>false</validate-on-match>
        <background-validation>true</background-validation>
        <background-validation-millis>60000</background-validation-millis>
        <stale-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker"/>
        <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"/>
      </validation>
    </datasource>
    <xa-datasource jndi-name="java:jboss/datasources/ECDS" pool-name="ECDS">
      <xa-datasource-property name="URL">
        jdbc:oracle:thin:@WL2013720.eu.tieto.com:1522:smr12ee
      </xa-datasource-property>
      <driver>oracle-jdbc-xa</driver>
      <xa-pool>
        <is-same-rm-override>false</is-same-rm-override>
        <no-tx-separate-pools>true</no-tx-separate-pools>
      </xa-pool>
      <security>
        <security-domain>EncryptedPassword</security-domain>
      </security>
      <recovery>
        <recover-credential>
          <security-domain>EncryptedPassword</security-domain>
        </recover-credential>
      </recovery>
      <validation>
        <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"/>
        <validate-on-match>false</validate-on-match>
        <background-validation>true</background-validation>
      </validation>
    </xa-datasource>
  </datasources>
</subsystem>

```

The security domain, named EncryptedPassword is defined in the domain.xml as shown below:

```

<security-domain name="EncryptedPassword" cache-type="default">
  <authentication>
    <login-module name="SecureIdentityLoginModule" code="org.picketbox.datasource.security.SecureIdentityLoginModule" flag="required">
      <module-option name="username" value="ECKERNEL_USER"/>
      <module-option name="password" value="-66d6bf0f7ee54805"/>
    </login-module>
  </authentication>
</security-domain>

```

The security domain is defined with a username and a password. The password is encrypted. The code, "org.picketbox.datasource.security.SecureIdentityLoginModule" is used as code. This code is a JBoss' built in tool and is used to decrypt the password when connection is created with the data source.

Then deploy EC, afterwards you should be able to login and access EC.

## Encrypting Password of Already Running EC

You have also an option to encrypt at later stages, if, for some reason, you didn't do so when installing EC. Follow these steps to then apply encryption.

1. Undeploy EC application. Run the jboss-cli.bat and trigger the undeploy command, as follows:

```

[domain@localhost:9990 /] connect
[domain@localhost:9990 /] undeploy ec-app.ear
--all-relevant-server-groups

```

2. Run the script, **configureECPasswordEncryption.bat** to remove the old data-source, and add the new one. This script has almost the same parameters as the EC install script (configureASForEC.bat), but only deals with the data-source configuration.

```
configureECPasswordEncryption.bat -j=C:\TEMP\wildfly-8.2.1.Final
-h=WL2013720.eu.tieto.com -p=1522 -s=smr12ee -u=ECKERNEL_USER
-pw=-66d6bf0f7ee54805
```

You should have an output that looks like:

```
C:\TEMP\ec-app-install>configureECPasswordEncryption.bat -j=C:\TEMP\wildfly-8.2.1.Final
-h=WL2013720.eu.tieto.com -p=1522 -s=smr12ee -u=ECKERNEL_USER
-pw=-66d6bf0f7ee54805
Configure with Parameters :
DB_SERVER_HOST      =WL2013720.eu.tieto.com
DB_SERVER_PORT       =1522
DB_SERVICE_NAME     =smr12ee
DB_SERVER_USERNAME   =ECKERNEL_SEMERE
DB_SERVER_PASSWORD   =-66d6bf0f7ee54805
JBOSSE_DIR          =C:\TEMP\wildfly-8.2.1.Final
BIND_ADDRESS         =localhost

xxxxxxxxxxxxxx
** Configuring EC **
xxxxxxxxxxxxxx
Jboss bin directory   : C:\TEMP\wildfly-8.2.1.Final\bin
The batch executed successfully
The batch executed successfully
The batch executed successfully
The batch executed successfully
(
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => undefined
)
(
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => undefined
)
C:\TEMP\ec-app-install>
```

3. Deploy EC again.

```
[domain@localhost:9990 /] connect
[domain@localhost:9990 /] deploy
C:\TEMP\ec-app-install\bin\ec-app\ec-app.ear
--server-groups=ec-server-group-a
```

EC should successfully deploy.

4. Access EC through browser. You should be able to login.

## Troubleshooting

- If Oracle is configured incorrectly, you will experience the following error in your log files:

```
WARN [com.arjuna.ats.logging.loggerI18N]
[com.arjuna.ats.internal.jta.recovery.xarecovery1] Local
XARecoveryModule.xaRecovery got XA exception
javax.transaction.xa.XAException, XAException.XAER_RMERR
```

To resolve this error, be sure that the Oracle user has access to the appropriate tables to accomplish the recovery:

```
GRANT SELECT ON sys.dba_pending_transactions TO user;
GRANT SELECT ON sys.pending_trans$ TO user;
GRANT SELECT ON sys.dba_2pc_pending TO user;
GRANT EXECUTE ON sys.dbms_xa TO user;
```

The above assumes that *user* is the user defined to connect from JBoss to Oracle.



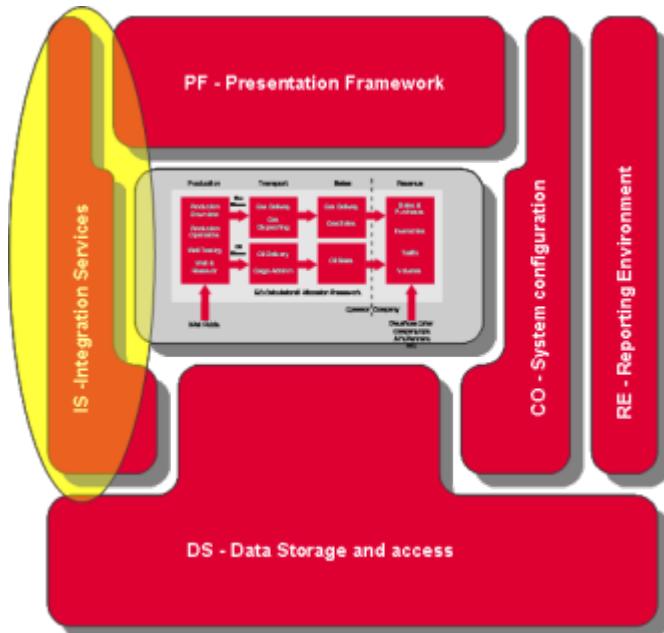
# Integration Services Technical Documentation

## Introduction

This section is the technical documentation of the EC Integration Services (EC IS). EC IS is the module handling interfacing of 3<sup>rd</sup> party systems to EC. This includes metering data, file based data, etc.

The section has a top-down structure. First an overview of the EC Integration Services is given. This is followed by one main section for each of the main services types, e.g. Tag Services, Web Services. For each service type, an overview and required configuration and operational details are provided. Supplementary information is included in appendixes.

## Overview



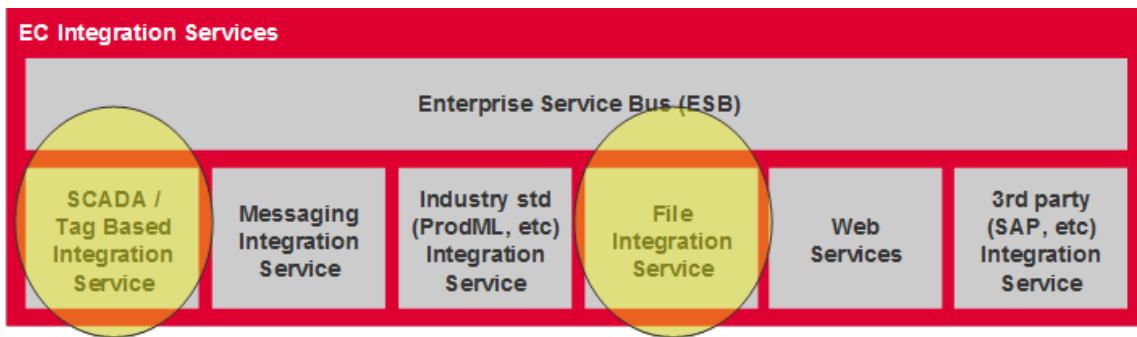
EC Integration Services (EC IS) is the EC Framework module that over time will handle all integration of EC with external parties. The first building blocks of EC IS were introduced in EC Release 9.1. Subsequent blocks were introduced in later releases.

The business scope for EC IS is to cover all integration methods relevant for EC. As shown in the below diagram, the current roadmap is to support the following methods:

- SCADA / Tag Based Integration Service: Tailored for integration with metering / tag based solutions
- Messaging Integration Service: Handling message oriented data flow
- Industry Standard Integration Service: Exchanging data based on the industry standards like ProdML, EDIGAS, etc.
- File Integration Service: Exchanging row based files.
- Web Services: Providing web service based solution interface
- 3rd Party Integration Service: Exchanging data with 3rd party software like SAP, etc.

This document covers the parts of EC IS available in EC Release 11.1:

- SCADA / Tag Based Integration Service
- File Integration Service



From an IT architecture point of view, EC IS is based on Service Oriented Architecture (SOA) principles, and is fully integrated with the EC technical platform (EC Framework).

The individual integration services are designed as modularised, configurable units. As an example, the SCADA / Tag Based Integration Service is segmented into a set of services typically dealing with connection to the specific source system (adapter), aggregating / transforming data, mapping data and inserting data in the database. Important reasons for doing this are to break down the complexity handled at each step, and to make it easier to plug in / out services as needed. This architecture also provides better control with the execution, resulting in a more robust behaviour.

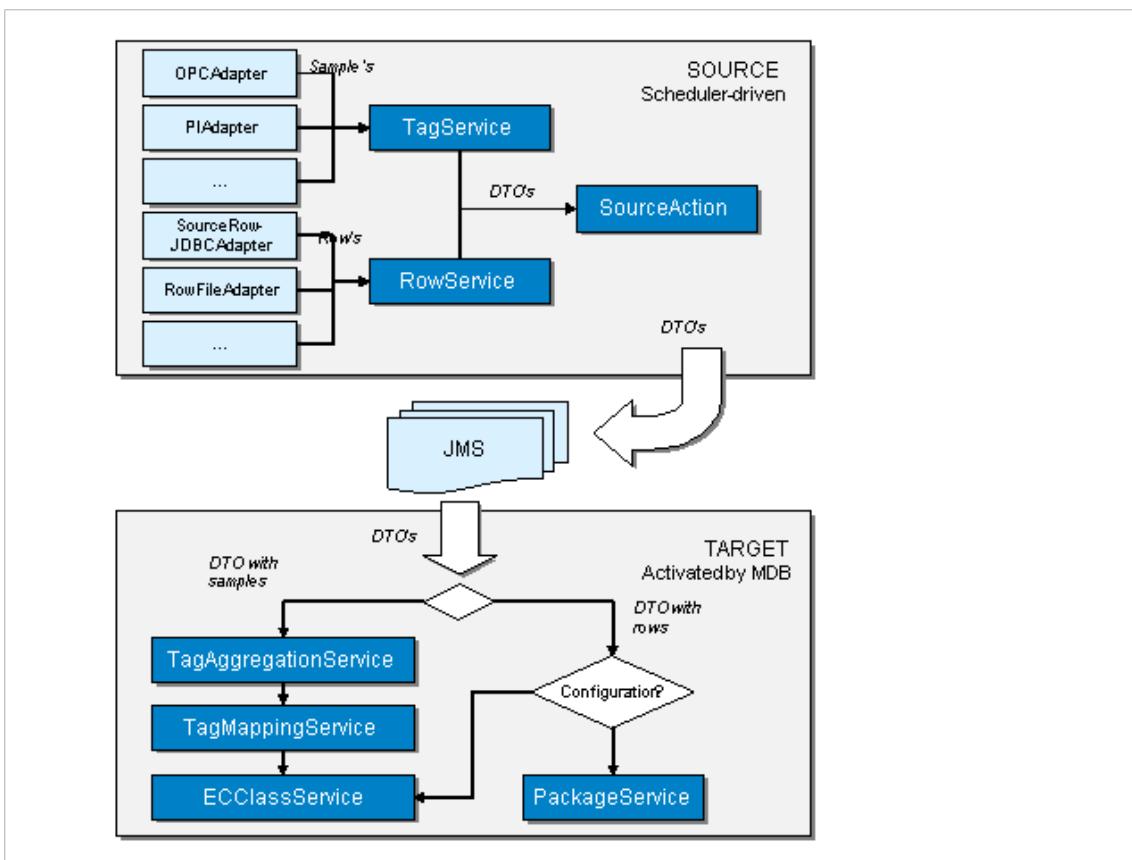
## Tag / Row Based Integration Service

### Overview

The SCADA / Tag Based Integration Service handle tag based data, either from SCADA systems or from files. Tag based data means data records consisting of a tag (typically identifying a specific metering equipment), a date/time value and a data value.

The File Integration Service handle row based data from files. Row based data means data records where each value is separated by a defined delimiter (typically ';'), and where the configuration holds information of what data element resides in each position.

The Tag and Row Based Integration Service of EC IS are built on the same concepts. From a technical point of view, the code-base is therefore common. The module handling these services is based on a traditional ETL ETL: Extract, Transform, Load principle.



The architecture solving this is split into two main areas, the source area for extraction, and the target area for transformation and load. The source components are separated from the target. It will always have an adapter plugged into it. This adapter provides access to a specific source system. Currently adapters for PI, OPC, File, IP21 and JDBC are included. Future releases will contain new adapters. Further, it is possible to implement customer specific adapters (Customized Adapter).

The source-side is triggered from the EC Scheduler, and runs once or by given frequencies. The target side will always be running and listening for data sent from the source side into the message queue (JMS). The target side will handle the data according to configured rules, and load them into the EC database. The engine controlling the target area is a Message Driven Bean (MDB).

Later sections in the document will describe in detail how the internal services work.

## Processing Steps

### **Source Data Extraction Using Adapters**

Generally, data extraction can be implemented in many different ways, often introducing unnecessary complexity to the end users. Two source systems may be conceptually equal to each other, but actual data extraction introduces different configuration concepts and call mechanisms. Hence, to build a generic extraction component supporting several source systems is a challenging task with a high risk of creating a "too big too black box".

EC IS has solved this by introducing system specific adapters which delivers data into a common data format for further processing. The configuration of adapters is also system specific. The setup process differs for each system. This strategy means that the configuration of an adapter deals directly with the source system terms instead of abstract and generic expressions.

The interface between the adapters and the common TagService component has been opened Open interface: Programmable components are available for Java and C++ to make new pluggable adapters to any source system. A "Programmers Guide" will be provided for external programmers, for external programmers, so that new adapters can be implemented and plugged directly into a running EC IS.

A common set of functions will always be available thru the open interface:

- Retrieval of tag list
- Retrieval of tag events (SAMPLE) for specific time intervals
- Retrieval of aggregated tag data, based on time weighted average, mean, min, max or sum
- Retrieval of events at times based on intervals (VALUE\_AT\_END, VALUE\_AT\_START)

All data will be time stamped using UTC at adapter level to ensure a common event data interpretation. Aggregated data will deliver data time stamped at the beginning of period. Given a data quality indicator is available from the source system, the adapter will only extract samples marked with good quality (100%).

For tag data each adapter will convert it's extractions to the columns:

- Tag name
- Time
- Value

### **Tag Aggregation service**

The Tag Aggregation Service transforms tag data into values prepared for the EC class model. It takes tag data, containing tag name, timestamp and value and checks it against the configured mapping rules. The main task is to handle data according to the fixed EC time resolutions. This assumes the Tag Aggregation Service to be aware of changes in daylight saving time, time shifts and to handle time weighting between periods.

The aggregation configuration calculates the expected number of samples for each period. To complete a specific value, it has to be built up by that number of samples. If a value cannot be released due to the completion rules, it will wait for a period timeout to occur. It will be further checked if it satisfies the minimum rules. The timeout can span over several runs, which also means that data will be persisted in the system for a specific time.

The output from the Tag Aggregation Service is:

- Mapping identifier
- Timestamp aligned to EC class specification
- Aggregated value

### **Tag Mapping Service**

The Tag Mapping Service is responsible for mapping aggregated values into EC class rows. Each value from the Tag Aggregation Service will be mapped to an attribute in a particular EC class. The basis for this service is a row buffer which groups the incoming values into its right place. When a specific number of values have been processed, the row buffer will be flushed. A row does not need to be 100% completed when it is flushed. The remaining attributes may come later, either in the next flush or in a later run. The only difference between a full and uncompleted row is that the resulting number of database updates will differ.

The output from the Tag Mapping Service is:

- Mapping identifier
- Data Class
- <Class attribute 1 value>
- <Class attribute 2 value>
- ...
- <Class attribute n value>

The attribute values will also be provided with UOM: Unit of measure. Timestamps will be converted from the internal UTC time zone to the time zone given by the EC system.

### **EC Class Service**

The EC Class Service is responsible for inserting or updating rows towards EC specific classes. It receives data from the tag mapping service about which data class and attributes to update. It has to be aware of the UOM, so that data is converted to the right class attribute unit.

This service will always look for existing data in the database. Hence, it tries to update data before it is inserted. Group of rows will be processed together to obtain optimized performance.

It is also possible to configure ECIS to send row-based data directly to the classservice. ECIS will then based on the configuration, write the data directly to EC classes.

### **Package Service**

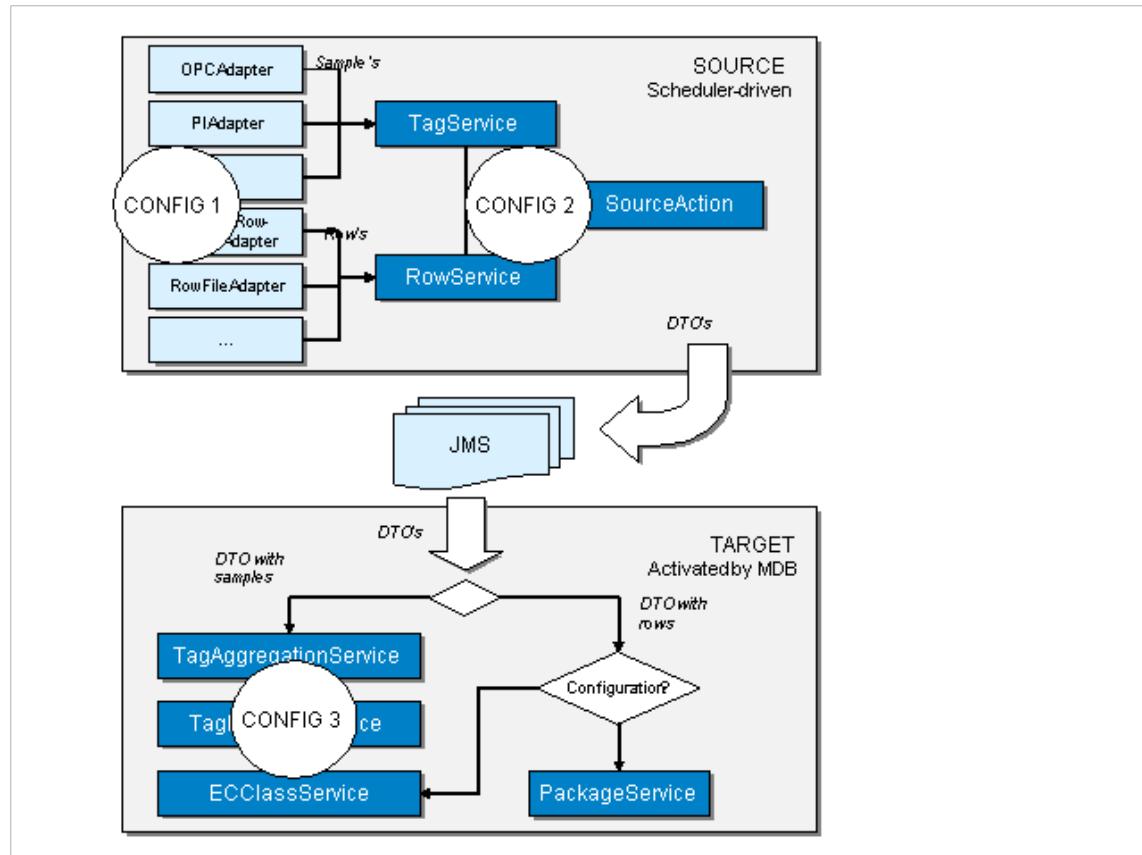
The Package Service is used to write row-based data. If you configure ECIS to use this service, you have to specify

which package ECIS should send the rows to. It will then send the rows to this package, and the person making the package decides what to do with the rows.

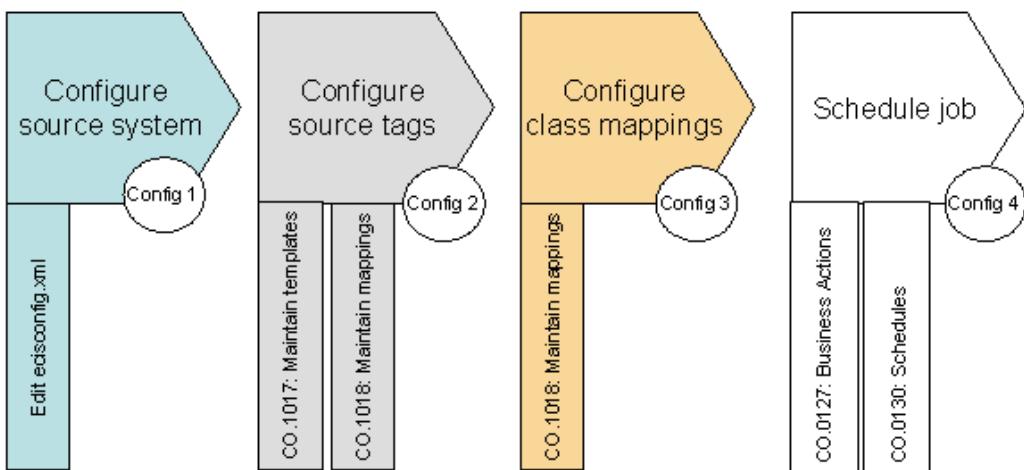
## Configuration

The configuration of EC IS tag data capture consists of four different contexts as shown in the drawing below:

1. Source System Configuration
2. Source Tag Configuration
3. Class Mapping Configuration
4. Schedule Job

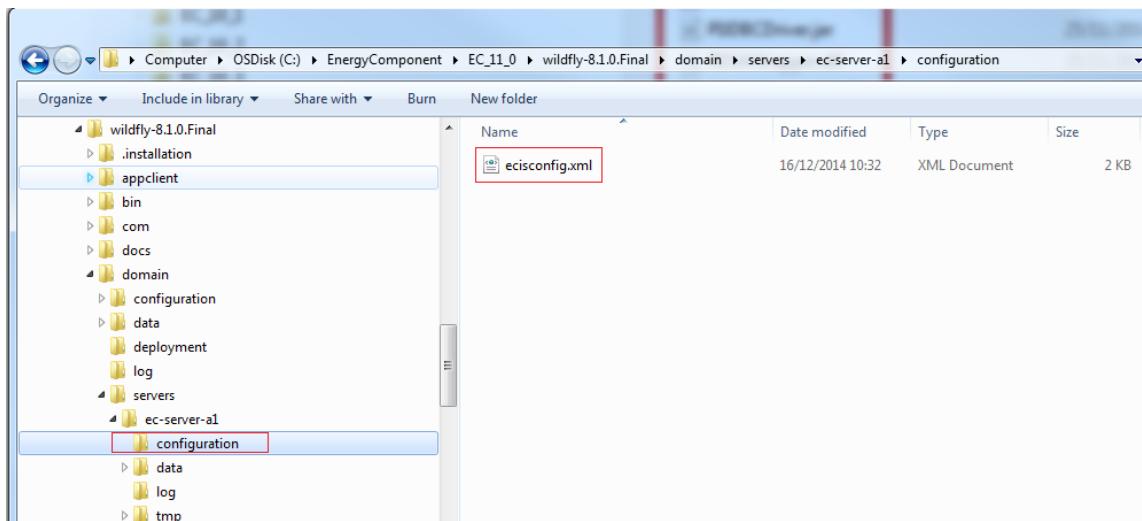


To complete a configuration and start executing in EC IS tag based service, the following steps can be used.



### **Source System configuration**

A source system is actually a configured source adapter and represents config step 1 according to the drawing above. To setup an adapter for data retrieval, the /<wildfly>/domain/servers/ec-server-a1/configuration/ecisconfig.xml has to be edited. Manually create the configuration folder and ecisconfig.xml file if configuration folder and ecisconfig.xml file are not exist/found inside the /<wildfly>/domain/servers/ec-server-a1 folder as below screenshot.



Snapshot of the ecisconfig.xml file content.

```

<configurations>
    <config name="PI_SOURCE_1">
        <sourceadapter>

            <class>com.ec.frmw.is.engine.adapter.pi.jinterop.PiAdapter</class>
                <parameters>
                    <parameter name="dsn">127.0.0.1</parameter>
                    <parameter name="username">piadmin</parameter>
                    <parameter name="pwd"></parameter>
                    <parameter name="timeout">180</parameter>
                    <parameter
name="logfile">c:\temp\logfile2.log</parameter>
                </parameters>
            </sourceadapter>
        </config>
        <config name="PI_SOURCE_2">
            <sourceadapter>

                <class>com.ec.frmw.is.engine.adapter.pi.jinterop.PiAdapter</class>
                    <sequential>Yes</sequential>
                    <parameters>
                        <parameter name="dsn">192.10.10.1</parameter>
                        <parameter name="username">piadmin</parameter>
                        <parameter name="pwd"></parameter>
                        <parameter name="timeout">180</parameter>
                        <parameter
name="logfile">c:\temp\logfile3.log</parameter>
                    </parameters>
            </sourceadapter>
        </config>
        <config name="FILEADAPTER">
            <sourceadapter>

                <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
                    <parameters>
                        <parameter name="DropFolder">c:\\temp</parameter>
                        <parameter
name="CompletedFolder">c:\\\\temp\\\\completed</parameter>
                        <parameter
name="ErrorFolder">c:\\\\temp\\\\error</parameter>
                        <parameter name="BadFolder">c:\\\\temp\\\\bad</parameter>
                        <parameter name="DateFormat">dd.MM.YYYY
HH:mm:ss</parameter>
                        <parameter name="FileFilter"></parameter>
                        <parameter name="QualityColumnIndex">0</parameter>
                        <parameter name="TagColumnIndex">1</parameter>
                        <parameter name="TimeColumnIndex">2</parameter>
                        <parameter name="ValueColumnIndex">3</parameter>
                    </parameters>
            </sourceadapter>
        </config>
    </configurations>

```

If encryption is wanted, please refer to the section How to Encrypt password in EC 11 in the technical documentation. Use this method to encrypt the password, and put the encrypted password as the parameter. In addition one needs to set encrypted="true" on the same parameter. An example of this would look like this:

```
<parameter name="pwd" encrypted="true">-66d6bf0f7ee54805</parameter>
```

Each config instance has a source adapter related to it. Note that all the configuration instances (*config name*) must be added as EC codes with the type DT\_SOURCE\_ID. The mandatory attributes for a source adapter are *class* and *library*. The *class* is a component specifying the specific adapter into EC IS. For specific configuration of EC provided adapters, please refer to Appendix A in this document. For more information about how to implement customer specific adapters, contact the EC product development team.

The screenshot shows a software interface for managing EC codes. On the left is a navigation menu with sections like Configuration, Access, Preferences, Codes (with sub-options for All, Non-System, and System Codes), System, Assets, Integration Services, Scheduler, and Messaging. The main area is titled "EC Codes - All" and contains a table with columns: Code Type, Code, Text, Description, Alt. Code, Sort Order, Default, Active, and System Code. The table data is as follows:

Code Type	Code	Text	Description	Alt. Code	Sort Order	Default	Active	System Code
DT_SOURCE_ID	PI	PI			1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DT_SOURCE_ID	NORWAY	NORWAY			2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DT_SOURCE_ID	BRAZIL	BRAZIL			3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DT_SOURCE_ID	NF	NF			4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Below the table, there are buttons for Code Type 1, Code 1, Code Type 2, Code 2, Code Type 3, and Code 3. A message "No records found." is displayed at the bottom of the table area.

The specification for the additional parameters can vary, dependent on the adapter type.

As you see in the example above, you can have multiple source systems in a configuration. You can even apply the same source adapter to different configurations but with different settings.

The next section shows how to map the configurations to specific data retrieval.

For verification of the configured adapter, TE has made a company internal tool for browsing tag data using the adapter. For more information please contact the EC product development team.

### Source System Failover Configuration

In order to obtain high availability of adapters it's possible to configure several adapters using the same configuration name. The first adapter in the file will always have highest priority. If it's not available, e.g. can't connect or fails during tag read, it will be degraded and the next schedule will use the next adapter in the sequence.

### Maintain Template (Trans\_template) usage

The CO.1017: *Maintain Templates* configuration screen is used as a primary description of how data will be captured. As an example, the source interval is used to calculate how many samples to expect for a given period in EC, and the target interval is used to specify the length of a period resolution represented by a resulting class. Please note the EC IS specific rules for templates:

1. Source function is mandatory
2. Source interval is mandatory
3. Source datatype is mandatory
4. Minimum samples is mandatory
5. Target function is mandatory
6. Target datetime is mandatory and upper case

Template Code	Description	Source Function	Source Datatype	Source Interval	Source Delay	Minimum Samples	Target Interval	Target Function	Target Datatype	Target Datetime	Use Summertime	Overwrite User	Overwrite Status	Prod Day Start	Shift Time To Period Start
AVG_LATEST	One Value P	Avg	DECIMAL	86400	600	1	86400	LATEST	DECIMAL	DAYTIME	<input checked="" type="checkbox"/>	P	P	0	
SAMPLE_LA	One Value P	Avg	DECIMAL	86400	600	1	86400	LATEST	DECIMAL	DAYTIME	<input checked="" type="checkbox"/>	P	P	0	
SAMPLE_LA	One Value P	Value_At_	DECIMAL	86400	600	1	86400	LATEST	DECIMAL	DAYTIME	<input checked="" type="checkbox"/>	P	P	0	

Below is a table of the configuration parameters available in the Maintain template screen, and a short explanation of the use in ECIS.

Source interval will in most cases be set equal to TARGET\_INTERVAL. Note that the SOURCE\_INTERVAL will be used together with SOURCE\_DELAY to find the end of a request, in a request to the source system (see next section for more details).

For Tags with source function=SAMPLE, the current implementation still requires a SOURCE\_INTERVAL <= TARGET\_INTERVAL.

Configuration parameter	Comments
TEMPLATE_CODE	Logical name for template
SOURCE_DATATYPE	Defines expected data type in the source system
SOURCE_FUNCTION	Function used when fetching data from source system
SOURCE_INTERVAL	Interval used in request from source system.
SOURCE_DELAY	Time to wait before requesting a given time from the source system
MINIMUM_SAMPLES	Minimum samples needed in target period before completing the period.
TARGET_INTERVAL	Defines resolution (Frequency) for values written to the EC database
TARGET_FUNCTION	Function used to aggregate values within ECIS.
TARGET_DATETIME	Name of Date column in EC where the timestamp of Target row will be written. Uppercase.
USE_SUMMERTIME	Indicates if the EC data class needs a Daylight saving flag as part of PK
TARGET_TRUNCATE	This should only be used if you are reading daily data in a timezone which change DST at 00.00 local time, like Brazil. (If so, set to DD)
OVERWRITE_USER	Flag indicating if Rows changed by non-system users can be updated by ECIS
OVERWRITE_STATUS	Highest Record status that can be changed by ECIS
PROD_DAY_START	Production day offset in hours.

### Maintain mappings (Trans\_mapping) Usage

To configure the actual mappings against the classes, the CO.1018: *Maintain Mappings* configuration screen is used. Each mapping is based on a data class attribute and related to a template. Please note that the source id for a mapping must be a registered EC code, configured by the DT\_SOURCE\_ID type (This code is also the same as the one applied to

config instances in the `ecisconfig.xml`).

Attribute	Source Id	Tag Id	Template Code	From Unit	To Unit	Last Transfer	Active
AVO_BR_PRESS	NORWAY	DEC_DAILY_NORWAY_TAG_2	SAMPLE_LATEST_DAY			2014-11-30 00:00	<input checked="" type="checkbox"/>
AVO_BR_EMP	NORWAY	DEC_DAILY_NORWAY_TAG_1	SAMPLE_LATEST_DAY			2014-11-30 00:00	<input checked="" type="checkbox"/>
AVO_IW_PRESS	NORWAY	DEC_DAILY_NORWAY_TAG_3	SAMPLE_LATEST_DAY			2014-11-30 00:00	<input checked="" type="checkbox"/>

Under is a table of the configuration parameters available in the Maintain mapping screen, and a short explanation of the use in ECIS. The screens here are based on the established "DT-classes". Note even if ECIS are not using the DT classes directly, they are still needed if the mappings are maintained with the Maintain Mapping screen. The Navigator here will select the target EC Class and the object the rest of the configuration parameters applies for.

Configuration parameter	Comments
ATTRIBUTE	Name of the EC Class attribute the value will be mapped to.
SOURCE_ID	Logical name of the source system. Must correspond with SOURCE_ID in Config 1.
TAG_ID	Name of the tag from the source system. Note the TC "-COPY-" postfix convention for mapping the same source tag to several EC Class attributes.
TEMPLATE_CODE	Transaction template that will be used for this tag in the transfer.
FROM_UNIT	For unit conversion (see under). Defines the UOM of the incoming value.
TO_UNIT	For unit conversion. Defines the UOM to convert to for the incoming value. Not needed if the Class attribute has a defined UOM.
LAST_TRANSFER	Shows the timestamp of the latest value written to the EC database. Note this can be moved back or forward to control the next period to read. Moving the timestamp forward will only have effect if it is done by a user that is not defined as a system user in CTRL_SYSTEM_ATTRIBUTE.ATTRIBUTE_TYPE=DT_OVERWRITE
LAST_TRANSFER_WRITE	Not used by ECIS
ACTIVE	Turn Mapping on/off
ATTRIBUTE PRIMARY KEY N	Class dependent , set additional PK columns
VALUE OF PRIMARY KEY N	Class dependent , set additional PK columns

If the PK\_ATTR\_2 attribute is set to "DATA\_CLASS\_NAME", the attribute and value should be removed (it is implicitly retrieved from the DATA\_CLASS attribute of the mapping table in EC IS). This attribute is for some classes mandatory. To ensure expected behavior, the mandatory flag should be turned off on class level.

### Unit Conversion under ECIS

The following rules applies for unit conversion under ECIS:

1. If "From Unit" and "To Unit" is missing (NULL) then there are no conversion done, and no warnings or errors loged.
2. If "From Unit" and "To Unit" are defined and the UOM for the class attribute is also defined then the value will be converted "From Unit-> To Unit" and also "To Unit -> Class Attribute UOM"
3. If the "From Unit" and the "Class Attribute UOM" are defined but not the "To Unit", then the value will be converted "From Unit-> Class Attribute UOM"

For 2 and 3 any mismatches or missing conversion factors will result in warnings or errors logged.

#### **Algorithm for when and what to request from the source system**

A typical request to the source-system in ECIS will be like this:

```
source_function (TAG_ID, FROM_TIME, TO_TIME, SOURCE_INTERVAL)
```

The function will return a list of samples.

The parameters for the source function are calculated in the following way:

- FROM\_TIME = TRANS\_SOURCE\_TIME.NEXT\_READ (this is again calculated from the previous read for the tag)
- TO\_TIME = <System time> - SOURCE\_DELAY, shifted back to end of completed intervals based on SOURCE\_INTERVAL
- INTERVAL\_RESOLUTION = SOURCE\_INTERVAL

After the result set has been received and processed, NEXT\_READ will be updated. This can happen in two ways:

1. If SOURCE\_FUNCTION is SAMPLE: NEXT\_READ will be set to the timestamp of the sample with the latest timestamp + 1 second.
2. If SOURCE\_FUNCTION is an aggregated function, NEXT\_READ will be set to the timestamp of the sample with the latest timestamp + 1 source\_interval (Thus the beginning of the next source\_interval).

For source function SAMPLE, the SOURCE\_INTERVAL will not be passed to the source system. But it will still be used in the calculation of the TO\_TIME. NEXT\_READ will be still be set to TO\_TIME.

As a more technical detail, ECIS will always divide a request into RequestIntervals. This is to avoid problems with daylight savings time. For SOURCE\_FUNCTION SAMPLE, ECIS will create one long requestinterval from FROM\_TIME to TO\_TIME. If you have aggregated functions, and your request passes a DST change, ECIS will divide your request into 3. One request before the DST-change, one request of 1 source\_interval over the DST change, and 1 request after and until TO\_TIME. It is also possible to specify that a request against the source\_system can only contain a given number of source\_intervals. This is typically used if you have a source\_system which does not handle too large requests. For more information about this, see section Max number of sourceintervals in one request. **Example:**

```
Input:
TAG_ID = 14-PT-0345
SOURCE_INTERVAL = 1800 sec
SOURCE_DELAY= 900 sec
NEXT_READ = 2008-09-01 06:00
System clock = 2008-09-01 09:05
Calculate to_time:
<System clock> - SOURCE_DELAY = 2008-09-01 08:50
Shifted back to complete period: To_time == 2008-09-01 08:30
Call: AVG('14-PT-0345', 2008-09-01 00 06:00, 2008-09-01 00 08:30, 1800)
Result:
Tag Timestamp Value
-----
'14-PT-0345' , 2008-09-01 00 06:00, 52,3
'14-PT-0345' , 2008-09-01 00 06:30, 50,5
'14-PT-0345' , 2008-09-01 00 07:00, 51,3
'14-PT-0345' , 2008-09-01 00 07:30, 51,8
'14-PT-0345' , 2008-09-01 00 08:00, 52,1
'14-PT-0345' , 2008-09-01 00 08:30, 52,4
```

NEXT\_READ will then be set to 2008-09-01 09:00 (last retrieved timestamp + 1 source\_interval)

#### **ECIS Source functions**



ECIS supports a number of source functions listed in the table below. Be aware of the following when setting up source functions:

- For a period defined by the from and to timestamps the aggregation function will return one value for all the complete intervals within the period.
- All aggregated values delivered from the source adapter should be timestamped at the beginning of the period.
- If a source system delivers aggregated values timestamped at end of intervals the source adapter should be configured to convert the timestamps to the beginning of the intervals. To configure this see 3.3.17.

SUM	(from,to, source_interval )	Dependent on source system implementation
MIN	(from,to, source _interval )	
MAX	(from,to, source_interval )	
MEAN	(from,to, source_interval )	Dependent on source system implementation.
AVG	(from,to, source_interval )	Time weighted, dependent on source system implementation
SAMPLE	(from,to )	Returns all samples with given period.
VALUE_AT_START	(from,to, source_interval )	For each interval, take the latest sample at or before the interval start time. Timestamp to start of interval. To be implemented
VALUE_AT_END	(from,to, source_interval )	For each interval, take the latest sample within or before the interval. Timestamp to start of interval. To be implemented
COMPRESSED	(from, to, source_interval)	Will give interpolated values. Only available for PI-adapter.
AVG_AT_END	(from, to, source_interval)	For each interval, if there is more than one value, take the average sample within the interval; else take the value before the interval. Timestamp to start of interval.

#### Sourcefunctions supported by adapter

	JDBC-adapter	PI-adapter	OPC-adapter	CSV-adapter	IP21-Adapter
SAMPLE	X	X	X	X	X
AVG	X*	X	X	X**	X
SUM	X*	X	X	X**	X
MAX	X*	X	X	X**	X
MIN	X*	X	X	X**	X
MEAN	X*	X	x	X**	
VALUE_AT_START	X***	x***	X***	X***	X
VALUE_AT_END	X***	X***	X***	X***	X
COMPRESSED		X			
AVG_AT_END	X***	x***	X***	X***	X***

- User need to specify a sql query against the database which gives the aggregated values.
  - Aggregation done in adapter
  - Aggregation done in ECIS except AVG which is done in SCADA system

## ECIS Target functions

If aggregation is not possible on the source-side, it may be necessary to aggregate in ECIS. To do this, ECIS supports a list of target functions. All target function operates on samples within a given target\_interval. For function SAMPLE the target\_interval is not applicable.

SUM	Sum of values within target_interval
MIN	Minimum value within target_interval
MAX	Maximum value within target_interval
MEAN	Mean value within target_interval
AVG	Time weighted average values within target_interval
SAMPLE	Sent through as is
LATEST	Returns the latest values within target_interval

### Overwrite Rules

There exists a list of system users which are allowed to overwrite data. This requires that the overwrite user flag is set to Y in the CO.1017: *Maintain Templates*. These users can be configured as a comma separated list in the system attributes. The attribute is called DT\_OVERWRITE.

### Scheduling data capture

The execution schedule for the tag Extraction step has to be defined in the EC Scheduler. It is recommended to align the execution schedule with the target interval defined for the Transform and Load step. The configuration of the execution schedule is described below. For more information about the scheduler and scheduler terms, please refer to *EC Framework – Configuration Manual*.

1. Make sure that a business action is available (ref. CO.0127: *Business Actions*).

The business action has to be linked to the following class: `com.ec.frmw.is.engine.source.action.SourceAction`. It must have two mandatory parameters with names:

- configurationid (EC Code Type DT\_SOURCE\_ID)
- jobid (String type)

2. Make a schedule of the business action (CO.0130: *Schedules*)

To schedule the business action, insert a new schedule and apply the business action. The two parameters have to be specified. The *configurationid* must match one of the config instances in the `ecisconfig.xml` file. The *jobid* is only an arbitrary name which identifies the job when it runs.

### Merging DTO's BEFORE putting on queue

Per default ECIS will merge DTO's before putting them on the JMS-queue. If you require ECIS to handle one tag at a time, you can use `dtomergeenabled`. The `dtomergeenabled` value is set to true by default. Set it to false and ECIS will send one DTO per tag. When set to true, the buisnessfunction will fill all DTO's put on the jms queue with 5000 samples (if there are enough samples in schedule) instead of sending one sample for each 1000 samples or each tag. If you do not have any good reason for setting this to false, leave it enabled. An example config using this parameter will look like this (notice that the xml tag is put under the config tag, not as an adapterparameter):

```

<config name="TAGADAPTER">
    <dtomergeenabled>false</dtomergeenabled>
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
        <parameters>
            <parameter
                name="BadFolder">C:\\temp\\jboss\\fileadapter\\bad</parameter>
            <parameter
                name="CompletedFolder">C:\\temp\\jboss\\fileadapter\\completed</parameter>
            <parameter
                name="DateFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
            <parameter
                name="DateValueFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
            <parameter
                name="DropFolder">C:\\temp\\jboss\\fileadapter</parameter>
            <parameter
                name="ErrorFolder">C:\\temp\\jboss\\fileadapter\\error</parameter>
                <parameter name="FileFilter">merge</parameter>
                <parameter name="QualityColumnIndex">1</parameter>
                <parameter name="TagColumnIndex">4</parameter>
                <parameter name="TimeColumnIndex">0</parameter>
                <parameter name="ValueColumnIndex">2</parameter>
            </parameters>
        </sourceadapter>
    </config>

```

### ***Rounding Values from source system***

It is possible to round the values from the source system into a specific number of decimals. This is done by setting a parameter in the configuration called *maxnumberofdecimals*. This specifies the maximum number of decimals the decimal-values retrieved using that configuration should have. It is using the roundingmode `BigDecimal.ROUND_HALF_UP`, described in the javadoc for `BigDecimal`.

```

<config name="TAGADAPTER"> maxnumberofdecimals

    <dtomergeenabled>true</dtomergeenabled>
    <maxnumberofdecimals>4</maxnumberofdecimals>
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
            <parameters>
                <parameter
                    name="BadFolder">C:\\temp\\jboss\\fileadapter\\bad</parameter>
                <parameter
                    name="CompletedFolder">C:\\temp\\jboss\\fileadapter\\completed</parameter>
                <parameter
                    name="DateFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
                <parameter
                    name="DateValueFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
                <parameter
                    name="DropFolder">C:\\temp\\jboss\\fileadapter</parameter>
                <parameter
                    name="ErrorFolder">C:\\temp\\jboss\\fileadapter\\error</parameter>
                <parameter name="FileFilter">merge</parameter>
                <parameter name="HaltOnMissingData">SAMPLE</parameter>
                <parameter name="QualityColumnIndex">1</parameter>
                <parameter name="TagColumnIndex">4</parameter>
                <parameter name="TimeColumnIndex">0</parameter>
                <parameter name="ValueColumnIndex">2</parameter>
            </parameters>
        </sourceadapter>
    </config>

```

#### **Max number of sourceintervals in one request**

Sometimes it might be desirable to reduce the size of the interval requested from the SCADA-system. This can be because the system cannot deliver more than a certain amount, or if it handles small amount of data better. To use this functionality, set the parameter *maxsourceintervalinrequest* to the number of source intervals you want to request in one go. Be aware that this functionality can lead to performance degradation if the intervals become small and many.

```

<config name="TAGADAPTER">
    <dtomergeenabled>true</dtomergeenabled>
    <maxnumberofdecimals>4</maxnumberofdecimals>
    <maxsourceintervalinrequest>1</maxsourceintervalinrequest>
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
            <parameters>
                <parameter
                    name="BadFolder">C:\\temp\\jboss\\fileadapter\\bad</parameter>
                <parameter
                    name="CompletedFolder">C:\\temp\\jboss\\fileadapter\\completed</parameter>
                <parameter
                    name="DateFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
                <parameter
                    name="DateValueFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
                <parameter
                    name="DropFolder">C:\\temp\\jboss\\fileadapter</parameter>
                <parameter
                    name="ErrorFolder">C:\\temp\\jboss\\fileadapter\\error</parameter>
                <parameter name="FileFilter">merge</parameter>
                <parameter name="HaltOnMissingData">AVG</parameter>
                <parameter name="QualityColumnIndex">1</parameter>
                <parameter name="TagColumnIndex">4</parameter>
                <parameter name="TimeColumnIndex">0</parameter>
                <parameter name="ValueColumnIndex">2</parameter>
            </parameters>
        </sourceadapter>
    </config>

```

### **Max number of rows/samples per DTO**

The Message Driven Bean used in ECIS has a transaction timeout of 5 minutes. This means that if a DTO use more than 5 minutes to complete, wildfly will do a rollback and retry the DTO. To avoid this, you can reduce the size of the DTO so that processing it will take less than 5 minutes. To do this, use the parameter maxrowsindto. Set the parameter to the number of rows/samples you want per DTO, see example below:

```

<config name="TAGADAPTER">
    <dtomergeenabled>true</dtomergeenabled>
    <maxnumberofdecimals>4</maxnumberofdecimals>
    <maxsourceintervalinrequest>1</maxsourceintervalinrequest>
    <maxrowsindto>10</maxrowsindto>
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
            <parameters>
                <parameter
name="BadFolder">C:\\temp\\jboss\\fileadapter\\bad</parameter>
                <parameter
name="CompletedFolder">C:\\temp\\jboss\\fileadapter\\completed</parameter>
                <parameter
name="DateFormat ">yyyy-MM-dd 'T' HH:mm:ss</parameter>
                <parameter
name="DateValueFormat ">yyyy-MM-dd 'T' HH:mm:ss</parameter>
                <parameter
name="DropFolder">C:\\temp\\jboss\\fileadapter</parameter>
                <parameter
name="ErrorFolder">C:\\temp\\jboss\\fileadapter\\error</parameter>
                <parameter name="FileFilter">merge</parameter>
                <parameter name="HaltOnMissingData">AVG</parameter>
                <parameter name="QualityColumnIndex">1</parameter>
                <parameter name="TagColumnIndex">4</parameter>
                <parameter name="TimeColumnIndex">0</parameter>
                <parameter name="ValueColumnIndex">2</parameter>
            </parameters>
        </sourceadapter>
    </config>

```

### **Recapture range**

In a standard ECIS run, next\_read will be moved to after the last retrieved sample so that you do not get the same data over and over again. However, sometimes it is useful to recapture the same data. This is special for cases where you do not know when the data will arrive. To do this, use the parameter *recapturerange*. If you set *recapturerange* to 0, ECIS will set next\_read to the same timestamp as the latest timestamp in the request. If you set *recapturerange* to a number higher than 0, ECIS will set next\_read to the latest timestamp in the request minus (source\_interval \* *recapturerange*). In this way you can specify that I want to recapture x numbers of source\_intervals each time I run to make sure I get all my data. See example below for how to use:

```

<config name="ECISTESTSCENARIO_1">
    <dtomergeenabled>true</dtomergeenabled>
    <recapturerange>90</recapturerange>
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
        <parameters>
            <parameter
                name="BadFolder">C:\\temp\\jboss\\fileadapter\\bad</parameter>
            <parameter
                name="CompletedFolder">C:\\temp\\jboss\\fileadapter\\completed</parameter>
            <parameter
                name="DateFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
            <parameter
                name="DateValueFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
            <parameter
                name="DropFolder">C:\\temp\\jboss\\fileadapter</parameter>
            <parameter
                name="ErrorFolder">C:\\temp\\jboss\\fileadapter\\error</parameter>
            <parameter name="FileFilter"
                encrypted="true">3790abdb6b90a044f27ed80111697a510501354c9b33821b</parameter>
                <parameter name="QualityColumnIndex">2</parameter>
                <parameter name="TagColumnIndex">4</parameter>
                <parameter name="TimeColumnIndex">0</parameter>
                <parameter name="ValueColumnIndex">1</parameter>
            </parameters>
        </sourceadapter>
    </config>

```

### **Shift Time To Period Start**

When ECIS gather aggregated data, it assumes that the SCADA-system timestamps the sample at the beginning of the period. Some SCADA-systems will however timestamp the sample either somewhere in the middle, or at the end of the period. To address this problem, there is an attribute in TRANS TEMPLATE called SHIFT\_TIME\_TO\_PERIOD\_ST. This attribute can have 3 values, END\_INCLUDED, START\_INCLUDED or NO\_SHIFT. NO\_SHIFT is equivalent to null-value, and is the default value. ECIS will then assume that the timestamp of the sample is at the beginning of the period. The other values are explained here:

- END\_INCLUDED: When this value is used, ECIS will assume that the timestamp of the sample is either in the middle of the sample-period, or at the end of the sample-period, and all timestamps will be moved to the beginning of the period. If the timestamp is moved back in time so that it passes a DST-change, the offset will be used to correct the timestamp accordingly.
- START\_INCLUDED: When this value is used, ECIS will assume that the timestamp of the sample is either at the beginning of the sample-period, or in the middle. All timestamps which is not at the beginning of a sample-period will be moved so that they are.

### **Move next read if empty values**

When ECIS read tagdata, some SCADA-systems might return samples containing date, tagid but no value. Per default, ECIS will not move next read for these values. However in some cases one might want to update next read for these values as well to avoid them being read over and over again, which might cause a performance penalty. To do this, set the parameter movenextreadonempty to true, like in the example below.

```

<config name="ECISTESTSCENARIO_1">
    <dtomergeenabled>true</dtomergeenabled>
    <movenextreadonempty>true</movenextreadonempty>
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
        <parameters>
            <parameter
                name="BadFolder">C:\\temp\\jboss\\fileadapter\\bad</parameter>
            <parameter
                name="CompletedFolder">C:\\temp\\jboss\\fileadapter\\completed</parameter>
            <parameter
                name="DateFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
            <parameter
                name="DateValueFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
            <parameter
                name="DropFolder">C:\\temp\\jboss\\fileadapter</parameter>
            <parameter
                name="ErrorFolder">C:\\temp\\jboss\\fileadapter\\error</parameter>
            <parameter name="FileFilter"
                encrypted="true">3790abdb6b90a044f27ed80111697a510501354c9b33821b</parameter>
            <parameter name="QualityColumnIndex">2</parameter>
            <parameter name="TagColumnIndex">4</parameter>
            <parameter name="TimeColumnIndex">0</parameter>
            <parameter name="ValueColumnIndex">1</parameter>
        </parameters>
    </sourceadapter>
</config>

```

### **Using sourcemapper to modify values**

It is possible to apply one or several mappers to a config. The mapper must implement the interface com.ec.frmw.is.engine.source.mapper.I\_SrcMapper which has the method **public List<I\_Sample> convert(List<I\_Sample> samples);**

Inside this method, the different implementations can modify the samples as they want. To apply these mappers to the config, add the parameter **srcmapper** like shown in the example below. This should contain a list of mappers separated by ;.

```

<config name="PI_SOURCE_1">

    <srcmapper>com.ec.frmw.is.engine.source.mapper.impl.PiMapperDigitalValuesYorN</srcmapper>
        <sourceadapter>

            <class>com.ec.frmw.is.engine.adapter.pi.jinterop.PiAdapter</class>
                <sequential>Yes</sequential>
                <parameters>
                    <parameter name="authdomain">MyDomain</parameter>
                    <parameter name="domainuser">auser</parameter>
                    <parameter name="domainpwd">password</parameter>
                    <parameter name="dsn">127.0.0.1</parameter>
                    <parameter name="username">piadmin</parameter>
                    <parameter name="pwd"></parameter>
                    <parameter name="timeout">180</parameter>
                    <parameter name="logfile">c:\temp\logfile2.log</parameter>
                </parameters>
            </sourceadapter>
        </config>

```

### **Retrieve digital values from pi.**

It is not possible to retrieve digital values directly through ECIS, however, there are two sourcemappers which lets you convert the incoming values from a digital tag to either Y/N or 0/1:

```

com.ec.frmw.is.engine.source.mapper.impl.PiMapperDigitalValuesYorN
com.ec.frmw.is.engine.source.mapper.impl.PiMapperDigitalValues1or0

```

To use one of these, apply them as described in the section about sourcemappers. If you are using the one with Y/N, set source datatype to STRING, if you are using 1/0, set source datatype to DECIMAL. ECIS will then retrieve the values from PI, and the mapper will convert them to the appropriate format.

### **Operation**

#### **Monitoring the jobs**

As shown in previous sections, the EC IS separates source and target systems. The two main components can run asynchronous from each other. The source system can run in parallel at the same time the target system is processing messages from another schedule. Hence the two processes are monitored in two contexts. Since the source system is started from the scheduler, monitoring this process can be done by the scheduler's own mechanisms. Target system is triggered by the DataTransferMDB and can be monitored by its own screens.

#### **Monitoring extraction by THE Scheduler**

Scheduling an extraction through the source system may involve different process settings. One of the settings relate to Log Level. Depending on the log level, the extraction will produce the information requested. Each time a schedule start, a process instance is produced in the Scheduler History Log. All available information about the specific extraction can be found here. If an extraction process fail, a log will show the problems which have arisen. Refer to the *EC Framework – Configuration Manual* to read more about how to use the history log.

#### **Monitoring transform and load by error log**

The first monitoring screen for the target system, is a common log, trans\_process\_log which will show all errors from ECIS target system.

From Daytime: 2014-12-01 06:09 To Daytime: 2014-12-11 07:09

Service name	Log Level	Daytime	Log Message
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T04:00:00)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T02:25:00)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:50:00)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:31:00)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:25:00)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:21:00)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	Error occurred when running sql insert into DV_PWEVL_DAY_STATUS(DAYTIME,OBJECT_ID,AVG_BH_PRESS,created_by) values (ts_date,2014-12-02T01:20:31)
DataTransferAction	ERROR	2014-12-02 04:00	An error occurred with one of the elements in the batch for [PWEVL_DAY_STATUS]. Error from system: ORA-00001: unique constraint (ECKERNEEL_GADE_KL_RC

In addition there is a tag data capture monitoring screen that will show a list of the latest 100 rows which has been inserted/updated by the Data capture process.

Last Transfer	Class	PK Value	Daytime
2014-12-05 04:38:12.0	PWEVL_DAY_STATUS	ASL_UNR_202	2014-12-05 00:00:00
	TAG_EVENT_STATUS_3	DUMMY_OBJECT_TAG_EVENT_STAT	2014-11-20 00:00:00
	TAG_EVENT_STATUS_1	DUMMY_OBJECT_TAG_EVENT_STAT	2014-11-20 00:00:00
	TAG_EVENT_STATUS_2	DUMMY_OBJECT_2TAG_EVENT_ST	2014-11-20 00:00:00
	TAG_EVENT_STATUS_1	DUMMY_OBJECT_2TAG_EVENT_ST	2014-11-20 00:00:00

The intention is to provide a good picture of the status for ongoing data capture, and to help detect if there is a stop in the data flow. It will also give an indication of the performance in a catch-up situation. There are future plans for extending EC IS with complementary monitoring screens, including status on auto-recovery, and recapture for limited periods back in time. Today recapture is handled by setting back the *last\_transfer* parameter.

Since EC IS is asynchronous and use a jms-queue as a middle storage, it can be useful to know that it is done processing all incoming data. This can be monitored in wildfly's management console like in the screenshot below:

Name	JNDI
ExpiryQueue	[java:/jms/queue/ExpiryQueue]
DLQ	[java:/jms/queue/DLQ]
ECISQueue	[queue/ECISQueue]
ECMHMInboundQueue	[queue/ECMHMInboundQueue]

## Recovery principles

EC IS will try to recover from different failure scenarios. The mechanisms behind this are persistence of data and states between the internal services.

1. If an extraction fails it will try to extract the data again at the next execution.
2. If the transformation fails to start, extracted data is still present waiting for the transformation to start processing.
3. If the transformation stops (system crash, system shutdown, lost computer power), aggregated data is still persisted, so the transformation can continue with the extracted data when it's up and running again.
4. If a database update fails for a specific row, it will write an error log instance and continue updating the other rows.

In some failure scenarios it will be necessary to get input from the user to specify recapture of tag data explicitly. If the mapping rules in EC IS don't match the EC class configuration, it will produce an error log specifying what to fix. It will also cause the last transfer date for the specific mapping to remain unchanged. The extraction processes may still continue reading new samples. To repair the mapping and synchronize the data extraction, the user can use the CO.1017: *Maintain Templates* and CO.1018: *Maintain Mappings* to repair mappings and to adjust the last transfer date for the specific mapping.

Note that if you are setting back the last transfer date, this must be done in a period when the scheduled data extraction is not running, otherwise you risk that your adjustment will be ignored. If you are running with tight intervals on the scheduled job, it is probably safest to stop the job before you adjust the last transfer date.

## Appendix A – Tag / Row Services Configuration Details

All of the adapters need to be configured in the file /<wildfly>/domain/servers/ec-server-a1/configuration/ecisconfig.xml. This file should contain a set of config instances. Each config instance has a source adapter related to it. The config instances share the same structure; an attribute uniquely identifying the config, a class element referring to the path to the class implementing the adapter, a sequential element indicating whether the values to retrieve is ordered by date and a set of parameter elements. In addition there are two parameters which are controlled by the source functions used. The first one indicates that the system should halt (not proceed) if data cannot be found for the requested period. The same interval will be requested over again at next run. The next parameter is used to modify the timestamp from a source function to the start of each period. This is often used if a specific function timestamps at the end instead of the at the start of period.

```

<configurations><--PI configuration start -->
  <config name="PI_SOURCE_1">
    <sourceadapter><-- path to implementing class-->

      <class>com.ec.frmw.is.engine.adapter.pi.PiJavaAdapter</class><-- is
      the values order by date when retrieved from PI OLEDB -->
        <sequential>Yes</sequential>
        <HaltOnMissingData>MAX,MIN</HaltOnMissingData>
        <RetryTimeout>3600</RetryTimeout><-- set of parameters
      specific for this adapter-->
        <parameters>
          <parameter name="dsn">127.0.0.1</parameter>
          <parameter name="username">piadmin</parameter>
          <parameter name="pwd"></parameter>
          <parameter name="timeout">180</parameter>
          <parameter
name="logfile">c:\temp\logfile2.log</parameter>
          <parameter name="DEFAULT_RECORD_STATUS">A</parameter>
        </parameters>
      </sourceadapter>
    </config><--PI configuration end --><--File adapter configuration
start -->
    <config name="FILEADAPTER">
      <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
          <sequential>Yes</sequential>
          <parameters>
            <parameter name="DropFolder">c:\\temp</parameter>
            <parameter
name="CompletedFolder">c:\\\\temp\\\\completed</parameter>
            <parameter
name="ErrorFolder">c:\\\\temp\\\\error</parameter>
            <parameter name="BadFolder">c:\\\\temp\\\\bad</parameter>
            <parameter name="DateFormat">dd.MM.yyyy
HH:mm:ss</parameter>
            <parameter name="FileFilter"></parameter>
            <parameter name="QualityColumnIndex">0</parameter>
            <parameter name="TagColumnIndex">1</parameter>
            <parameter name="TimeColumnIndex">2</parameter>
            <parameter name="ValueColumnIndex">3</parameter>
            <parameter name="DEFAULT_RECORD_STATUS">V</parameter>
          </parameters>
        </sourceadapter>
    </config><--File adapter configuration end-->
  </configurations>

```

Some parameters/ element types are valid for all tag adapters:

Parameter/ element	Value	Comments
config name	Unique string	Identifier of the configuration. Must be added as EC codes with the type DT_SOURCE_ID.
class (mandatory)	<path to implementation>	Referring to the class of the adapter
Sequential (optional)	Yes   True OR No   False	Indicates if the data coming from the adapter can be expected to be ordered by date / time.
HaltOnMissingData (optional)	Comma separated list of source functions	Each of the functions listed will be checked if data is missing for the requested period. If no data is found, the same period will be requested at the next run.
ShiftTimeToPeriodStart (optional) <DEPRECATED, use attribute in TRANS_TEMPLATE instead>	Comma separated list of source functions	For each of the functions listed the timestamp will be moved one period back. This is commonly used if the timestamp is set at the end of a period for a specific function. The parameter will indicate that it should be moved to the start of period.
DEFAULT_RECORD_STATUS(optional)	P   V   A	The recordstatus on the values retrieved from the adapter. Valid values are: <ul style="list-style-type: none"> <li>• P (provisional)</li> <li>• V (verified)</li> <li>• A (approved)</li> </ul> If no value is set the records status originating from this adapter will be P. If the value differs from P this must also be changed in the db; trans_template.overwrite_status must be set to the same value for all affected templates.
RetryTimeout	Number of seconds.	How many seconds the timeout for the failover mechanism of the adapters should be. Only needed if you plan on using several adapters with the same configuration name for failover.

To enable failover mechanism for ECIS adapters, you can create multiple configurations with the same config name. EC IS will then use the first one as the default adapter. If this adapter fails during initialize, EC IS will use the next adapter until there are no more adapters left. If the last specified adapter fails while running, EC IS will fail the schedule and the next schedule that runs will use the next adapter with the same configuration name.

If an adapter fails EC IS will wait for a timeout. When this timeout occurs the next schedule that runs will use the first adapter once again. If this adapter fails it will proceed to the next adapter in line etc. How long this timeout should be is specified by the adapter RetryTimeout parameter which can be set as in the example above. This parameter specifies the amount of seconds the timeout should be. It is recommended that this timeout is large enough that many schedules can run before it occurs.

## PI Adapter

### **Required PI Middleware.**

To be able to connect to PI the EC application server need the following PI products installed:

PI SDK (Tested with version 1.4.0.416)



PI OLEDB (Tested with version 3.3.1.2)

For further information on the PI OLEDB software requirements, please refer to the documentation from OSIsoft.

### **Source system configuration.**

We have three different implementations of the PI adapter. Our recommended solution is to use the JDBC version, which use PI JDBC driver provided with PI SQL DAS. We also have one adapter using MS SQL Express server connecting to PI OLEDB, and one using java - com bridge J-Interop to do the same job.

#### **PI JDBC**

The recommended approach for connecting ECIS to PI is to use PI JDBC driver. Following is an example of how this adapter can be configured:

```
<config name="PI_SOURCE_1">
    <sourceadapter>
        <class>com.ec.frmw.is.engine.adapter.pi.jdbc.PiAdapter</class>
        <sequential>Yes</sequential>
        <parameters>
            <parameter
name="dbdriver">com.osisoft.jdbc.Driver</parameter>
            <parameter name="connecturl">jdbc:pisql://<PI-server>;
/Data Source=<PI-server>; Integrated Security=SSPI;</parameter>
            <parameter name="username">piadmin</parameter>
            <parameter name="pwd"></parameter>
            <parameter name="timezone">Asia/Singapore</parameter>
            <parameter name="avg_calcbasis">EventWeighted</parameter>
        </parameters>
    </sourceadapter>
</config>
```

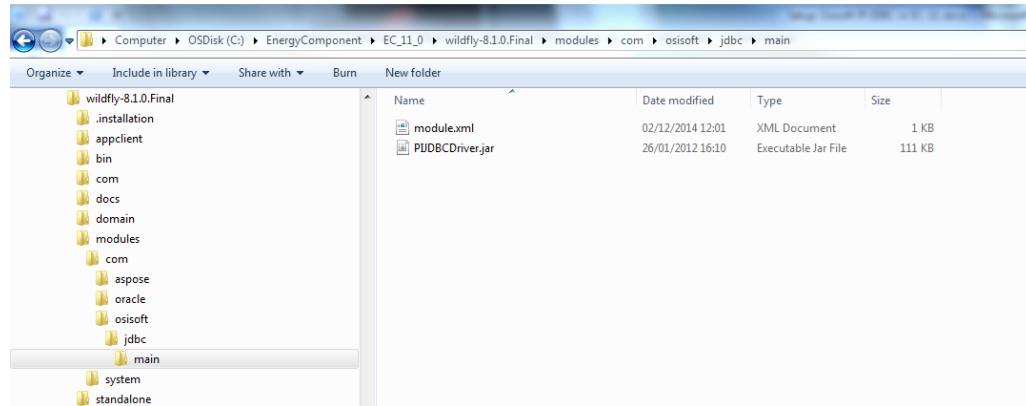
Parameter name	Value	Comments
class (mandatory)	com.ec.frmw.is.engine.adapter.pi.jdbc.PiAdapter	Referring to the class of the adapter.
Sequential (optional)	Yes   True OR No   False	Indicates if the data coming from the adapter can be expected to be ordered by date / time.
connecturl (mandatory)	jdbc:pisql://<PI-server>/Data Source=<PI-server>; Integrated Security=SSPI; where <PI-server> is the machine name of the PI server	The connection string used to connect to the PI server.
username (mandatory)	<PIUser>	User id for logging into the PI server.
pwd (mandatory)	<PIPPassword>	Password for logging into the PI server.
timezone (mandatory)	<TimeZone>	PI server time zone (e.g. Asia/Singapore, CET & etc)

avg_calcbasis (optional)	<TimeWeighted>	PI AVG aggregation calculation function methods (e.g. TimeWeighted, EventWeighted & etc. Please refer to PI-OLEDB-Provider help file from osisoft)
max_calcbasis (optional)	<TimeWeighted>	PI MAX aggregation calculation function methods (e.g. TimeWeighted, EventWeighted & etc. Please refer to PI-OLEDB-Provider help file from osisoft)
min_calcbasis (optional)	<TimeWeighted>	PI MIN aggregation calculation function methods (e.g. TimeWeighted, EventWeighted & etc. Please refer to PI-OLEDB-Provider help file from osisoft)
sum_calcbasis (optional)	<TimeWeighted>	PI SUM aggregation calculation function methods (e.g. TimeWeighted, EventWeighted & etc. Please refer to PI-OLEDB-Provider help file from osisoft)
check-valid-connection-sql (optional)		SQL-statement that will be run upon startup of the adapter. This should return a record if there is a connection to the db, and none if not. (Do not use count★ in sql, since this will always return something)
logfile (optional)	c:\temp\PIJDBClogfile.log	Location of log file for PI adapter.
pctgood	<Integer stating how many percent>	For aggregated queries, how many percent is considered good quality in PI? If this parameter is set, ECIS will add a check saying AND PCTGOOD >= <given value> to aggregated queries. Value should be between 0 and 100.
statuschecklatest	True/false	If set to true, latest queries against PI will check for status=0. Default set to false.

This approach requires the installation of the PI JDBC driver. For further information on the PI JDBC software requirements and installation procedures, please refer to the documentation from OSISoft. PI JDBC is part of Osisofts PI SQL Data Access Server (PI SQL DAS).

It is assumed that JDBC has already been installed. Please take note of the following steps:

1. Check the installation of PI SQL DAS and PI JDBC by locating the PIPC directory which is by default in c:\Program Files.
2. Verify that there is a folder PIPC\JDBC containing the file PIJDBCDriver.jar (platform independent).
3. Also verify the file RDSAWrapper.dll (native code library) is installed. (Take note that, if used with a 64 bit Java VM the names were RDSAWrapper64.dll).
4. Create a global module for OsiSoft PI JDBC inside the JBoss server. (e.g /<JBoss>/modules/com/osisoft/jdbc/main folder) as below screenshot.



- copy the PIJDBCDriver.jar file from installed PI JDBC folder (e.g. C:\Program Files\PIPC\JDBC) into above newly created global module folder.
- create a module.xml and save it into above newly created global module folder as below screenshot.

```

<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.osisoft.jdbc">
<resources>
<resource-root path="PIJDBCDriver.jar"/>
<resource-root path=". "/>
</resources>
</module>

```

- Modify the <JBoss>/domain/configuration/domain.xml to add an global module entry for the OsiSoft PI JDBC under the <global-modules> section as below screenshot.

```

<driver name="oracle-jdbc-xa" module="com.oracle.jdbc7_g">
<xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
</driver>
</drivers>
</datasources>
</subsystem>
<subsystem xmlns="urn:jboss:domain:ee:2.0">
<global-modules>
<module name="com.aspose.cells" slot="main"/>

</global-modules>
<spec-descriptor-property-replacement>true</spec-descriptor-property-replacement>
<concurrent>
<context-services>

```

### Supported source functions

The following source functions are supported in the PI OLE DB adapter:

Function name	Description
SAMPLE	Reads raw samples from the PIArchive. Can be slow when used on tags with high sampling frequency.
AVG	Timeweighted average for given period using linear interpolation.
MAX	Maximum tag value within given period
MIN	Minimum tag value within given period
MEAN	Plain average of all values within given period
VALUE_AT_START	Values will be the last event before each period change.
VALUE_AT_END	Value will be the last event in each period.
COMPRESSED	Interpolated values from pi

In the mapping configuration, these can be set up as the **Source Function** in the Maintain template screen. (CO.1017: *Maintain templates*).

The extraction of these data in the adapter is done with selects against the tables in the *piarchive*.

Under is the Pseudo code for the Select related to each function. The implementation can be slightly different, but the queries illustrates that there is not much function specific logic in the adapter. The adapter is using the aggregation functionality provided in PI.

#### **SAMPLE**

```
SELECT tag,CAST(value AS Float64),time FROM piarchive..picomp2 WHERE tag = '<tagid>' AND status = 0 AND time >= DATE('<fromDate>') AND time < DATE('<toDate>')"
```

#### **AVG**

```
SELECT tag,CAST(value AS Float64),time FROM piarchive..piavg WHERE tag = '<tagid>' AND time >= DATE('<fromDate>') AND time < DATE('<toDate>') AND timestep = '<timeStep>'
```

#### **MIN**

```
SELECT tag,CAST(value AS Float64),time FROM piarchive..pimin WHERE tag = '<tagid>' AND time >= DATE('<fromDate>') AND time < DATE('<toDate>') AND timestep = '<timeStep>'
```

#### **MAX**

```
SELECT tag,CAST(value AS Float64),time FROM piarchive..pimax WHERE tag = '<tagid>' AND time >= DATE('<fromDate>') AND time < DATE('<toDate>') AND timestep = '<timeStep>'
```

#### **MEAN**

```
SELECT tag,CAST(value AS Float64),time FROM piarchive..pimean WHERE tag = '<tagid>' AND time >= DATE('<fromDate>') AND time < DATE('<toDate>') AND timestep = '<timeStep>'
```

#### **SUM**

```
SELECT tag,CAST(value AS Float64),time FROM piarchive..pitotal WHERE tag = '<tagid>' AND time >= DATE('<fromDate>') AND time < DATE('<toDate>') AND timestep = '<timeStep>'
```

#### **COMPRESSED**

```
SELECT tag,CAST(value AS Float64),time FROM piarchive..piinterp2 WHERE tag = '<tagid>' AND time >= DATE('<fromDate>') AND time < DATE('<toDate>') AND timestep = '<timeStep>'
```

#### **Jinterop**

This adapter uses the jinterop bridge for talking to PI. It needs to authenticate against the domain or the dom server it is querying. Here are an example of how it can look.

```

<config name="PI_SOURCE_1">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.pi.jinterop.PiAdapter</class>
        <sequential>Yes</sequential>
        <parameters>
            <parameter name="authdomain">MyDomain</parameter>
            <parameter name="domainuser">auser</parameter>
            <parameter name="domainpwd">password</parameter>
            <parameter name="dsn">127.0.0.1</parameter>
            <parameter name="username">piadmin</parameter>
            <parameter name="pwd"></parameter>
            <parameter name="timeout">180</parameter>
            <parameter name="logfile">c:\temp\logfile2.log</parameter>
        </parameters>
    </sourceadapter>
</config>

```

Parameter name	Value	Comments
class (mandatory)	com.ec.frmw.is.engine.adapter.pi.jinterop.PiAdapter	Referring to the class of the adapter
Sequential (optional)	Yes   True OR No   False	Indicates if the data coming from the adapter can be expected to be ordered by date / time.
authdomain (mandatory)	<Domain>	The domain the adapter is going to be authenticated towards. This could be a windows domain or a host.
domainuser (mandatory)	<Username>	Username to a user with privileges to call the PI OLE DB dll. This user needs to have dcom access to the jboss server. More info can be found here: <a href="http://j-interop.org/quickstart.html">http://j-interop.org/quickstart.html</a> .
domainpwd (mandatory)	<Password>	Password to a user with privileges to call the PI OLE DB dll.
domhost (optional)	<host>	An optional parameter for use when the PI OLE DB dll is exposed as an executable on an other host. If the parameter is not present or set to an empty string the adapter will contact "localhost".
dsn (mandatory)	<PI Server Machine>	Machine name or IP address of the PI server
username (mandatory)	<PiUser>	User id for logging in to the PI server.
pwd	<PiPassword>	Password for logging into the PI server.
timeout	180	Seconds to wait for a request to PI
commandtimeout	<seconds>	Sets the command timeout. This will override command timeout set by timeout parameter

connectiontimeout	<seconds>	Sets the connection timeout. This will override connection timeout set by the timeout parameter
logfile	c:\temp\logfile2.log	Location of log file for PI adapter.
usentlmv2	true/false	Use NTLMv2 instead of NTLMv1
avg_calcbasis (optional)	<TimeWeighted>	PI AVG aggregation calculation function methods (e.g. TimeWeighted, EventWeighted & etc. Please refer to PI-OLEDB-Provider help file from osisoft)
max_calcbasis (optional)	<TimeWeighted>	PI MAX aggregation calculation function methods (e.g. TimeWeighted, EventWeighted & etc. Please refer to PI-OLEDB-Provider help file from osisoft)
min_calcbasis (optional)	<TimeWeighted>	PI MIN aggregation calculation function methods (e.g. TimeWeighted, EventWeighted & etc. Please refer to PI-OLEDB-Provider help file from osisoft)
sum_calcbasis (optional)	<TimeWeighted>	PI SUM aggregation calculation function methods (e.g. TimeWeighted, EventWeighted & etc. Please refer to PI-OLEDB-Provider help file from osisoft)
pctgood	<Integer stating how many percent>	For aggregated queries, how many percent is considered good quality in PI? If this parameter is set, ECIS will add a check saying AND PCTGOOD >= <given value> to aggregated queries. Value should be between 0 and 100.
statuschecklatest	True/false	If set to true, latest queries against PI will check for status=0. Default set to false.

This adapter requires that the user setup to connect to PI is given DCOM privileges, so that it can fetch the PI OLEDB com object from Windows Com Server. Futher information on how to configure dcom privileges can be found at: <http://j-interop.org/quickstart.html>. Be aware that PI J-interop require PI SDK 1.3.8.388 or earlier version.

#### MS SQL Linked Server

Another approach for connecting ECIS to PI is to use MS SQL Linked Server. In this case the MS SQL Server will act as a bridge between ECIS and PI, thus making it possible to query data using a JDBC connection instead of having to depend on COM or DCOM. Following is an example of how this adapter can be configured:

```

<config name="PI_SOURCE_1">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.pi.jdbc.PiJdbcAdapter</class>
            <sequential>Yes</sequential>
            <parameters>
                <parameter
name="dbdriver">net.sourceforge.jtds.jdbc.Driver</parameter>
                <parameter
name="connecturl">jdbc:jtds:sqlserver://localhost:1433</parameter>
                    <parameter name="username">sa </parameter>
                    <parameter name="pwd"></parameter>
                    <parameter name="linkedserver">PI</parameter>
                </parameters>
            </sourceadapter>
    </config>

```

Parameter name	Value	Comments
class (mandatory)	com.ec.frmw.is.engine.adapter.pi.jdbc.PiJdbcAdapter	Referring to the class of the adapter
connecturl (mandatory)	jdbc:jtds:sqlserver://<host>:<port>, where <host> is a hostname or IP address and <port> is a number	The connection string used to connect to the MS SQL Server
username (mandatory)	<Username>	Username to a user with privileges to log on the MS SQL Server
pwd (mandatory)	<Password>	Password to a user with privileges to log on the MS SQL Server
linkedserver (mandatory)	<LinkedServer>	Name of the linked server
check-valid-connection-sql (optional)		SQL-statement that will be run upon startup of the adapter. This should return a record if there is a connection to the db, and none if not. (Do not use count  in sql, since this will always return something)
pctgood	<Integer stating how many percent>	For aggregated queries, how many percent is considered good quality in PI? If this parameter is set, ECIS will add a check saying AND PCTGOOD >= <given value> to aggregated queries. Value should be between 0 and 100.
statuschecklatest	True/false	If set to true, latest queries against PI will check for status=0. Default set to false.

In addition, the MS SQL Server must be configured as a linked server. The following steps shows how this is done for *MS SQL Server Express Edition*. The process should however be similar independent on which version of MS SQL Server you have. It is assumed that a MS SQL Server has already been installed and is available:

1. Start *Microsoft SQL Server Management Studio* (if this is not available it needs to be installed). In the Object Explorer window expand the Server Objects folder and right click on Linked Servers. Select **New Linked Server** (see figure 5.1). Fill in the information as per pictures below. Make sure to also provide the Provider string, to ensure you get the samples with correct timestamps.

2. In the *New Linked Server* window that appears, select **General** page and fill in the fields according to figure 5.2. Please note that the name of the Linked server must correspond with the *linkedserver* parameter in the configuration above. Furthermore, the value of Data source should be hostname or IP address of the PI server. Also, the Log File location, Log Level and Command Timeout parameters can be changed in the Provider string, if required. The other parameters should be kept as in figure 5.2.
3. In the **Security** page in the *New Linked Server* window, select '**Be made using this security context:**' and provide a username and password with access to the PI server (see figure 5.3). For other ways of authenticating against the PI server, please refer to *Chapter 8 – MS SQL Server Linked Server* in the *PI OLEDB Data Provider for the PI System*.
4. In the **Server Options** page set the parameters as according to figure 5.4.

You should now be able to connect to PI using the MS SQL Server (JDBC) adapter in ECIS. Please refer to *Chapter 8 – MS SQL Server Linked Server* in the *PI OLEDB Data Provider for the PI System* for further information on how to run a MS SQL Server as a Linked Server.

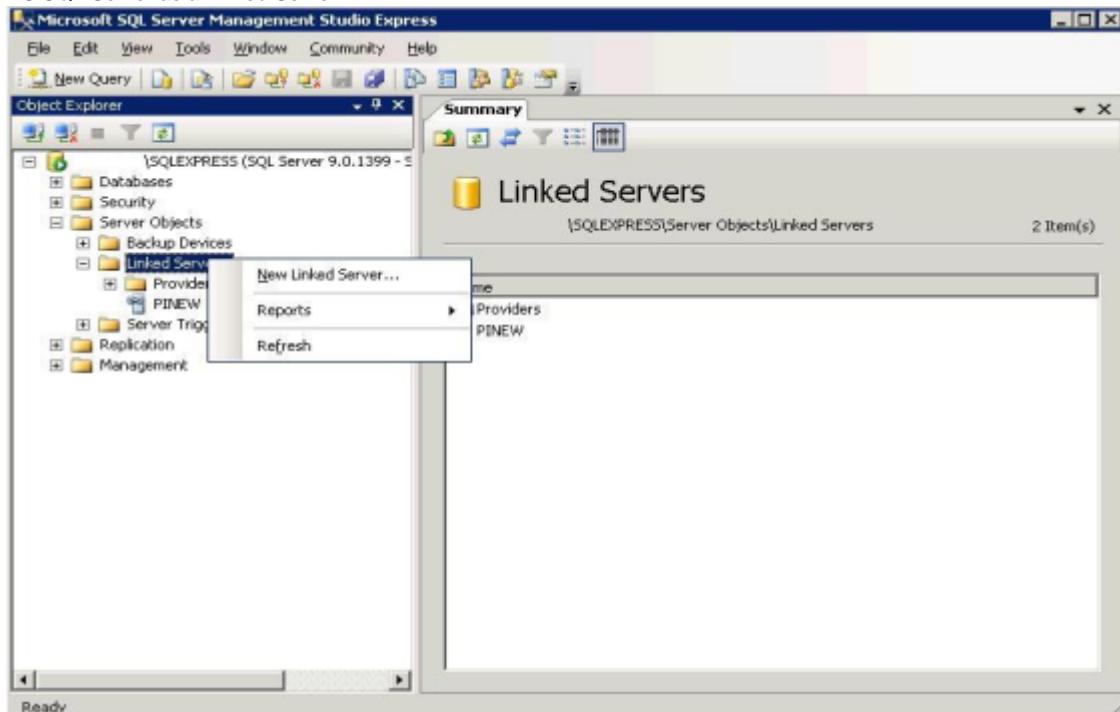


Figure 5.1: Right click on Linked Server and select New Linked Server

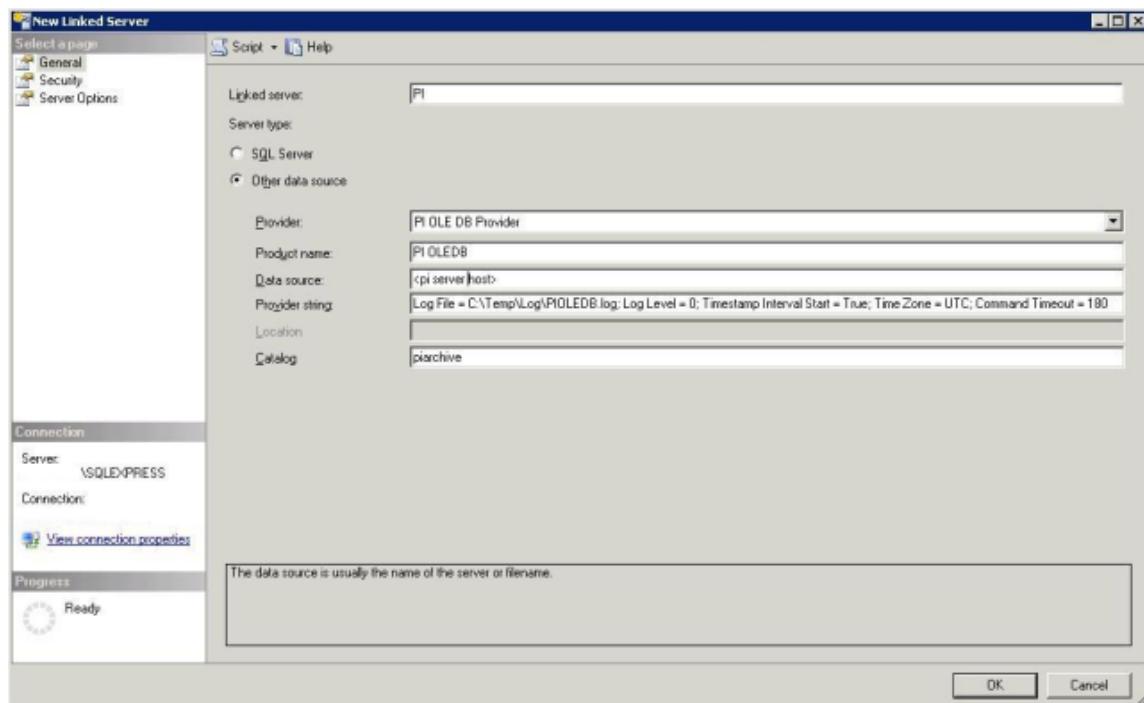


Figure 5.2: General page

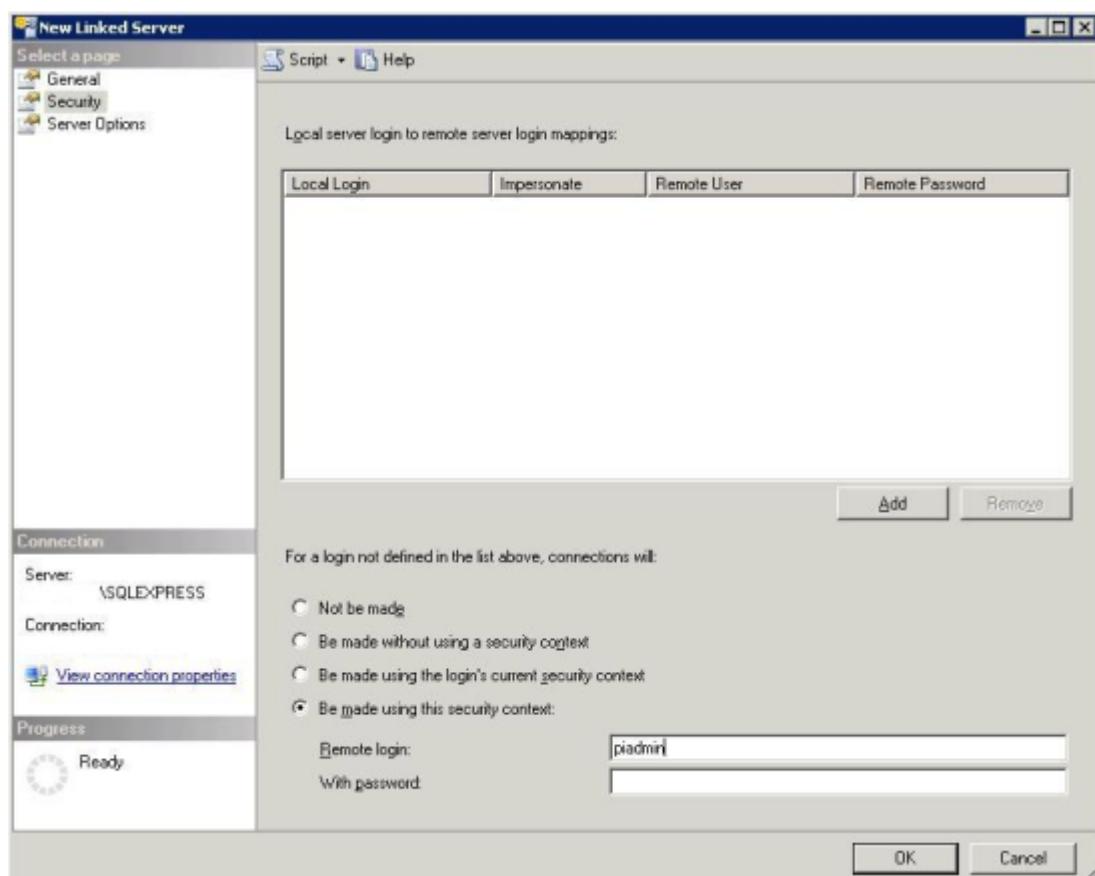


Figure 5.3: Security page

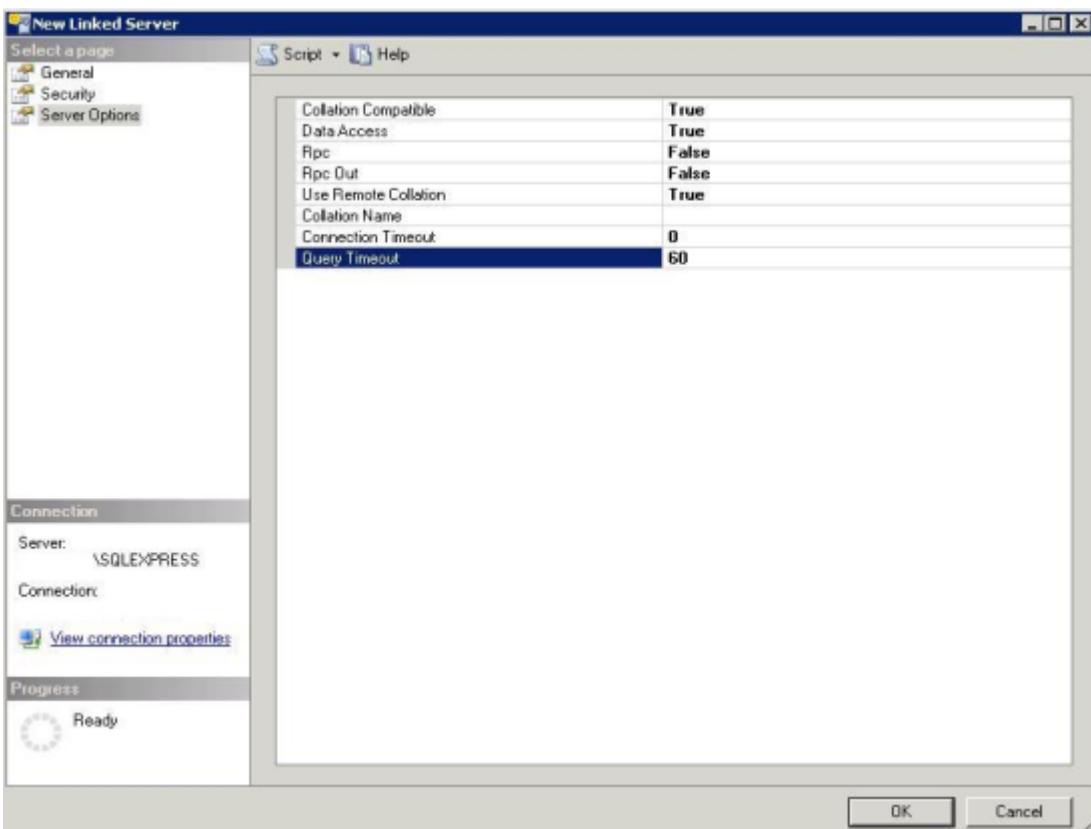


Figure 5.4: Server Options page

### Troubleshooting

In case of problems using this adapter or configuring the MS SQL Server as a Linked Server, try the following:

#### TCP/IP for SQL Server Express

By default, TCP/IP for SQL Server Express is disabled, so JDBC cannot connect to it and you may get the following exception: *Network error IOException: Connection refused: connect*

To enable TCP/IP, start *SQL Server Configuration Manager*.

1. Expand *SQL Server 2005 Network Configuration* node.
2. In the right pane, select *Protocols for SQLEXPRESS*. The right pane should now show *Protocols* and *Status* columns.
3. Select *Enable* from the *TCP/IP* context menu.

#### Find or configure TCP/IP Port

After enabling TCP/IP, you have to find out which port number to use. *SQL Server Express* allocates a port dynamically each time it is started, so to find or configure the port number, continue using *SQL Server Configuration Manager* ...

1. Select *Properties* from the *TCP/IP* context menu. The *TCP/IP Properties* dialog should open.
2. Select the *IP Addresses* tab.
3. In the *IPAll* node ...
  - a. The *TCP Dynamic Ports* field shows the currently used port number. If you set that field to blank, then SQL Server Express should not automatically choose another port when it restarts.
  - b. Set the desired port number in the *TCP Port* field.
  - c. Press *OK* to apply your settings and close the dialog.

#### Test TCP/IP

If you change the TCP/IP port, you have to restart *SQL Server Express* before it can use the new port number. To test that your port number is used, start a *cmd* window and type: *netstat -an*. For instance, if you used port 1433, you should see this line in the list of ports used:

TCP 0.0.0.0:1433 0.0.0.0:0 LISTENING

### **Authentication Method**

By default, SQL Server Express uses *Windows Authentication Mode* to authenticate connections. If you get this exception: *Login failed for user '<User name>'. The user is not associated with a trusted SQL Server connection*, then you may have to enable *SQL Server Authentication Mode* and create or enable a user.

1. Start *Microsoft SQL Server Management Studio Express (SSMSE)* and connect to your database server.
2. In *Object Explorer* pane, select *Properties* from your database's context menu. The *Server Properties* dialog should open.
3. Select *Security* page.
4. Select *SQL Server and Windows Authentication Mode* check box.
5. Press *OK* button to close the dialog.
6. In *Object Explorer* pane, expand *Security / Logins* node.
7. Select existing user *sa*. Note that there is a red downward arrow next to that user's image.
8. View *sa*'s properties. The *Login Properties* dialog should open.
9. Select *Status* page.
10. Ensure that the *Login: Enabled* radio button is selected.
11. Select *General* page.
12. Enter a password for this user.
13. Press *OK* button to close the dialog.
14. If you refresh the *Object Explorer* pane, note that user *sa* no longer has a red downward arrow.

### **OPC Adapter**

#### **Required Software and Configuration**

The OPC Adapter attempts to contact an OPC server over the OPCHDA protocol, hence the OPC server must be OPCHDA 1.0/1.1 compliant and expose the DCOM interfaces with the correct security constraints. The configured user must have com privileges to launch/activate and access the interface from the node the adapter is running. Futher information on how to configure dcom privileges can be found at: <http://j-interop.org/quickstart.html>.

#### **Source System Configuration**

This must be configured in the file /<JBoss>/domain/servers/ec-server-a1/configuration/ecisconfig.xml

```
<config name="OPC_SOURCE_1">
    <sourceadapter>
        <class>com.ec.frmw.is.engine.adapter.opc.OPCAdapter</class>
        <sequential>Yes</sequential>
        <parameters>
            <parameter name="AUTH_DOMAIN">MyDomain</parameter>
            <parameter name="USR">opcreader</parameter>
            <parameter name="PWD">password</parameter>
            <parameter name="OPC_HOST">127.0.0.1</parameter>
            <parameter name="PROG_ID">Some.OPC.Server.1</parameter>
            <parameter name="IID"></parameter>
            <parameter name="USE_PROG_ID">true</parameter>
        </parameters>
    </sourceadapter>
</config>
```

In the section (config name) refereeing to the OPC adapter, the following must be configured

Parameter name	Value	Comments
class (mandatory)	com.ec.frmw.is.engine.adapter.opc.OPCAdapter	Referring to the class of the adapter
AUTH_DOMAIN (mandatory)	<Domain>	The domain the adapter is going to be authenticated towards. This could be a windows domain or a host.
USR (mandatory)	<OPC Username>	Username to a user with privileges to call the OPC server.
PWD (mandatory)	<OPC Password>	Password to a user with privileges to call the OPC server.
OPC_HOST (mandatory)	<OPC Host>	The dns/ip to the server responding to OPCHDA requests
PROG_ID	<OPC progId>	The progId for the com interface exposed by the server. If this value is not provided, the IID parameter must be provided.
IID	<OPC clsId>	The clsId for the com interface exposed by the server. If this value is not provided, the progId parameter must be provided.
USE_PROG_ID	True   False	Only used if both PROG_ID and IID parameters have value. Used to determine which field to use.
ADD_SHUTDOWN_CALLBACK	"ADD_SHUTDOWN_CALLBACK"	Optional field, when present the client will register itself to the server in order to receive a callback upon server shutdown. The specification states this callback is needed to enable a clean shutdown, but as the adapter frees up resources for each call, this is not necessary in the current implementation. However some servers (Rapid) might require the client to register using the interface.
goodQuality	List of integers separated by ,	If this attribute is specified, ECIS will only accept the integers given in this list as good quality.
badQuality	List of integers separated by ,	If goodQuality is not given, you can give a list of bad qualities. ECIS will then throw away all samples having a quality value like the ones given in this list. If neither goodQuality or badQuality is given, ECIS will accept all samples.
usentlmv2	true/false	Use NTLMv2 instead of NTLMv1

The parameter values must be changed to match the local infrastructure with machine names and user ids.

### **Supported Source Functions**

The following source functions are supported as specified in the OPCHDA specification. However this can be configured to use other values if the OPC server has vendor specific aggregate functions.

Function name	Description
SAMPLE	Reads raw samples from the OPC Server. Can be slow when used on tags with high sampling frequency.
AVG	Timeweighted average for given period using linear interpolation. Default configuration mapped up to OPCHDA TIMEAVERAGE
MAX	Maximum tag value within given period Default configuration mapped up to OPCHDA MAXIMUM ACTUAL TIME
MIN	Minimum tag value within given period. Default configuration mapped up to OPCHDA MINIMUM ACTUAL TIME
MEAN	Not supported in default configuration. But can be configured if the OPC server support this aggregate.
SUM	Not supported in default configuration. But can be configured if the OPC server support this aggregate.
VALUE_AT_START	Values will be the last event before each period change.
VALUE_AT_END	Value will be the last event in each period.

#### CONFIGURING SOURCE FUNCTIONS

As mention above the OPC Adapter use the standard aggregate functionality as specified by the OPC foundation without any extra configuration. But if different aggregate ids are needed these can be mapped up in the sourceadapter configuration described in section Source System Configuration. And the default mapping could be overrided by providing an aggregate id the OPC server supports like the example below shows:

```

<config name="OPC_SOURCE_1">
    <sourceadapter>
        <class>com.ec.frmw.is.engine.adapter.opc.OPCAdapter</class>
        <parameters>
            <parameter name="AUTH_DOMAIN">MyDomain</parameter>
            <parameter name="USR">opcreader</parameter>
            <parameter name="PWD">password</parameter>
            <parameter name="OPC_HOST">127.0.0.1</parameter>
            <parameter name="PROG_ID">SOME.OPC.Server.1</parameter>
            <parameter name="FUNCTION_AVG">4</parameter>
            <parameter name="FUNCTION_MAX">9</parameter>
            <parameter name="FUNCTION_MIN">7</parameter>
            <parameter name="FUNCTION_SUM">11</parameter>
            <parameter name="FUNCTION_MEAN">13</parameter>
        </parameters>
    </sourceadapter>
</config>

```

Each function parameter is separate, so none is needed and any of the five function parameters in the example above can exist on its own.

## Required Software and Configuration

The IP21 Adapter will talk to a IP21 server through its ODBC interface. Thus a IP21 server needs to be in place.

### Source System Configuration

This must be configured in the file /<JBoss>/domain/servers/ec-server-a1/configuration/ecisconfig.xml

```
<config name="IP21_SOURCE_1">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.ip21.IP21JdbcAdapter</class>
            <sequential>Yes</sequential>
            <parameters>
                <parameter name="pwd">password</parameter>
                <parameter name="username">username</parameter>
                <parameter name="connecturl">connecturl, example:
jdbc:odbc:IP21SNA</parameter>
                <parameter
name="dbdriver">sun.jdbc.odbc.JdbcOdbcDriver</parameter>
            </parameters>
        </sourceadapter>
    </config>
```

In the section (config name) referring to the OPC adapter, the following must be configured

Parameter name	Value	Comments
class (mandatory)	com.ec.frmw.is.engine.adapter.ip21.IP21JdbcAdapter	Referring to the class of the adapter
username (mandatory)	<IP21 username>	The username needed to authenticate against the IP21 server.
pwd (mandatory)	<IP21 password>	The password needed to authenticate against the IP21 server.
Connecturl (mandatory)	<IP21 connecturl>	The url to connect against IP21's ODBC interface. Example: jdbc:odbc:SERVER
dbdriver (mandatory)	sun.jdbc.odbc.JdbcOdbcDriver	The driver used to connect to the ODBC interface of IP21.
check-valid-connection-sql	<SQL to check if the connection is valid>	If you want a check upon initialization to verify that the IP21 server is up and running, this is the parameter to use.
fieldid	<FIELD>	If you need to specify FIELD_ID in your queries, set this parameter to the field you want.

Request	<p>REQUEST can take the following values:</p> <p>1 = TIMES request (evenly spaced data).</p> <p>2 = TIMES2 request (evenly spaced data plus a data point for the end of the time range).</p> <p>3 = FITS request (The FITS request reduces the number of actual data points, while retaining the shape of the overall plot. The time period is divided into equally spaced intervals. The maximum, minimum, first, last, and first bad values in each interval are returned.)</p> <p>4 = VALUES request (actual recorded data).</p> <p>5 = ACTUALS AND CURRENT request (actual recorded data plus a data point for the current value).</p> <p>6 = TIMES_EXTENDED request (same as TIMES but can return data for time periods after the most recent data point. This only applies if STEPPED=1.)</p> <p>7 = TIMES2_EXTENDED request (same as TIMES2 but can return data for time periods after the most recent data point).</p> <p>STEPPEDInteger0 is the default and means that the data is treated as analog or continuous.</p>	Adds the REQUEST parameter to all queries against HISTORY table.
timestamp	<p>Values examples:</p> <ul style="list-style-type: none"> <li>• TS</li> <li>• TS_START</li> </ul>	Enables user to ask against different timestamps to get the timestamp for the sample. If not set, the default value is TS when querying against HISTORY and TS_START against AGGREGATES. All the values will be embraced by ISO8601(<timestamp>) to ensure the parameters meets ECIS's requirement to format.
timestampstart	<p>Values examples:</p> <ul style="list-style-type: none"> <li>• TS</li> <li>• TS_START</li> </ul>	Same as timestamp, however this attribute is only used against aggregated functions.
valuefunction	<p>Value example: F15.8</p>	If set, we will not ask for value, but for value using '<valuefunction>'. Enables the possibility to specify format of output.
checkstatussample	Yes/No or true/false	If this value is set to false or no, ECIS will not add AND STATUS = 0 to SAMPLE queries towards PI. If this value is not set, default is considered to be yes/true.
checkstatusaggregate	Yes/No or true/false	If this value is set to yes or true, ECIS will add AND STATUS = 0 to AGGREGATED queries towards PI. If this value is not set, default is considered to be no/false.

The parameter values must be changed to match the local infrastructure with machine names and user ids.

## Supported Source Functions

The supported source functions for the adapter are:

Function name	Description
SAMPLE	Reads raw samples from the IP21 Server. Can be slow when used on tags with high sampling frequency.
AVG	Time weighted average values
MAX	Maximum tag value within given period
MIN	Minimum tag value within given period.
SUM	Sum tag value within given period.
VALUE_AT_START	Values will be the last event before each period change.
VALUE_AT_END	Value will be the last event in each period.

## SQL Queries against IP21

Here is a list of the functions used against IP21. In addition the following things can be added:

- If request parameter is set, then for all queries against HISTORY table, AND REQUEST = <request> will be added to the end of the query.
- If timestamp parameter is set, then <timestamp> will be replaced by that value, if not TS will be the default value.
- If fielded parameter is set, then for all queries, AND FIELD\_ID = FT('<fielded>') will be added to the end of the query.

### **SAMPLE**

```
SELECT NAME, Value, ISO8601(<timestamp>) FROM HISTORY WHERE NAME = '<tagid>' AND status = 0 AND
TS >= '<fromDate>.000Z' AND TS < '<toDate>.000Z'
```

### **AVG**

```
SELECT NAME, AVG, ISO8601(<timestamp>) FROM aggregates WHERE NAME = '<tagid>' AND TS >=
'<fromDate>.000Z' AND TS_END < '<toDate>.000Z' AND period = <timeStep>
```

### **MIN**

```
SELECT NAME, MIN, ISO8601(<timestamp>) FROM aggregates WHERE NAME = '<tagid>' AND TS >=
'<fromDate>.000Z' AND TS_END < '<toDate>.000Z' AND period = <timeStep>
```

### **MAX**

```
SELECT NAME, MAX, ISO8601(<timestamp>) FROM aggregates WHERE NAME = '<tagid>' AND TS >=
'<fromDate>.000Z' AND TS_END < '<toDate>.000Z' AND period = <timeStep>
```

### **SUM**

```
SELECT NAME, SUM, ISO8601(<timestamp>) FROM aggregates WHERE NAME = '<tagid>' AND TS >=
'<fromDate>.000Z' AND TS_END < '<toDate>.000Z' AND period = <timeStep>
```

## Tag Based JDBC Adapter.

The tag based jdbc adapter simply selects values and quality based on a tag identifier and a daytime field from a table row in a database. In the simplest form of configuration the adapter will build the SQL, but there is also a possibility to provide the SQLs to the adapter with markers to where the adapter should insert tag Id and dates. Tag based jdbc adapter only supports source function SAMPLE and LATEST.

## Source System Configuration Oracle DBMS

This must be configured in the file /<JBoss>/domain/servers/ec-server-a1/configuration/ecisconfig.xml

```
<config name="TAGADAPTER">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.jdbc.SourceTagJdbcAdapter</class>
        <sequential>Yes</sequential>
        <parameters>
            <parameter
name="DRIVER">oracle.jdbc.driver.OracleDriver</parameter>
            <parameter
name="DB_URL">jdbc:oracle:thin:@host:1521:SID</parameter>
            <parameter name="DB_USER">user</parameter>
            <parameter name="DB_PWD">pwd</parameter>
            <parameter name="SQL_TAG_ID">IDENTIFIER</parameter>
            <parameter name="SQL_TABLE">db_name.SOMETABLE</parameter>
            <parameter name="SQL_VALUE">SOMEVALUE</parameter>
            <parameter name="SQL_DAYTIME">DAYTIME</parameter>
        </parameters>
    </sourceadapter>
</config>
```

In the section (config name) referring to the JDBC adapter, the following must be configured

Parameter name	Value	Comments
class (mandatory)	com.ec.frmw.is.engine.adapter.jdbc.SourceTagJdbcAdapter	Referring to the class of the adapter
DRIVER (mandatory)	Path to the jdbcdriver	Each dbms is providing a driver which is used by java to communicate with the dbms. If a non oracle driver is used. The library containing the driver should be available for ECIS classloader (add library to <JBoss-HOME>/server/ec/lib-EnergyX-ext/)
DB_URL (mandatory)	jdbc:oracle:thin:@<host>:1521:<SID>	Connection string to the database. Look in the vendors driver documentation for the format the selected driver.
DB_USER (mandatory)	Username	Id to user with access to read from the database
DB_PWD (mandatory)	Password	Password to user with access to read from the database
SQL_TAG_ID	String	Field in source table identifying the tag. Used to create the select statements.
SQL_TABLE	String	The table in the source database where the adapter should retrieve data from
SQL_VALUE	String	Field in source table holding the tag value. Used to create the select statements.
SQL_DAYTIME	String	Field in source table holding the timestamp for the tag value. Used to create the select statements. This field must be a date type.
SQL_QUALITY	String	Field in source table holding the quality value. This is not mandatory Used to create the select statements.

The parameter values must be changed to match the local infrastructure with machine names and user ids.

#### **Source System Configuration Other DBMS/ Non Standard Queries**

Since the jdbc adapter produces the select statement in a syntax tailored for oracle dbms, there is a possibility to configure the adapter to use predefined SQLs instead of the generated. In order for the adapter to inject the needed variables, the SQL should contain keywords for tagId, startDate and endDate. For aggregated functions, the keyword resolution is also available. The select must also provide the value as the first return column, the timestamp as the second and an optional quality column as the third. GET\_SAMPLES\_SQL and GET\_LATEST\_SQL is mandatory if you want to use the functionality this way. In addition it is possible to specify sql statements for the following parameters (Source functions in () ): GET\_AVG\_SQL (AVG), GET\_MEAN\_SQL (MEAN), GET\_MAX\_SQL (MAX), GET\_MIN\_SQL (MIN), GET\_SUM\_SQL (SUM). If you are using aggregated queries, you will need to use the parameter *maxsourceintervalinrequest*, and set it to 1. This way the adapter will ask for 1 source interval at a time.

```

<config name="TAGADAPTER">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.jdbc.SourceTagJdbcAdapter</class>
        <sequential>Yes</sequential>
        <parameters>
            <parameter
name="DRIVER">oracle.jdbc.driver.OracleDriver</parameter>
            <parameter
name="DB_URL">jdbc:oracle:thin:@host:1521:SID</parameter>
            <parameter name="DB_USER">user</parameter>
            <parameter name="DB_PWD">pwd</parameter>
            <parameter name="GET_SAMPLES_SQL">SELECT AVG_WH_TEMP ,
DAYTIME FROM ECKERNEL_VER_9_4_INC6_01.PWEL_SUB_DAY_STATUS WHERE
DATA_CLASS_NAME='$tagId$' AND DAYTIME>to_date('$startDate$', 'yyyy-MM-dd"T"HH24:Mi:SS') AND DAYTIME<to_date('$endDate$', 'yyyy-MM-dd"T"HH24:Mi:SS')</parameter>
            <parameter name="GET_LATEST_SQL">SELECT AVG_WH_TEMP ,
DAYTIME FROM ECKERNEL_VER_9_4_INC6_01.PWEL_SUB_DAY_STATUS WHERE
DATA_CLASS_NAME='$tagId$' AND DAYTIME=(SELECT MAX(DAYTIME) FROM
ECKERNEL_VER_9_4_INC6_01.PWEL_SUB_DAY_STATUS WHERE
DATA_CLASS_NAME='$tagId$' AND DAYTIME<to_date('$endDate$', 'yyyy-MM-dd"T"HH24:Mi:SS'))</parameter>
        </parameters>
    </sourceadapter>
</config>

```

## Tag File Adapter

### **Source System Configuration**

This must be configured in the file /<JBoss>/domain/servers/ec-server-a1/configuration/ecisconfig.xml

```
<config name="FILEADAPTER">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
        <sequential>Yes</sequential>
        <parameters>
            <parameter name="DropFolder">c:\\temp</parameter>
            <parameter
name="CompletedFolder">c:\\temp\\completed</parameter>
            <parameter name="ErrorFolder">c:\\temp\\error</parameter>
            <parameter name="BadFolder">c:\\temp\\bad</parameter>
            <parameter name="DateFormat">dd.MM.yyyy
HH:mm:ss</parameter>
            <parameter name="FileFilter"></parameter>
            <parameter name="QualityColumnIndex">0</parameter>
            <parameter name="TagColumnIndex">1</parameter>
            <parameter name="TimeColumnIndex">2</parameter>
            <parameter name="ValueColumnIndex">3</parameter>
        </parameters>
    </sourceadapter>
</config>
```

In the section (config name) refereeing to the FileAdapter, the following must be configured

Parameter name	Value	Comments
class (mandatory)	com.ec.frmw.is.engine.adapter.file.FileAdapter	Referring to the class of the adapter
sequential (optional)	Yes   True OR No   False	Indicates if the data coming from the adapter can be expected to be ordered by date / time
DropFolder (mandatory)	.	The folder where the adapter will look for files containing data.
ErrorFolder (mandatory)	/error/	The folder where the adapter will write rows with errors.
BadFolder (mandatory)	/bad/	The folder where the adapter will move bad files.
CompletedFolder (mandatory)	/completed/	The folder where the adapter will move files it has parsed.
DateFormat	yyyy-MM-dd'T'HH:mm:ss	The format the date column contains. Uses same notation as SimpleDateFormat in java.
FileFilter	Null per default, will parse all files in folder.	Filter to filter out the files the adapter should parse.
HeaderRow	N	Decides if there is a header row at the beginning of the file.
QualityColumnIndex	-1	Gives the index of the quality column. If this value is -1, then the data does not contain any quality column. Default is -1.
TagColumnIndex	0	Gives the index of the tag column.
TimeColumnIndex	1	Gives the index of the time column.
ValueColumnIndex	2	Gives the index of the value column.

The following parameters needs to be set to be able to read text-files:

Parameter name	Value	Comments
DecimalSeparator	.	Sets the separator used in decimal numbers.
FieldSeparator	;	Sets the separator used to separate columns.

The following parameters needs to be set to be able to read excel-files. If these are not set, the parameter assumes that the excel sheet starts at cell A1:

Parameter name	Value	Comments
StartCell	Null per default, will assume A1 as starting cell	The the cell where the data starts.
EndCell	Null per default	The last cell with data (Only EndCell or EndRow needs to be set).
EndRow	Null per default	The last row that contains data.

The parameter values must be changed to match the values needed to read your input files.

### **Supported source functions**

The following source functions are supported in the fileadapter:

Function name	Description
SAMPLE	Reads raw samples from the files.
AVG	Time weighted average for given period using linear interpolation.
MAX	Maximum tag value within given period
MIN	Minimum tag value within given period
MEAN	Plain average of all values within given period
SUM	Sum of all values within given period
VALUE_AT_START	Values will be the last event before each period change.
VALUE_AT_END	Value will be the last event in each period.

In the mapping configuration, these can be set up as the **Source Function** in the Maintain template screen. (CO.1017: *Maintain templates*).

### **Row File Adapter**

#### **Source System Configuration**

This must be configured in the file /<JBoss>/domain/servers/ec-server-a1/configuration/ecisconfig.xml. The example uses a package to write the data to EC. If you want to use another approach, please look at the target system configuration for rowadapters.

```

<config name="ROWADAPTER">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.file.RowFileAdapter</class>
            <parameters>
                <parameter
name="DropFolder">c:\\temp\\jboss\\rowadapter</parameter>
                    <parameter
name="CompletedFolder">c:\\temp\\jboss\\rowadapter\\completed</parameter>
                <parameter
name="ErrorFolder">c:\\temp\\jboss\\rowadapter\\error</parameter>
                    <parameter
name="BadFolder">c:\\temp\\jboss\\rowadapter\\bad</parameter>
                <parameter
name="DateFormat ">yyyy-MM-dd'T'HH:mm:ss</parameter>
                    <parameter name="FileFilter">rowdata</parameter>
                <parameter
name="DbPackageProcedure">UE_ECISROWTEST.ReceiveTestdat</parameter>
                <parameter
name="TargetDriver">oracle.jdbc.driver.OracleDriver</parameter>
                    <parameter name="TargetDbURL">
jdbc:oracle:thin:@<host>:1521:<SID></parameter>
                <parameter name="TargetDbUser">dbuser</parameter>
                <parameter name="TargetDbPassword">pwd</parameter>
                <parameter name="Col_1" type="list">
                    <subparameter name="columnIndex">0</subparameter>
                    <subparameter name="columnType">Date</subparameter>
                    <subparameter name="packageIndex">1</subparameter>
                </parameter>
                <parameter name="Col_2" type="list">
                    <subparameter name="columnIndex">1</subparameter>
                    <subparameter name="columnType">Double</subparameter>
                    <subparameter name="packageIndex">2</subparameter>
                </parameter>
                <parameter name="Col_3" type="list">
                    <subparameter name="columnIndex">2</subparameter>
                    <subparameter name="columnType">Double</subparameter>
                    <subparameter name="packageIndex">3</subparameter>
                </parameter>
                <parameter name="Col_4" type="list">
                    <subparameter name="columnIndex">4</subparameter>
                    <subparameter name="columnType">String</subparameter>
                    <subparameter name="packageIndex">4</subparameter>
                </parameter>
                <parameter
name="ErrorFile">c:\\temp\\jboss\\rowadapter\\errors\\errorfile.log</parameter>
            </parameters>
        </sourceadapter>
    </config>

```

In the section (config name) refereeing to the RowAdapter, the following must be configured

Parameter name	Value	Comments
DropFolder (mandatory)	.	The folder where the adapter will look for files containing data.
ErrorFolder (mandatory)	/error/	The folder where the adapter will write rows with errors.
BadFolder (mandatory)	/bad/	The folder where the adapter will move bad files.
CompletedFolder (mandatory)	/completed/	The folder where the adapter will move files it has parsed.
DateFormat	yyyy-MM-dd'T'HH:mm:ss	The format the date column contains. Uses same notation as SimpleDateFormat in java.
FileFilter	Null per default, will parse all files in folder.	Filter to filter out the files the adapter should parse.
HeaderRow	N	Decides if there is a header row at the beginning of the file.
ErrorFile	c:\\temp\\jboss\\rowadapter errorRows.log	A errorfile that will contain all the rows that threw an exception when the package tried to write them to DB.
AppendErrorCode	true/false	If this is set to true, a text saying "Errorcode: <errorcode>" will be appended to the end of each row of the file specified in the parameter ErrorFile.
SeparateErrorfiles	true/false	If this is set, errorfiles generated during reading of files will end up in the bad folder, while errorfiles generated during writing will end up in the error folder.

The following parameters needs to be set to be able to read text-files:

Parameter name	Value	Comments
DecimalSeparator	.	Sets the separator used in decimal numbers.
FieldSeparator	;	Sets the separator used to separate columns.

The following parameters needs to be set to be able to read excel-files. If these are not set, the parameter assumes that the excel sheet starts at cell A1:

Parameter name	Value	Comments
StartCell	Null per default, will assume A1 as starting cell	The the cell where the data starts.
EndCell	Null per default	The last cell with data (Only EndCell or EndRow needs to be set).
EndRow	Null per default	The last row that contains data.

In addition, you need to specify each column you want the RowAdapter to read, each row is specified as a parameter with name starting with *Col\_* and type set to list.

Parameter name	Value	Comments
Col_<number>	No value for this element, but needs to contain subparameters. Type of this element needs to be set to list.	Defines a new column
SubParameter name	Value	Comments
columnIndex	No default value	Describes which column in the file the adapter should map to this column.
columnType	No default value	The type of this column. Legal values are: <ul style="list-style-type: none"> <li>• String</li> <li>• Date</li> <li>• Double</li> </ul>
startPosition	No default value	If the file is a fixed format file, this field defines the startposition for this column.
endPosition	No default value	If the file is a fixed format file, this field defines the endposition for this column.

The parameter values must be changed to match the values needed to read your input files.

The adapter reads files from the drop folder without any checking if the file is ready first. Because of this, the files should be copied or moved into the folder after they are fully populated. This way you avoid the adapter reading files that are not yet ready.

## Row Based JDBC Adapter

### **Source System Configuration**

This must be configured in the file /<JBoss>/domain/servers/ec-server-a1/configuration/ecisconfig.xml. This example is writing directly to an EcClass using the classadapter. For other ways of handling rows on target side, see section about target system configuration for rowadapters.

```

<config name="JDBCADAPTER">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.jdbc.SourceRowJdbcAdapter</class>
        <parameters>
            <parameter
                name="DRIVER">oracle.jdbc.driver.OracleDriver</parameter>
            <parameter
                name="DB_URL">jdbc:oracle:thin:@<host>:1521:<SID></parameter>
                <parameter name="DB_USER">db_user</parameter>
                <parameter name="DB_PWD">db_password</parameter>
                <parameter name="SQL">select s.daytime,
s.avg_wh_press,s.avg_wh_temp, s.max_sand_rate, s.object_id,
s.object_code from dv_pwel_sub_day_status s where s.daytime &gt;
$last_fetch_oracle_format$</parameter>
                <parameter name="EcClass">PWEL_SAMPLE_1</parameter>
                <parameter name="Col_1" type="list">
                    <subparameter name=" EcClassAttribute
">DAYTIME</subparameter>
                </parameter>
                <parameter name="Col_2" type="list">
                    <subparameter name=" EcClassAttribute
">AVG_WH_PRESS</subparameter>
                </parameter>
                <parameter name="Col_3" type="list">
                    <subparameter name=" EcClassAttribute
">WH_TEMP</subparameter>
                </parameter>
                <parameter name="Col_4" type="list">
                    <subparameter name=" EcClassAttribute
">MAX SAND RATE</subparameter>
                </parameter>
                <parameter name="Col_5" type="list">
                    <subparameter name=" EcClassAttribute
">OBJECT_ID</subparameter>
                </parameter>
                <subparameter name=" EcAttributeFunction
">UE_ECISFUNCTION.mapObjectID</subparameter>
            </parameters>
            <parameter name="Col_6" type="list">
                <subparameter name=" EcClassAttribute
">OBJECT_CODE</subparameter>
            </parameters>
        </sourceadapter>
    </config>

```

In the section (config name) refereeing to the JDBCAdapter, the following must be configured

Parameter name	Value	Comments
DRIVER (mandatory)	Path to the jdbcdriver or JBoss_DATASOURCE	Each dbms is providing a driver which is used by java to communicate with the dbms. If a non oracle driver is used. The library containing the driver should be available for ECIS classloader(add library to .<JBoss-HOME>/server/ec/lib-EnergyX-ext/). It is also possible to use a JBoss datasource instead of defining the connection directly. This can be accomplished by setting JBOSS_DATASOURCE as the driver and the JNDI name in the DB_URL parameter
DB_URL (mandatory)	jdbc:oracle:thin:@<host>:1521:<SID> or JNDI name	Connection string to the database. Look in the vendors driver documentation for the format the selected driver. If JBOSS_DATASOURCE has been provided as driver, this parameter should specify the JNDI name of the datasource to use
DB_USER (mandatory)	Username	Id to user with access to read from the database
DB_PWD (mandatory)	Password	Password to user with access to read from the database
SQL	Select statement(s)	Statement which will retrieve data from the database. By inserting \$last_fetch\$ in the statement the adapter will inject a timestamp of when it last started the retrieval of data as a string with format: yyyy-mm-ddThh:MM:ss. You can use multiple selects in one payload by separating them with ;. Note: if this feature is used the consuming package must be able to handle the variance in the columns.
SQLFN	DB function that dynamically builds the select statement used to retrieve data.  Example value: XYZ.generateQuery('QUERY1')	Calls DB function to dynamically build the SQL statement used to retrieve data from the target DB. Use the SQLFN parameter if the DB function resides in the target database. Use ECSQLFN or ECSQLCLOBFN if the function resides in the EC database.  Note that SQLFN and ECSQLFN assume that the function returns the query as string/VARCHAR (i.e. limited length). ECSQLCLOBFN assume that the function returns a CLOB.  The returned query is subject to user event parameter substitution before being executed.
ECSQLFN ECSQLCLOBFN		



*A JDBC row adapter config must provide a value for exactly one of the SQL, SQLFN, ECSQLFN or ECSQLCLOBFN parameters.*

See next section Row File Adapter for the details on the target configuration for the details on the target configuration. It is also possible to define a JBoss datasource for the row based JDBC adapter. In this case the datasource file must be deployed to the /<JBoss>/server/ec/deploy/ec/ directory and the JNDI name referenced in the DB\_URL parameter. Please see the other datasource files in /<JBoss>/server/ec/deploy/ec/ for an example. Also, please note that if a JBoss datasource is used, the DB\_USER and DB\_PWD parameters are not required.

#### Updating the source after extraction

It is possible to use the adapter to perform an update on the rows which has been read. If the parameter with name UPDATE\_SQL is defined, the adapter will execute this for each row read sucessfully. the format of this parameter is normal sql syntax except that you can create markers for the adapter can insert columns for each row. A marker is of the format \$<col\_no> where col\_no is the index in the select starting from 1.

Example:

```
...
<parameter name="UPDATE_SQL ">update dv_pwel_sub_day_status set status='processed' where somekey=$1 and
someOtherKey=$2</parameter>
```

...  
Theese update statements is runned within the same transaction, so it is possible to have the selectstatement be a SELECT FOR UPDATE.

#### Running multiple sql statements to multiple target configurations

If several datasets are fetched from the same connection, you can configure a collection of SourceRowJdbcAdapters. The parameters are the same as described under section Source System Configuration and Supported source functions, but the sql, update and target parameters are subparameters to a new parameter where the name must contain the string "SRC\_TARGET".

```
<config name="JDBCADAPTER">
  <sourceadapter>
    <!-- NB: This has changed from previous class
    SourceJdbcAdapterCollection can execute several SRC_TARGET mappings
    while SourceRowJdbcAdapter can only execute one sql --->

    <class>com.ec.frmw.is.engine.adapter.jdbc.SourceJdbcAdapterCollection</class>
    <parameters>
      <parameter
        name="DRIVER">oracle.jdbc.driver.OracleDriver</parameter>
      <parameter
        name="DB_URL">jdbc:oracle:thin:@<host>:1521:<SID></parameter>
        <parameter name="DB_USER">db_user</parameter>
        <parameter name="DB_PWD">db_password</parameter>
        <!-- All sqls target mappings are ordered alphabetically
        and must contain string 'SRC_TARGET' -->
        <parameter name=" SRC_TARGET_SOME_IDENTIFYING_NAME_1 "
          type="list">
          <subparameter name="SQL">select pa.daytime,
            pa.avg_wh_press,pa.record_status from parentA pa where pa.lastupdate
            &gt; to_date('$last_fetch$', 'yyyy-MM-dd''T''HH24:Mi:SS')</subparameter>
          <subparameter
            name="EcClass">ECclassName</subparameter>
            <subparameter name="Col_1" type="list">
              <subparameter
                name="EcClassAttribute">daytime</subparameter>
                  <subparameter
                    name="EcAttributeFunction">package.function</subparameter>
```

```
</subparameter>
<subparameter name="Col_2" type="list">
    <subparameter
name="EcClassAttribute">avg_wh_press</subparameter>
    </subparameter>
    <subparameter name="Col_3" type="list">
        <subparameter
name="EcClassAttribute">record_status</subparameter>
        </subparameter>
    </parameter>
    <!-- All sqls target mappings are ordered alphabetically
and must contain string 'SRC_TARGET' -->
    <!-- depends attribute indicates that it should not
execute
        if the name inside depends has returned no values -->
    <parameter name=" SRC_TARGET_SOME_IDENTIFYING_NAME_2"
type="list" depends="SRC_TARGET_SOME_IDENTIFYING_NAME_1">
        <subparameter name="SQL">select ca.daytime,
ca.avg_wh_press,ca.record_status from childtA ca  where ca.lastupdate
&gt; to_date('$last_fetch$', 'yyyy-MM-dd"T"HH24:Mi:SS')</subparameter>
        <subparameter
name="EcClass">ECclassName2</subparameter>
        <subparameter name="Col_1" type="list">
            <subparameter
name="EcClassAttribute">daytime</subparameter>
            </subparameter>
            <subparameter name="Col_2" type="list">
                <subparameter
name="EcClassAttribute">avg_wh_press</subparameter>
                </subparameter>
                <subparameter name="Col_3" type="list">
                    <subparameter
name="EcClassAttribute">record_status</subparameter>
                    </subparameter>
                </parameters>
            </sourceadapter>
```

```
</config>
```

The configured SRC\_TARGET elements will share the database connection, but will create independent data units which it will send in sequence for writing to the defined targets. Like the other parameters in ecisconfig they are sorted alphabetically. However this will be overridden by the depends attribute. In the depends attribute it is possible to define one or more SRC\_TARGET elements which must return at least one row before the depending SRC\_TARGET is executed.

#### Fetching data from a backup solution

Bouth the SourceRowJdbcAdapter and the SourceJdbcAdapter collection have the possibility to retrieve data from a failover source. This is achieved by defining the connection parameters inside parameters which name contains "CONNECTION". Like the example below:

```
<!-- All connections are ordered alphabetically and must contain
string 'CONNECTION' -->
<parameter name="CONNECTION_PROPERTIES" type="list">
    <subparameter
        name="DRIVER">oracle.jdbc.driver.OracleDriver</subparameter>
    <subparameter
        name="DB_URL">jdbc:oracle:thin:@host:1521:SID</subparameter>
        <subparameter name="DB_USER">db_user</subparameter>
        <subparameter name="DB_PWD">db_password</subparameter>
        <!-- should return atleast 1 row if connection is active -->
        <subparameter name="ACTIVE_SQL">select * from dbstatus where
active_db=1</subparameter>
    </parameters>
    <!-- All connections are ordered alphabetically and must contain
string 'CONNECTION' -->
    <parameter name="CONNECTION_PROPERTIES_ALTERNATIVE" type="list">
        <subparameter
            name="DRIVER">oracle.jdbc.driver.OracleDriver</subparameter>
        <subparameter
            name="DB_URL">jdbc:oracle:thin:@host:1521:SID</subparameter>
            <subparameter name="DB_USER">db_user</subparameter>
            <subparameter name="DB_PWD">db_password</subparameter>
            <!-- should return atleast 1 row if connection is active -->
            <subparameter name="ACTIVE_SQL">select * from dbstatus where
active_db=1</subparameter>
    </parameters>
```

Like the other parameters in ecisconfig they are sorted alphabetically. If the first fails the adapter will try to execute the next. The subparameter ACTIVE\_SQL is not mandatory, but if it is present the SQL must return at least one row if the connection should be treated as operational.

#### Target system configuration for row adapters.

The target system is configured in the same config as the source system. Just add the parameters among all the others.

#### **Write to a package**

To write to a package, you need to specify which package and procedure you are going to write to. You can also specify if you want to write to another db than the one assosiated with your running jboss. This is done using these parameters:

Parameter name	Value	Comments
DbPackageProcedure	No default value	Gives the procedure in the PL/SQL package that will be run to insert the data into EC.
TargetDriver	No default value	Give the driver to the db you want to write to. If you don't specify any targetdriver and targetdburl, it will write to the db you are logged into jboss with. It is also possible to use a JBoss datasource instead of defining the connection directly. This can be accomplished by setting JBOSS_DATASOURCE as the driver and the JNDI name in the DB_URL parameter
TargetDbURL	No default value	Give the dbURL to the db you want to write to. If JBOSS_DATASOURCE has been provided as driver, this parameter should specify the JNDI name of the datasource to use
TargetDbUser	No default value	Username (not always needed)
TargetDbPassword	No default value	Password (not always needed)
ErrorFile	c:\\temp\\jboss\\rowadapter errorRows.log	A errorfile that will contain all the rows that threw an exception when the package tried to write them to DB.
AppendErrorCode	true/false	If this is set to true, a text saying "Errorcode: <errorcode>" will be appended to the end of each row of the file specified in the parameter ErrorFile.
SendUser	true/false	If this is set, the username of the user running the ECIS schedule will be sent as the last incoming value into the package.

Please note that if a JBoss datasource is used, the TargetDbUser and TargetDbPassword parameters are not required.

In addition, one need to specify which column in the source configuration goes to which inparameter in the procedure. This is done using the packageIndex subparameter. You don't make new Col\_ parameters, just put the subparameter packageIndex in the same as the sourceadapter.

Parameter name	Value	Comments
Col_<number>	No value for this element, but needs to contain subparameters. Type of this element needs to be set to list.	Defines a new column
SubParameter name	Value	Comments
packageIndex	No default value	Describes which position this column will have when putting it into the PL/SQL package. Only used with row file adapter.

### **Write directly to an EC class**

To write to a class, you need to specify which class you are going to write to, this is done using the EcClass parameter.

Be aware that this parameter is case sensitive.

Parameter name	Value	Comments
EcClass	No default value	The class you are going to write to. (Case sensitive)

In addition, you need to specify which field each column in the row is going to be written to. This is done using the subparameter EcClassAttribute. Be aware that this attribute is also case sensitive, and that you need to fulfill the primary key requirements for each class to be allowed to write to it. In addition, you might get a value from the source system which you want to map to something else before you can write it. This can be done specifying the parameter EdAttributeFunction. The value will then be put through this function before it is written to the db.

Parameter name	Value	Comments
Col_<number>	No value for this element, but needs to contain subparameters. Type of this element needs to be set to list.	Defines a new column
SubParameter name	Value	Comments
EcClassAttribute	No default value	The class attribute you want the column to be mapped to. (Case sensitive)
EcAttributeFunction	No default value	If you specify this parameter, the value will be run through this plsql function before it is written to database.

### **Write row data to a file**

It is possible to write row output to a file. To do that you can use the following parameters:

CSV_OUTFILE	No default value	Path to a file where the data is to be written.
CSV_DELIMITER (optional)	:	String separating the values in the csv file. If none given a single ; is used. If the string is larger than one character the first character will be appended before the first value and after the last value in each row.
OVERWRITE_OUTFILE (optional)	false	If the adapter should always write to the same file this value must be set to true. The default behavior is that for each run it will append a timestamp before the file ending. I.e If CSV_OUTFILE is set to C:\\temp\\csvouttest.txt the adapter will write the results to a file named C:\\temp\\csvouttest2008-07-09-06-43-37.txt

This functionality works combined with an other target system configuration. This means you can add these parameters in combination with writing through a package or directly to a EC Class. The system will then write the rows to a file before sending the DataTransferObjects to the jms queue. An example of a system using this functionality can look like this:

```

<config name="CSVOUT">
    <sourceadapter>

        <class>com.ec.frmw.is.engine.adapter.jdbc.SourceRowJdbcAdapter</class>
            <parameters>
                <parameter
name="DRIVER">oracle.jdbc.driver.OracleDriver</parameter>
                <parameter
name="DB_URL">jdbc:oracle:thin:@<host>:1521:<SID></parameter>
                <parameter name="DB_USER">user</parameter>
                <parameter name="DB_PWD"
encrypted="true">-66d6bf0f7ee54805</parameter>
                <parameter name="SQL">select s.daytime, s.avg_wh_press,
s.object_id, s.object_code from dv_pwel_sub_day_status s where
s.daytime > $last_fetch$</parameter>
                <parameter name="CSV_OUTFILE">C:\\temp
csvouttest.txt</parameter>
                <parameter name="CSV_DELIMITER">,"</parameter>
                <parameter name="OVERWRITE_OUTFILE">true</parameter>
            </parameters>
        </sourceadapter>
    </config>

```

## Appendix B - Example of ecisconfig.xml

Here are some examples of how configurations can look like. Remember that the values inside the parameters needs to be changed to fit your setup.

```

<configurations>
    <config name="TagFileAdapter">
        <sourceadapter>

            <class>com.ec.frmw.is.engine.adapter.file.TagFileAdapter</class>
                <sequential>Yes/NO</sequential>
                <parameters>
                    <parameter name="DropFolder">C:\\temp
dropfolder</parameter>
                    <parameter name="CompletedFolder">C:\\temp
completed</parameter>
                    <parameter name="ErrorFolder">C:\\temp
error</parameter>
                    <parameter name="BadFolder">C:\\temp
bad</parameter>
                    <parameter name="FileFilter">filefilter</parameter>
                    <parameter
name="DateFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
                    <parameter
name="DateValueFormat">yyyy-MM-dd'T'HH:mm:ss</parameter>
                    <parameter name="HeaderRow">N</parameter>
                    <parameter name="QualityColumnIndex">-1</parameter>
                    <parameter name="TagColumnIndex">4</parameter>
                    <parameter name="TimeColumnIndex">0</parameter>
                    <parameter name="ValueColumnIndex">2</parameter>

```

```

<parameter name="DecimalSeparator">.</parameter>
<parameter name="FieldSeparator">;</parameter>
<parameter name="StartCell">A1</parameter>
<parameter name="EndCell"></parameter>
<parameter name="EndRow"></parameter>
</parameters>
</sourceadapter>
</config>
<config name="RowFileAdapter to package">
    <sourceadapter>

<class>com.ec.frmw.is.engine.adapter.file.RowFileAdapter</class>
    <parameters>
        <parameter
name="DropFolder">C:\\temp\\\\dropfolder</parameter>
        <parameter
name="CompletedFolder">C:\\temp\\\\completed</parameter>
        <parameter
name="ErrorFolder">C:\\temp\\\\error</parameter>
        <parameter name="BadFolder">C:\\temp\\\\bad</parameter>
        <parameter name="FileFilter">filefilter</parameter>
        <parameter
name="DateFormat ">yyyy-MM-dd'T'HH:mm:ss</parameter>
        <parameter name="HeaderRow">N</parameter>
        <parameter name="DecimalSeparator">.</parameter>
        <parameter name="FieldSeparator">;</parameter>
        <parameter name="StartCell">A1</parameter>
        <parameter name="EndCell"></parameter>
        <parameter name="EndRow"></parameter>
        <parameter
name="DbPackageProcedure">UE_ECISROWTEST.ConsTestdata</parameter>
        <parameter name="Col_1" type="list">
            <subparameter name="columnIndex">0</subparameter>
            <subparameter
name="columnType">Date/Double/String</subparameter>
            <subparameter name="packageIndex">1</subparameter>
            <subparameter name="startPosition"></subparameter>
            <subparameter name="endPosition"></subparameter>
        </parameter>
        </parameters>
    </sourceadapter>
</config>
<config name="RowFileAdapter to class">
    <sourceadapter>

<class>com.ec.frmw.is.engine.adapter.file.RowFileAdapter</class>
    <parameters>
        <parameter
name="DropFolder">C:\\temp\\\\dropfolder</parameter>
        <parameter
name="CompletedFolder">C:\\temp\\\\completed</parameter>
        <parameter
name="ErrorFolder">C:\\temp\\\\error</parameter>
        <parameter name="BadFolder">C:\\temp\\\\bad</parameter>
        <parameter name="FileFilter">filefilter</parameter>
        <parameter
name="DateFormat ">yyyy-MM-dd'T'HH:mm:ss</parameter>

```

```

<parameter name="HeaderRow">N</parameter>
<parameter name="DecimalSeparator">.</parameter>
<parameter name="FieldSeparator">;</parameter>
<parameter name="StartCell">A1</parameter>
<parameter name="EndCell"></parameter>
<parameter name="EndRow"></parameter>
<parameter name="EcClass">PWEL_SAMPLE_1</parameter>
<parameter name="Col_1" type="list">
    <subparameter name="columnIndex">0</subparameter>
    <subparameter
        name="columnType">Date/Double/String</subparameter>
        <subparameter name="startPosition"></subparameter>
        <subparameter name="endPosition"></subparameter>
        <subparameter
            name="EcClassAttribute">DAYTIME</subparameter>
            <subparameter
                name="EcAttributeFunction">UE_ECISTESTPACKAGE.MAPVALUE</subparameter>
                </parameter>
            </parameters>
        </sourceadapter>
    </config>
    <config name="JDBCAdapter to class">
        <sourceadapter>

<class>com.ec.frmw.is.engine.adapter.jdbc.SourceRowJdbcAdapter</class>
        <parameters>
            <parameter
                name="DRIVER">oracle.jdbc.driver.OracleDriver</parameter>
                <parameter
                    name="DB_URL">jdbc:oracle:thin:@localhost:1521:ECPM</parameter>
                    <parameter name="DB_USER">user</parameter>
                    <parameter name="DB_PWD">password</parameter>
                    <parameter
                        name="ErrorFile">C:\\\\temp\\\\errorFile.log</parameter>
                        <parameter name="SQL">select s.daytime,
s.avg_wh_press,s.avg_wh_temp, s.max_sand_rate, s.object_id,
s.object_code from dv_pwel_sub_day_status s where s.daytime >
$last_fetch_oracle_format$ for update</parameter>
                        <parameter name="UPDATE_SQL">update
dv_pwel_sub_day_status s set s.avg_wh_press = 14 where s.daytime = $1
and s.object_id = $5 and s.object_code = $6</parameter>
                        <parameter name="EcClass">PWEL_SAMPLE_1</parameter>
                        <parameter name="Col_1" type="list">
                            <subparameter
                                name="EcClassAttribute">DAYTIME</subparameter>
                                <subparameter
                                    name="EcAttributeFunction">UE_ECISTESTPACKAGE.MAPVALUE</subparameter>
                                    </parameter>
                                </parameters>
                            </sourceadapter>
                        </config>
                        <config name="JDBCAdapter to package on different DB">
                            <sourceadapter>

<class>com.ec.frmw.is.engine.adapter.jdbc.SourceRowJdbcAdapter</class>
                            <parameters>
                                <parameter

```

```

name="DRIVER">>oracle.jdbc.driver.OracleDriver</parameter>
    <parameter
name="DB_URL">>jdbc:oracle:thin:@localhost:1521:ECPM</parameter>
    <parameter name="DB_USER">>user</parameter>
    <parameter name="DB_PWD">>password</parameter>
    <parameter
name="TargetDriver">>oracle.jdbc.driver.OracleDriver</parameter>
    <parameter
name="TargetDbURL">>jdbc:oracle:thin:@localhost:1521:ECPM</parameter>
    <parameter
name="TargetDbUser">>targetDbUser</parameter>
    <parameter
name="TargetDbPassword">>password</parameter>
    <parameter
name="ErrorFile">>C:\\temp\\\\errorFile.log</parameter>
    <parameter name="SQL">>select s.daytime,
s.avg_wh_press,s.avg_wh_temp, s.max_sand_rate, s.object_id,
s.object_code from dv_pwel_sub_day_status s where s.daytime &gt;
$last_fetch_oracle_format$ for update</parameter>
    <parameter name="UPDATE_SQL">>update
dv_pwel_sub_day_status s set s.avg_wh_press = 14 where s.daytime = $1
and s.object_id = $5 and s.object_code = $6</parameter>
    <parameter
name="DbPackageProcedure">>UE_ECISROWTEST.ConsTestdata</parameter>
    </parameters>
    </sourceadapter>
</config>
<config name="JDBCAdapter collection to class">
    <sourceadapter>

<class>com.ec.frmw.is.engine.adapter.jdbc.SourceJdbcAdapterCollection<
/class>
    <parameters>
        <parameter
name="DRIVER">>oracle.jdbc.driver.OracleDriver</parameter>
        <parameter
name="DB_URL">>jdbc:oracle:thin:@localhost:1521:ECPM</parameter>
        <parameter name="DB_USER">>user</parameter>
        <parameter name="DB_PWD">>password</parameter>
        <parameter
name="ErrorFile">>C:\\temp\\\\errorFile.log</parameter>
        <parameter name="SRC_TARGET_CONFIG_1" type="list">
            <subparameter name="SQL">>select s.daytime,
s.avg_wh_press,s.avg_wh_temp, s.max_sand_rate, s.object_id,
s.object_code from dv_pwel_sub_day_status s where s.daytime &gt;
$last_fetch_oracle_format$</subparameter>
            <subparameter
name="EcClass">>PWEL_SAMPLE_2</subparameter>
            <subparameter name="Col_01" type="list">
                <subparameter
name="EcClassAttribute">>DAYTIME</subparameter>
                </subparameter>
                <subparameter name="Col_02" type="list">
                    <subparameter
name="EcClassAttribute">>WH_PRESS</subparameter>
                    </subparameter>
                    <subparameter name="Col_03" type="list">

```

```

        <subparameter
name="EcClassAttribute">WH_TEMP</subparameter>
            </subparameter>
            <subparameter name="Col_04" type="list">
                <subparameter
name="EcClassAttribute">MAX_SAND_RATE</subparameter>
                    </subparameter>
                    <subparameter name="Col_05" type="list">
                        <subparameter
name="EcClassAttribute">OBJECT_ID</subparameter>
                            <subparameter
name="EcAttributeFunction">UE_ECISCONVERTVALUE.StreamCode2Object_Id</su
bparameter>
                                </subparameter>
                                <subparameter name="Col_06" type="list">
                                    <subparameter
name="EcClassAttribute">OBJECT_CODE</subparameter>
                                        </subparameter>
                                    </parameter>
                                    <parameter name="SRC_TARGET_CONFIG_2" type="list">
                                        <subparameter name="SQL">select s.daytime,
s.avg_wh_press, s.object_id, s.object_code from dv_pwel_sample_1 s
where s.daytime > $last_fetch_oracle_format$</subparameter>
                                    <subparameter
name="EcClass">PWEL_SAMPLE_3</subparameters>
                                        <subparameter name="Col_01" type="list">
                                            <subparameter
name="EcClassAttribute">DAYTIME</subparameter>
                                                <subparameter
name="EcAttributeFunction">UE_ECISCONVERTVALUE.FindCorrectDaytime</su
bparameter>
                                                    </subparameter>
                                                    <subparameter name="Col_02" type="list">
                                                        <subparameter
name="EcClassAttribute">WH_PRESS</subparameter>
                                                            </subparameter>
                                                            <subparameter name="Col_03" type="list">
                                                                <subparameter
name="EcClassAttribute">OBJECT_ID</subparameter>
                                                                    </subparameter>
                                                                    <subparameter name="Col_04" type="list">
                                                                        <subparameter
name="EcClassAttribute">OBJECT_CODE</subparameter>
                                                                            </subparameter>
                                                                            </parameter>
                                                                            </parameters>
                                                                        </sourceadapter>
                                                                    </config>
                                                                <config name="JDBCAdapter collection to package">
                                                                    <sourceadapter>

<class>com.ec.frmw.is.engine.adapter.jdbc.SourceJdbcAdapterCollection<
/class>
                <parameters>
                    <parameter
name="DRIVER">oracle.jdbc.driver.OracleDriver</parameter>
                    <parameter

```

```

name="DB_URL">>jdbc:oracle:thin:@localhost:1521:ECPM</parameter>
    <parameter name="DB_USER">user</parameter>
    <parameter name="DB_PWD">password</parameter>
    <parameter
name="ErrorFile">>C:\\\\temp\\\\errorFile.log</parameter>
    <parameter name="SRC_TARGET_CONFIG_1" type="list">
        <subparameter name="SQL">select s.daytime,
s.avg_wh_press,s.avg_wh_temp, s.max_sand_rate, s.object_id,
s.object_code from dv_pwel_sub_day_status s where s.daytime &gt;
$last_fetch_oracle_format$</subparameter>
        <subparameter
name="DbPackageProcedure">>UE_ECISROWTEST.ConsTestdata</subparameter>
    </parameter>
    <parameter name="SRC_TARGET_CONFIG_2" type="list">
        <subparameter name="SQL">select s.daytime,
s.avg_wh_press, s.object_id, s.object_code from dv_pwel_sample_1 s
where s.daytime &gt; $last_fetch_oracle_format$</subparameter>
        <subparameter
name="DbPackageProcedure">>UE_ECISROWTEST.ConsTestdata2</subparameter>
    </parameter>
    </parameters>
</sourceadapter>
</config>
<config name="OPCAdapter">
    <sourceadapter>

<class>com.ec.frmw.is.engine.adapter.opc.OPCAdapter</class>
    <sequential>Yes/NO</sequential>
    <parameters>
        <parameter name="AUTH_DOMAIN">MyDomain</parameter>
        <parameter name="USR">opcreader</parameter>
        <parameter name="PWD">password</parameter>
        <parameter name="OPC_HOST">127.0.0.1</parameter>
        <parameter
name="PROG_ID">Some.OPC.Server.1</parameter>
        <parameter name="IID"></parameter>
        <parameter name="USE_PROG_ID">true</parameter>
        <parameter
name="ADD_SHUTDOWN_CALLBACK">>ADD_SHUTDOWN_CALLBACK</parameter>
    </parameters>
    </sourceadapter>
</config>
<config name="PIAdapter MSSQL Server">
    <sourceadapter>

<class>com.ec.frmw.is.engine.adapter.pi.jdbc.PiJdbcAdapter</class>
    <sequential>Yes/NO</sequential>
    <parameters>
        <parameter
name="dbdriver">>net.sourceforge.jtds.jdbc.Driver</parameter>
        <parameter
name="connecturl">>jdbc:jtds:sqlserver://localhost:1433</parameter>
        <parameter name="username">username</parameter>
        <parameter name="pwd">password</parameter>
        <parameter
name="linkedserver">>linkedserver</parameter>
    </parameters>

```

```
</sourceadapter>
</config>
<config name="PIAdapter jinterop">
  <sourceadapter>

<class>com.ec.frmw.is.engine.adapter.pi.jinterop.PiAdapter</class>
  <sequential>Yes/NO</sequential>
  <parameters>
    <parameter name="authdomain">MyDomain</parameter>
    <parameter name="domainuser">domainuser</parameter>
    <parameter name="domainpwd">password</parameter>
    <parameter name="dsn">127.0.0.1</parameter>
    <parameter name="username">username</parameter>
    <parameter name="pwd"></parameter>
    <parameter name="timeout">180</parameter>
    <parameter
name="logfile">c:\\temp\\logfile.log</parameter>
  </parameters>
```

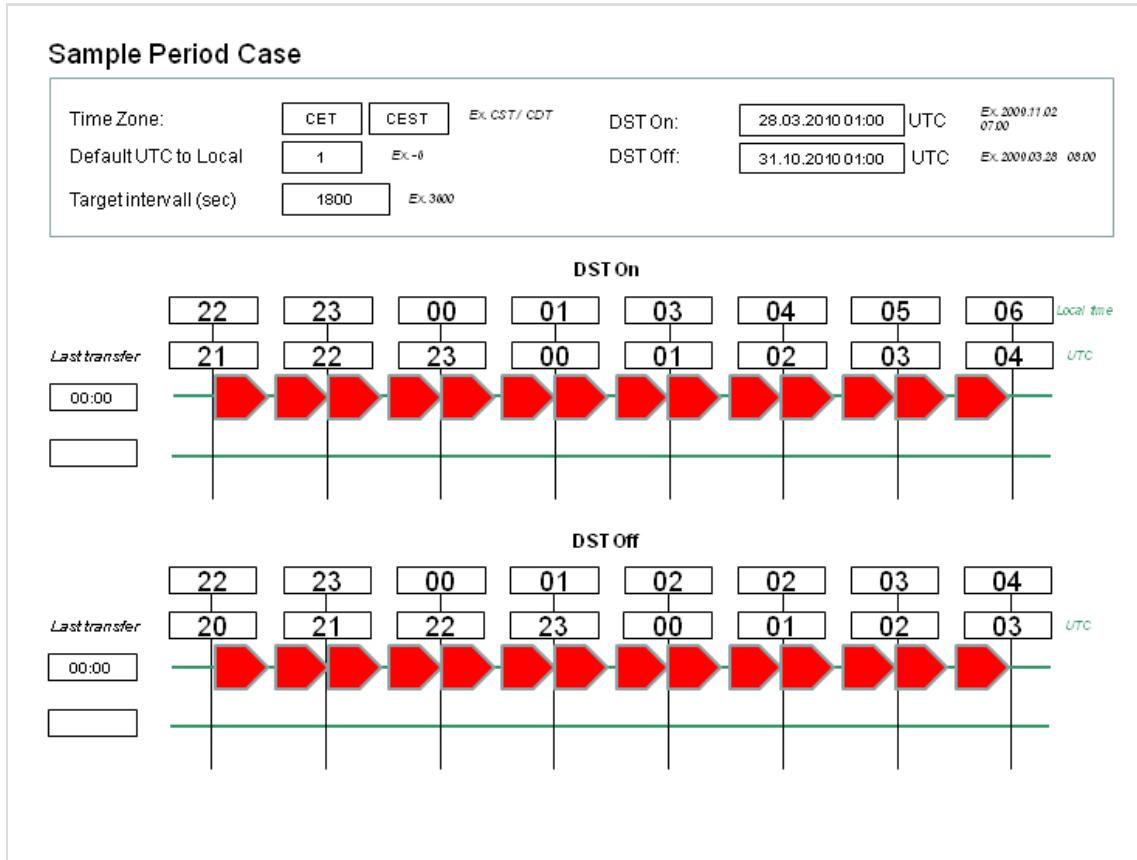
```

    </sourceadapter>
  </config>
</configurations>

```

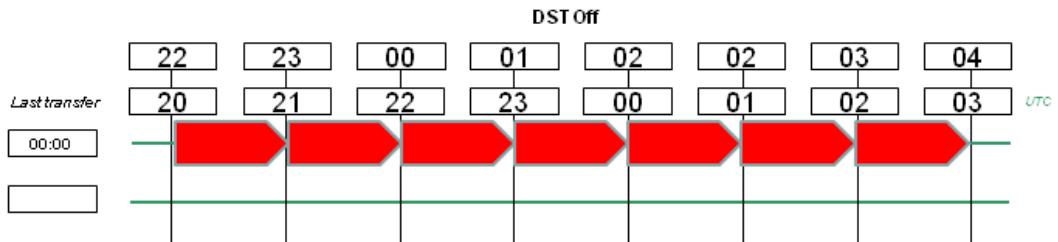
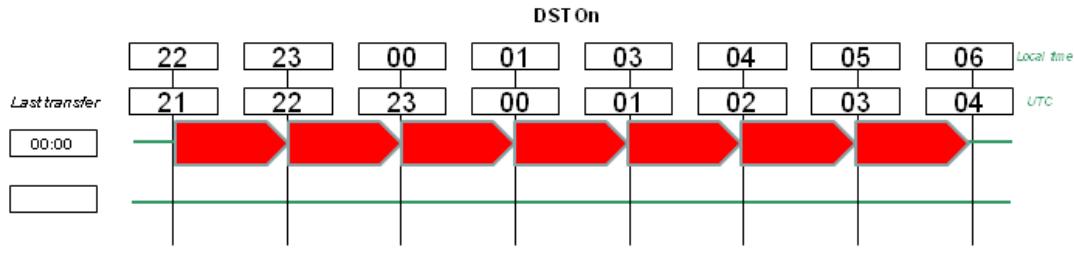
## Appendix C – Examples of sample period handling during DST

In this section there will be a number of cases with different targetintervall and timezones to show how ECIS will create the sampleperiods during aggregation.



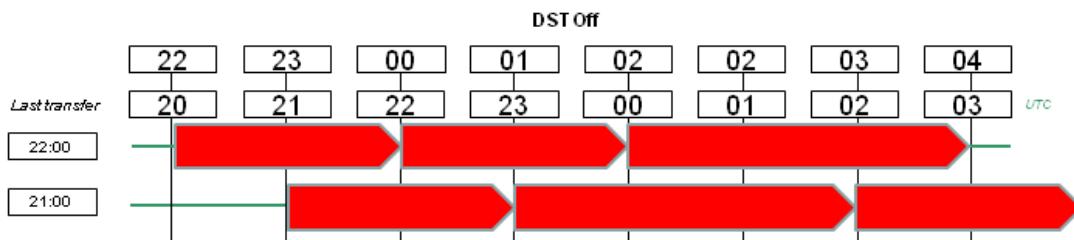
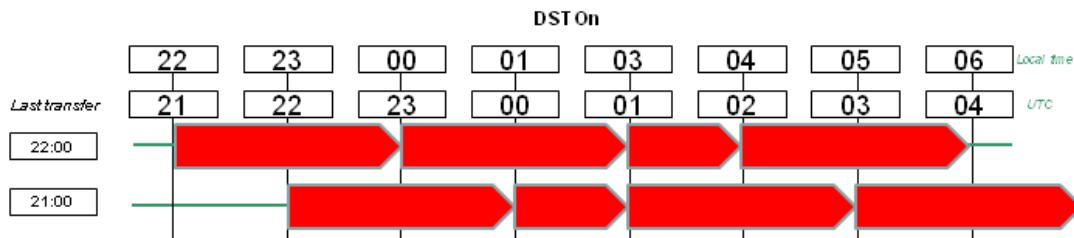
### Sample Period Case

Time Zone:	CET	CEST	Ex. CST/ CDT	DST On:	28.03.2010 01:00 UTC	Ex. 2009.11.02 07:00
Default UTC to Local	1	Ex. -6		DST Off:	31.10.2010 01:00 UTC	Ex. 2009.03.28 08:00
Target interval (sec)	3600	Ex. 3600				



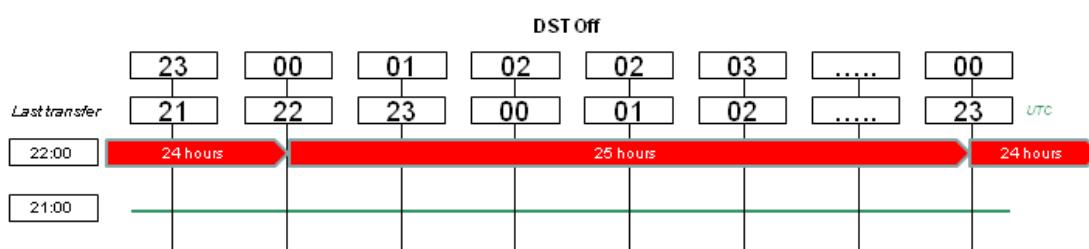
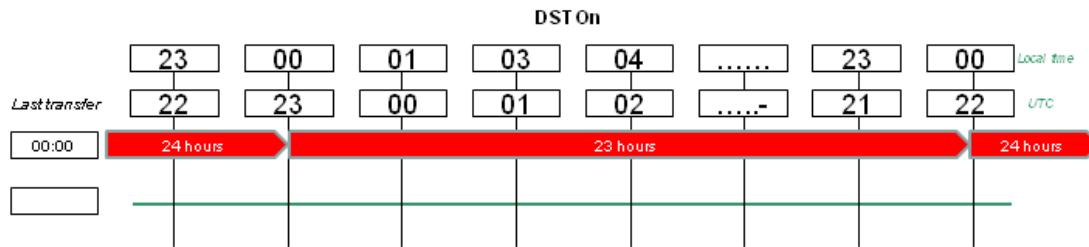
### Sample Period Case

Time Zone:	CET	CEST	Ex. CST/ CDT	DST On:	28.03.2010 01:00 UTC	Ex. 2009.11.02 07:00
Default UTC to Local	1	Ex. -6		DST Off:	31.10.2010 01:00 UTC	Ex. 2009.03.28 08:00
Target interval (sec)	7200	Ex. 3600				



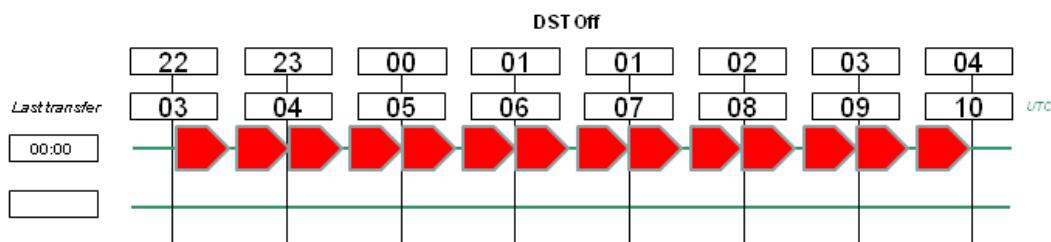
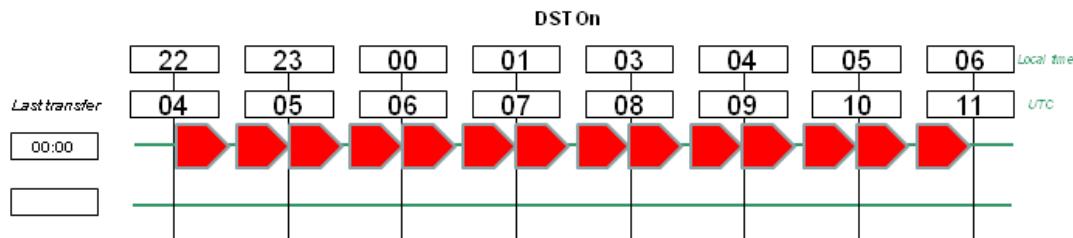
### Sample Period Case

Time Zone:	CET	CEST	Ex. CST/ CDT	DST On:	28.03.2010 01:00 UTC	Ex. 2009/11/02 07:00
Default UTC to Local	1	-6	Ex. -6	DST Off:	31.10.2010 01:00 UTC	Ex. 2009/03/28 08:00
Target interval (sec)	86400	Ex. 3600				



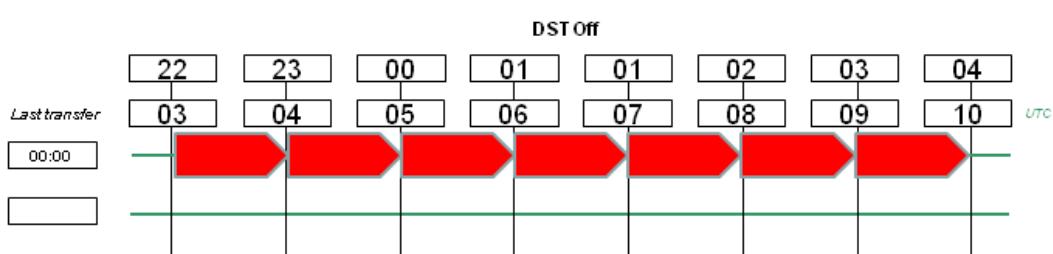
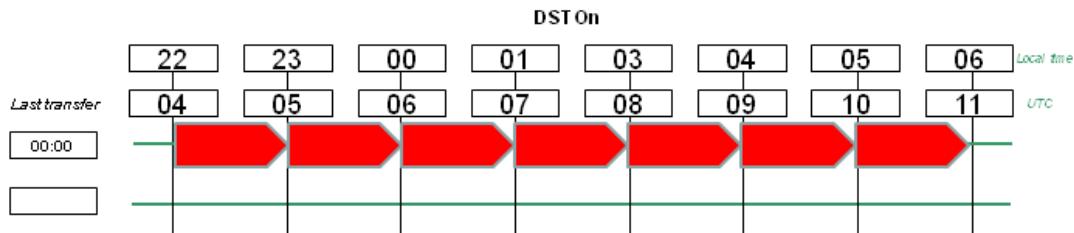
### Sample Period Case

Time Zone:	CST	CDT	Ex. CST/ CDT	DST On:	14.03.2010 08:00 UTC	Ex. 2009/11/02 07:00
Default UTC to Local	-6	-6	Ex. -6	DST Off:	20.10.2010 07:00 UTC	Ex. 2009/03/28 08:00
Target interval (sec)	1800	Ex. 3600				

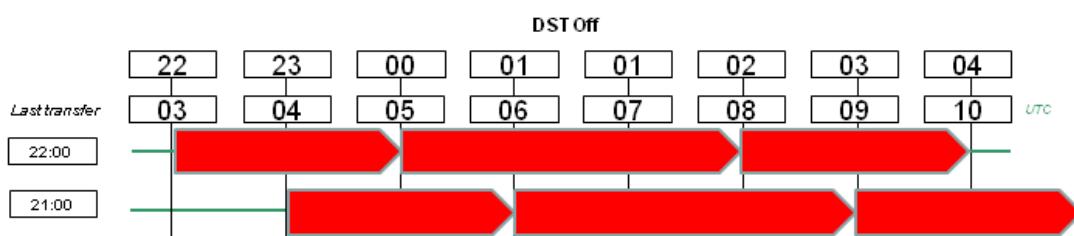
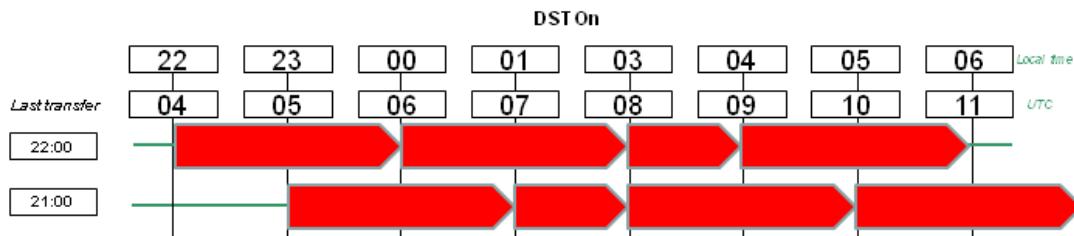


**Sample Period Case**

Time Zone:	CST	CDT	Ex: CST / CDT	DST On:	14.03.2010 08:00	UTC	Ex: 2009.11.02 07:00
Default UTC to Local	-6		Ex: -6	DST Off:	20.10.2010 07:00	UTC	Ex: 2009.03.28 08:00
Target interval (sec)	3600		Ex: 3600				

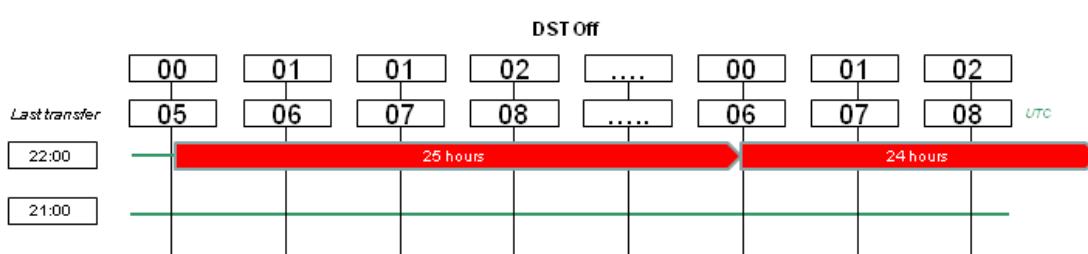
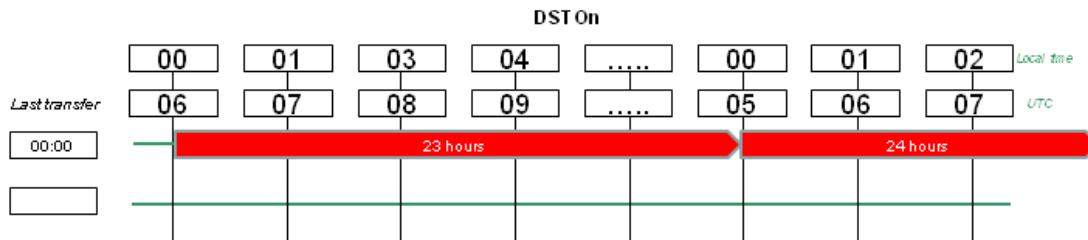
**Sample Period Case**

Time Zone:	CST	CDT	Ex: CST / CDT	DST On:	14.03.2010 08:00	UTC	Ex: 2009.11.02 07:00
Default UTC to Local	-6		Ex: -6	DST Off:	20.10.2010 07:00	UTC	Ex: 2009.03.28 08:00
Target interval (sec)	7200		Ex: 3600				

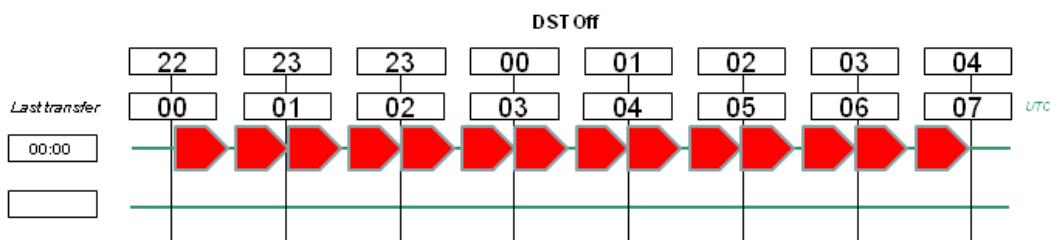
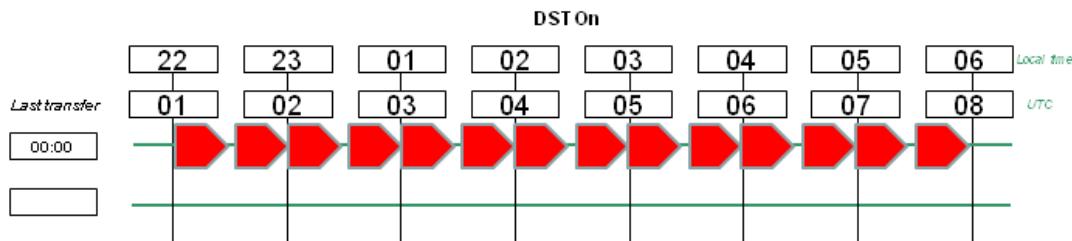


**Sample Period Case**

Time Zone:	CST	CDT	Ex: CST / CDT	DST On:	14.03.2010 08:00 UTC	Ex: 2009.11.02 07:00
Default UTC to Local	-6		Ex: -6	DST Off:	20.10.2010 07:00 UTC	Ex: 2009.03.28 08:00
Target interval (sec)	86400		Ex: 3600			

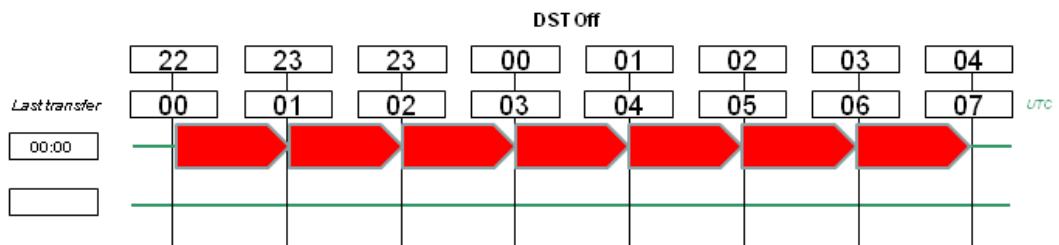
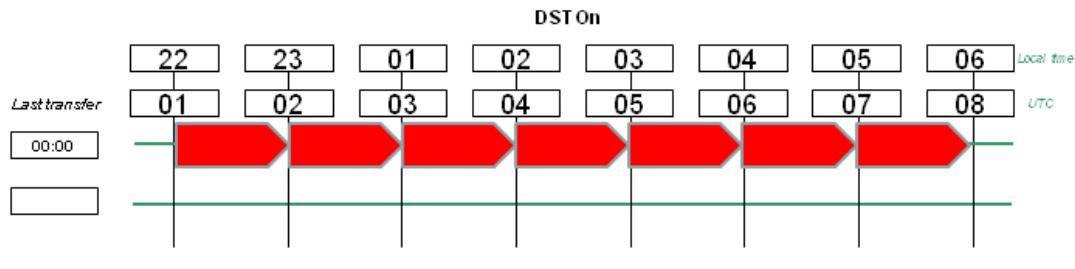
**Sample Period Case**

Time Zone:	BRT	BRST	Ex: CST / CDT	DST On:	17.10.2010 03:00 UTC	Ex: 2009.11.02 07:00
Default UTC to Local	-3		Ex: -6	DST Off:	21.02.2010 02:00 UTC	Ex: 2009.03.28 08:00
Target interval (sec)	1800		Ex: 3600			

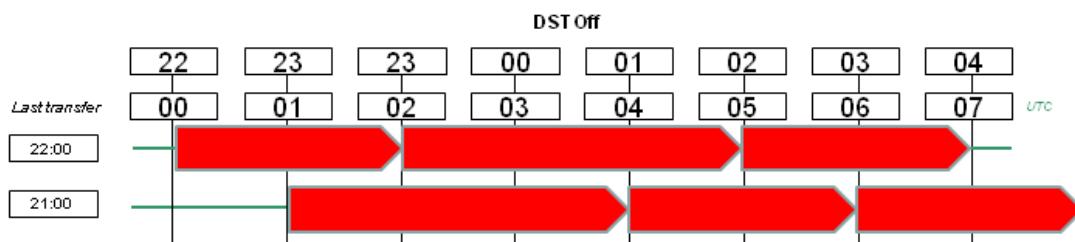
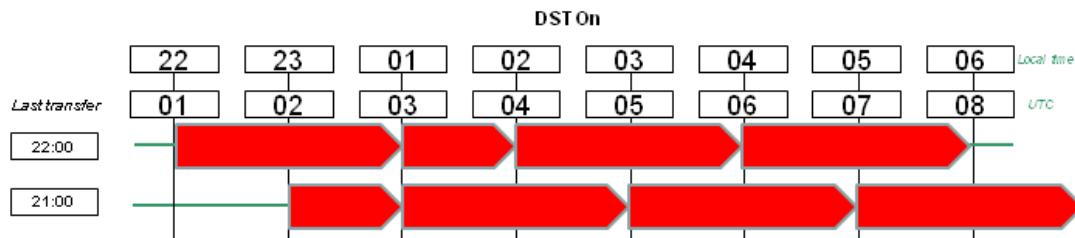


**Sample Period Case**

Time Zone:	BRT	BRST	Ex. CST/ CDT	DST On:	17.10.2010 03:00 UTC	Ex. 2009.11.02 07:00
Default UTC to Local	-3		Ex. -6	DST Off:	21.02.2010 02:00 UTC	Ex. 2009.03.28 08:00
Target interval (sec)	3600		Ex. 3600			

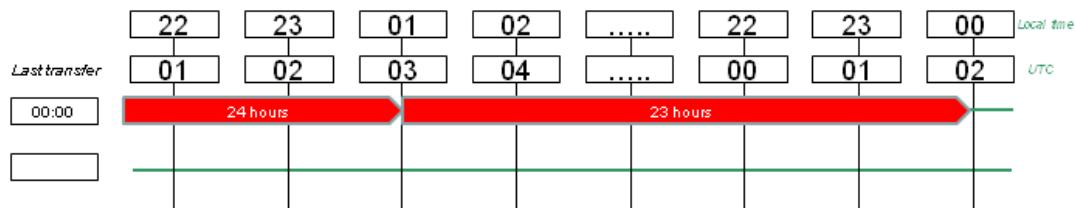
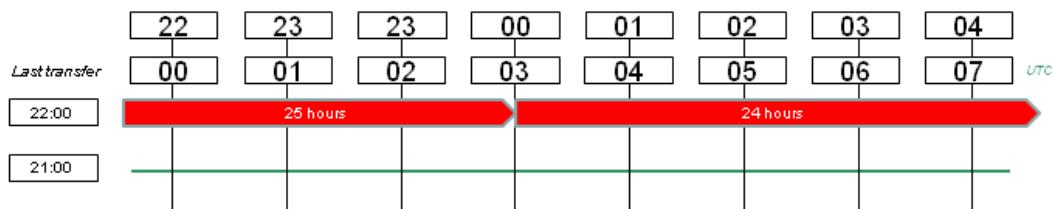
**Sample Period Case**

Time Zone:	BRT	BRST	Ex. CST/ CDT	DST On:	17.10.2010 03:00 UTC	Ex. 2009.11.02 07:00
Default UTC to Local	-3		Ex. -6	DST Off:	21.02.2010 02:00 UTC	Ex. 2009.03.28 08:00
Target interval (sec)	7200		Ex. 3600			



**Sample Period Case**

Time Zone:	BRT	BRST	Ex. CST / CDT	DST On:	17.10.2010 03:00	UTC	Ex. 2009.11.02 07:00
Default UTC to Local	-3		Ex. -6	DST Off:	21.02.2010 02:00	UTC	Ex. 2009.03.28 08:00
Target interval (sec)	86400		Ex. 3600				

**DST On****DST Off**

new

# EC Web Services

## Introduction

This section will give a description of how a web service based on a class in EC will look like. It will then give a guide about how to generate these web services, and how to build and deploy them.

## Specification of a web service

The user specifies which class they want to generate web service from, through a flag called INCLUDE\_WEBSERVICE in the class CLASS. For each of these web services there will be generated a set of methods. Methods that return data, will return a Plain old java object (POJO) of the type of the given class. Here we will give a description of how that POJO will look like, and then a description of all the methods that will be available for the web service.

### POJO

Each class will have its own POJO containing all the attributes for the class, including get and set methods for these attributes. If you want a list of all the attributes for a class, you can fetch them from the following query against the database:

```
SELECT PROPERTY_NAME, DATA_TYPE FROM DAO_META
WHERE CLASS_NAME = '<class_name>'
AND PROPERTY_NAME NOT LIKE '%_POPUP'
```

Each attribute will have a datatype according to the following map:

Type in database	Type in java
• DATE	• Date
• NUMBER	• BigDecimal
• STRING	• String
• BOOLEAN	• Boolean
• INTEGER	• Integer

## Methods in a web service

### Find object

- findByPK(<Primary key values>)

Method will run a query against the DAO requesting the object with the given primary key values. The primary key values will be the same values defined as key values on the class. If it is an object-class, the key is always objectId and daytime. The method will return a single POJO of the type of the web service class, containing the appropriate values. If no rows are found, the method will return nothing.

### Find object using object code

- findByPKCode(<Primary key values>)

Method will run a query against the DAO requesting the object with the given primary key values. The primary key values will be the same values defined as key values on the class. If it is an object-class, the keys are always objectCode and daytime. This method will internally translate the objectCode into corresponding objectId. The method will return a single POJO of the type of the web service class, containing the appropriate values. If no rows are found, the method will return nothing.

### Find objects for a time range

- `findByPkTimeRange(<pk-VALUES>, FROMTIME, TOTIME)`

This method will be generated for all classes not having NONE as DATE\_HANDLING attribute. It will take the primary key values except daytime as in-parameters, and also a from-time and to-time. It will return one POJO per row that match the primary key values, and has a daytime between fromtime and totime. Method is from and including from-time, to not including to-time.

#### ***Find objects for a time range using object code***

- `findByPkCodeTimeRange(<pk-VALUES>, FROMTIME, TOTIME)`

This method will be generated for all classes not having NONE as DATE\_HANDLING attribute. It will take the primary key values except daytime as in parameters, and also a from-time and to-time. This method will internally translate the objectCode into corresponding objectId. It will return one POJO per row that match the primary key values, and has a daytime between from-time and to-time. Method is from and including from-time, to not including to-time.

#### ***Find latest value***

- `findLatestByPk(<pk-values>, beforetime)`

This method will be generated for all classes not having NONE as DATE\_HANDLING attribute. It will take the primary key values except daytime as in-parameters, and also a before-time. It will return one POJO containing the latest value with the given primary key values and a daytime before the beforetime.

#### ***Find latest value using object code***

- `findLatestByPkCode(<pk-values>, beforetime)`

This method will be generated for all classes not having NONE as DATE\_HANDLING attribute. It will take the primary key values except daytime as in-parameters, and also a before-time. This method will internally translate the objectCode into corresponding objectId. It will return one POJO containing the latest value with the given primary key values and a daytime before the before-time.

#### ***Update or insert rows***

- `update(<Array of POJO's>)`

Method takes an array of POJO's. It will loop through this list, and if the row represented by the POJO is already present in EC, it will run an update against it. If the row represented by the POJO is not present in EC, the method will insert it into EC.

#### ***Delete rows***

- `delete(<Array of POJO's>)`

Method takes an array of POJO's, and will delete all rows present in EC with the given values. It is not possible to delete rows in a class of type OBJECT.

#### ***Retrieve class metadata***

- `getMetaData()`

Returns an object of type ClassMetaData containing all the meta information about the given class.

#### ***Get related class values***

- `get<RoleName>(<pk-values>)`

Many classes in EC have relations to other classes. If you run the following query, you will get all roles and attributes defining them:

```
select PROPERTY_NAME, REL_CLASS_NAME, ROLE_NAME
from DAO_META where class_name = '<class_name>'
AND ROLE_NAME IS NOT NULL;
```

There will be generated one method per ROLE\_NAME available for the class. The method will be called

get<ROLE\_NAME>. The method will take the primary key attributes of the class the web service is made for as in parameters. It will then fetch the correct row from EC, use that to locate the values it need to fetch the related class, and then fetch the correct row from the related class. The method will return a pojo of the type of the related class, containing the row with info about the related object.

### **Get related class values using object code**

- get<RoleNameWithCode> (<pk-values>)

Many classes in EC have relations to other classes. If you run the following query, you will get all roles and attributes defining them:

```
select PROPERTY_NAME, REL_CLASS_NAME, ROLE_NAME
from DAO_META where class_name = '<class_name>'
AND ROLE_NAME IS NOT NULL;
```

There will be generated one method per ROLE\_NAME available for the class. The method will be called get<ROLE\_NAME>. The method will take the primary key attributes of the class the web service is made for as in parameters. This method will internally translate the objectCode into corresponding objectId. It will then fetch the correct row from EC, use that to locate the values it need to fetch the related class, and then fetch the correct row from the related class. The method will return a pojo of the type of the related class, containing the row with info about the related object.

### **Security**

The web services is per default set up to use basic http authentication. Encryption is provided through SSL. The container (wildfly) supports many real world scenarios requiring WS-Security functionalities. This includes signature and encryption support through X509 certificates, authentication and authorization through username tokens as well as all ws-security configurations covered by WS-SecurityPolicy specification. This provides the possibility to configure the web services to use the type of security needed for your setup.

### **Access permissions**

The controlling access mechanism for insert, update and delete functionality is governed by the EC tables *BUSINESS\_FUNCTION* and *BF\_COMPONENT* and *BF\_COMPONENT\_ACTION*. Currently these need to be set up specially, since there is no EC screen for configuring their contents.

Once the access for various classes has been set up, the users' access to these classes, can be configured through the User Role configuration screen in EC.

An example of how this can be inserted:

```
insert into t_basis_object (OBJECT_ID, APP_ID, OBJECT_NAME,
OBJECT_TYPE, OBJECT_DESCR) VALUES
('35551','1','/PWEL_DAY_STATUS','CLASS','GIVE ECIS ACCESS TO CLASS
PWEL_DAY_STATUS');
insert into t_basis_access (t_Basis_Access_Id, Role_Id, app_id,
level_id, object_id) VALUES ('79510', 'SYST.ADM', '1', '60', '35551');
insert into business_function
(Business_Function_No,bf_code,jsf_screen_ind,name,url,record_status)
values (9100,'WTEST01','N','PWEL_DAY_STATUS',
'/PWEL_DAY_STATUS','P');
insert into bf_component (bf_component_no,
business_function_no,bf_code,comp_code,class_name,url) values
(91000,9100,'WTEST01','PWEL_DAY_STATUS','PWEL_DAY_STATUS','/PWEL_DAY_
STATUS');
insert into bf_component_action
(bf_component_action_no,bf_component_no,name,url) values
(9100,91000,'PWEL_DAY_STATUS','/PWEL_DAY_STATUS');
```

## Generate and deploy web service

### Setup which class To generate web service

To setup which class you want to generate web services from, go into Configuration->System->Class, and choose the class you want and press "Go" button. Now you will get all the information about the class, and there is a checkbox called "Include Webservice". Tick on this checkbox and save. Do this for all the classes you want to generate web service.

### Run schedule

Create a new schedule; use business action "WebServiceGenerator". This business action will generate your webservice java-code. It will place the code in the wildfly tmp path (..\wildfly-8.2.1.Final\domain\servers\ec-server\tmp\WS).

### Setup build enviroment and build the code

1. Make sure you have the prerequisites section of the document [How to set up development enviroment and build custom\(er\) extensions](#) in place.
2. Unzip ec-sdk.x.x.zip from the same document. In this folder we assume that you unzipped to a folder called ec-sdk.
3. Copy the src folder from your wildfly location (..\wildfly-8.2.1.Final\domain\servers\ec-server\tmp\WS). Go to ..\ec-sdk\custom-webapp and paste the code in here. The file ..\ec-sdk\custom-webapp\src\main\webapp\WEB-INF\web.xml will be overwritten in this process, so if you have code you already have modified here, you will need to merge the two files.
4. Open the file ec-sdk\custom-webapp\pom.xml. Add the following three dependencies to the dependencies section of the xml-file:

```
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>com.ec.frmw</groupId>
  <artifactId>frmw-ecis</artifactId>
  <version>${ec.version}</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.jboss.spec.javaee</groupId>
  <artifactId>jboss-ejb-api_3.2_spec</artifactId>
  <version>1.0.0.Final</version>
  <scope>provided</scope>
</dependency>
```

5. Open command prompt and navigate to your ec-sdk folder. Run the command `mvn clean package`. Verify that the operation is successful.
6. Navigate to the folder ..\ec-sdk\custom-webapp\target\. Here you will find a custom-webapp-<version>.war file. Take this file, and follow the instructions in [How to assemble a customer specific EC Application](#) to build it into your ec-app.ear file.
7. Deploy the resulting ec-app-custom.ear file.

# EC Dashboard Configuration Guide

## Introduction

This document describes how to create (configure) new Widgets so they become available for the end-users in the EC Dashboard screen.

## Database configuration

All widgets made available for end-users needs to be configured in two tables in the EC database.

First, new Widget needs to be defined, as a new row, in CTRL\_DASHBOARD. This table describes key attributes of a widget, like the label and the type of widget to use.

Second, any configuration parameters needs to be added to the table CTRL\_DASHBOARD\_PARAM. Different widgets have different set of possible configuration parameters, that can be used to change how the widget retrieve data, how the widget display data and so on.

## Example

As an example we will create a new Widget, called *Oracle Version*, that displays the Oracle database version number. It will look something like this:



First we need to insert a new row into the database table CTRL\_DASHBOARD. In this table each row represent one widget. Here we defined the name of the Widget and also the widget type (WIDGET\_CLASS). We use *BigWidget* in this example.

### CTRL\_DASHBOARD

WIDGET_CODE	CATEGORY	NAME	LABEL	DESCRIPTION	WIDGET_CLASSES
ORACLEVERSION		OracleVersion	Oracle Version	Display the Oracle Database version number.	com.ec.frmw.jsf.dashboard.wg.BigWidget

The *BigWidget* needs some configuration parameters to be able to retrieve and display data. In this example we use 3 parameters: **QUERY**, **footer** and **text**.

The **QUERY** parameter contains the actual SQL query that the widget will run.

The **footer** parameter contains a literal text that will be displayed as a footer.

The **text** parameter contains an DATAMODEL expression that will be resolved after the widget has retrieved data. The expression 'EcDatamodel.row[0].col[VERSION].datavalue' means: lookup the *datavalue* attribute in the *ECDatamodel* on row 0 for the *VERSION* column.

### CTRL\_DASHBOARD\_PARAM

WIDGET_CODE	NAME	LABEL	RESOLVE_TYPE	PARAMETER_TYPE	PARAMETER_SUB_TYPE	PARAMETER_VALUE
ORACLEVERSION	QUERY					<pre>&lt;renderer&gt; &lt;model class="com .ec.ecomm on.generic model.mode l.web.Gene ricSqlMode l"&gt; &lt;arg name="sqlX ml" value="dat a" valuetype= "subXml"&gt; &lt;data&gt; &lt;sql&gt;SELEC T VERSION FROM PRODUCT_CO MPONENT_VE RSION WHERE product like 'Oracle Database%' &lt;/sql&gt; &lt;/data&gt; &lt;/arg&gt; &lt;/model&gt; &lt;/renderer &gt;</pre>
ORACLEVERSION	footer					Oracle Version
ORACLEVERSION	text		DATAMODEL			EcDatamodel .row[0].col[VE RSION].datav alue

### The QUERY Parameter

The QUERY parameter is an important parameter used in many of the widgets, it is responsible for doing the actual retrieval of data for the widget.

This parameter has to contain a <renderer> element just like in screen xml's and the renderer element must contain a model element, and may contain transformer elements as needed.

PlaceHolders enclosed in \$ will be replaced with the parameter of the same name.

For example \$FROMDATE\$ will be replaced with the value of FROMDATE parameter

**Example using GenericDaoModel :**

```

<?xml version="1.0" encoding="utf-8"?>
<renderer>
    <model
class="com.ec.eccommon.genericmodel.model.web.GenericDaoModel">
        <arg name="daoQueryXml" value="data" valuetype="subXml">
            <data>
                <query>
                    <recordstatus>false</recordstatus>
                    <readonly>true</readonly>
                </query>
                <class name="DASH_INV_VALUES">
                    <property name="DAYTIME" />
                    <property name="OBJECT_ID" />
                    <property name="INV_CODE" />
                    <property name="INV_NAME" />
                    <property name="UOM_CODE" />
                    <property name="INV_CLOSING_QTY" />
                    <property name="DASH_HEADER_QTY" />
                </class>
                <object name="DASH_INV_VALUES">
                    <property datavalue="$PRODUCT$" name="PROD_ID"
operator="=" />
                    <property datavalue="$COMPANY$" name="CO_ID"
operator="=" />
                    <property datavalue="$UOM$" name="UOM_CODE"
operator="=" />
                    <property datavalue="$FROMDATE$" name="DAYTIME"
operator=">=" />
                    <property datavalue="$TODATE$" name="DAYTIME"
operator="<=" />
                    </object>
                    <sort name="DASH_INV_VALUES">
                        <property name="DAYTIME" order="ASC" />
                        <property name="INV_NAME" order="ASC" />
                    </sort>
                </data>
            </arg>
            <arg name="transform"
value="com.ec(pf).screenlet.common.model.web.ModifyAttributesTransforme
r">
                <arg name="property1_name" value="DAYTIME"
valuetype="constant" />
                <arg name="attributel_name" value="viewformatmask"
valuetype="constant" />
                    <arg name="attributel_value" value="yyyy-MM-dd"
valuetype="constant" />
                <arg name="update_object" value="false"
valuetype="constant" />
            </arg>
        </model>
    </renderer>

```

**Example using GenericSqlModel :**

```

<?xml version="1.0" encoding="utf-8"?>
<renderer>
    <model
        class="com.ec.eccommon.genericmodel.model.web.GenericSqlModel">
            <arg name="sqlXml" value="data" valuetype="subXml">
                <data>
                    <sql>
                        select
                            m.DAYTIME,
                            m.OBJECT_ID,
                            m.INV_CODE,
                            m.INV_NAME,
                            m.INV_CLOSING_QTY,
                            m.UOM_CODE,
                            m.DASH_HEADER_VAL
                        from TV_DASH_INV_VALUES m
                        where m.daytime &gt;=
                            trunc(to_date('$FROMDATE$', 'YYYY-MM-DD''T''hh24:mi:ss'), 'MM')
                                and m.daytime &lt; trunc(to_date('$TODATE$', 'YYYY-MM-DD''T''hh24:mi:ss'), 'MM')
                                and m.co_id = '$COMPANY$'
                                and m.prod_id = '$PRODUCT$'
                                and m.uom_code = '$UOM$'
                    </sql>
                </data>
            </arg>
        </model>
    </renderer>

```

### Configuring parameters in the CTRL\_DASHBOARD\_PARAM table

The CTRL\_DASHBOARD\_PARAM table has these columns. Values in each column define how a parameter will be resolved into an actual value.

The resolving of an parameter into an actual value can be done in many ways:

- **Literal.** The value is just entered into the PARAMETER\_VALUE column.
- **Dynamic.** Lookup the value in the datamodel by using an expression. RESOLVE\_TYPE must be 'DATAMODEL' and PARAMETER\_VALUE an expression.
- **User.** Let the end user select a value.

A resolved parameter can also be made available to the resolving of the QUERY parameter, like in the example below. Here the FACILITY parameter is resolved (choosen) by the end user, then used by QUERY parameter when retrieving data.

#### **CTRL\_DASHBOARD\_PARAM**

WIDGET_CODE	NAME	LABEL	RESOLVE_TYPE	PARAMETER_TYPE	PARAMETER_SUB_TYPE	PARAMETER_VALUE	MANDATORY_IND
MYCHART	showLegend	Show Legend	USER	BASIC_TYPE	BOOL		N
MYCHART	FACILITY		USER	EC_OBJECT_TYPE	FCTY_CLASSES_1		Y
QUERY						<renderer>....\$FACILITY\$....</renderer>	
ORACLEVERSION	footer						
ORACLEVERSION	text		DATAMODEL			EcDatamodel.row[0].COL[VERSION].datavalue	

## Access Control

The access url to the dashboard page is :

`/xhtml/pages/dashboard`

This URL is added to the ROOT role with READ(10) accesslevel and to the SYSTADM role with CHANGE(20) accesslevel as default

The individual widgets has access controll on the dashboard page url + /widget\_code, example :

`/xhtml/pages/dashboard/UPTIME`

Widgets need accesslevel 20 to show up

## Widget types

### BigWidget

BigWidget can be used to display any text. The text can be retrieved from the EC database by adding a QUERY parameter, then defining an expression in text parameter to 'select' what to display.

BigWidget will try to display the text as big as possible.

If metadata on the retrieved data has any *verification* attributes, the resulting text will be formatted accordingly. E.g. if the retrieved data has a verificationStatus=error then the resulting text will be displayed as an error (red color).



### **Definition**

Widget definition in CTRL\_DASHBOARD

WIDGET_CLASS
com.ec.frmw.jsf.dashboard.wg.BigWidget

### **Parameters**

Parameters in CTRL\_DASHBOARD\_PARAM

Name	Description	Required	Possible values	Default
QUERY	Renderer-xml that the widget will fetch data from	Y	<renderer> element like in screen xml's.	
footer	Text that will be displayed as a footer.	N	Any text	
text	Text that will be displayed as a footer.	Y	<ul style="list-style-type: none"> <li>• Any text.</li> <li>• If RESOLVE_TYPE is defined as DATAMODEL, then an datamodel expression can be used to select text to display. E.g: <b>Ec Datamodel.row [0].col[VERSIO N].datavalue</b></li> </ul>	

### **ActiveSessionsWidget**

ActiveSessionsWidget will display the current number of active sessions on the EC server.



### **Definition**

Widget definition in CTRL\_DASHBOARD

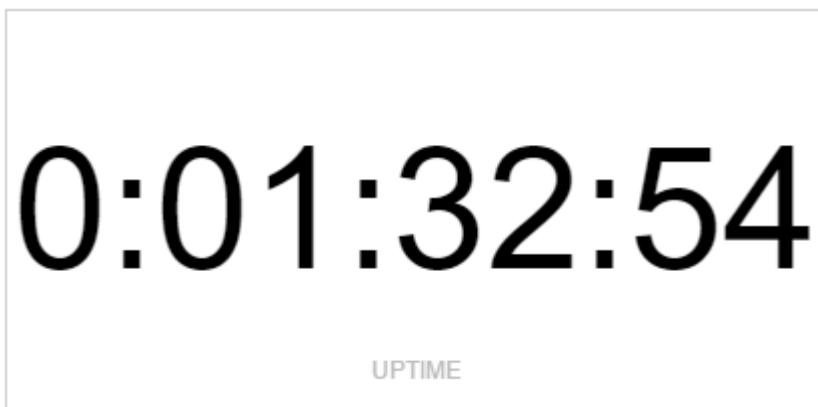
WIDGET_CLASS
com.ec.frmw.jsf.dashboard.wg.ActiveSessionsWidget

### **Parameters**

No parameters necessary.

### **UptimeWidget**

UptimeWidget will display the JavaVM's time since last restart.



#### ***Definition***

Widget definition in CTRL\_DASHBOARD

WIDGET_CLASS
com.ec.frmw.jsf.dashboard.wg.UptimeWidget

#### ***Parameters***

No parameters necessary.

### **WildFlyMgmtWidget**

This widget will execute a WildFly management expression (cli command) against the WildFly application server and display the result.

WILDFLY MGM 01	
used	212102608
committed	643145728
max	4124573696

#### ***Definition***

Widget definition in CTRL\_DASHBOARD

WIDGET_CLASS
com.ec.frmw.jsf.dashboard.wg.WildFlyMgmtWidget

#### ***Parameters***

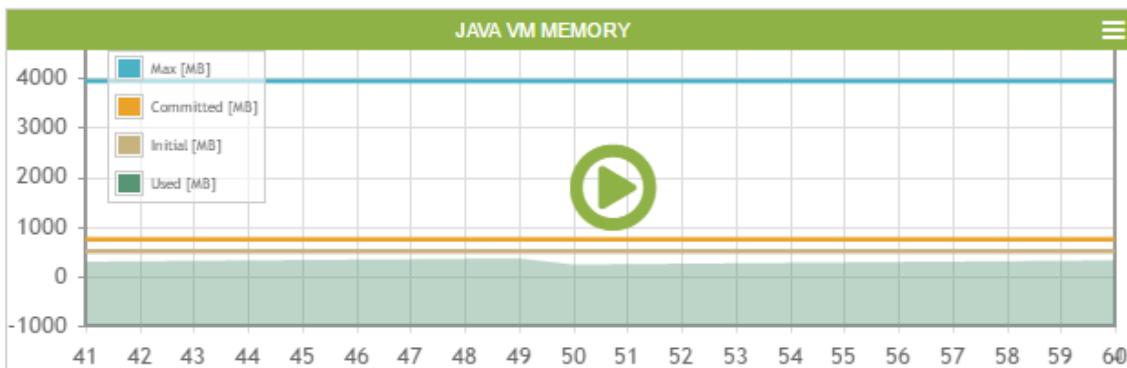
new

#### Parameters in CTRL\_DASHBOARD\_PARAM

Name	Description	Required	Possible values	Default
wildFlyMgmtExpression	WildFly Management Expression	Y	Any CLI expression, e.g this expression will show Java Heap Memory usage: <code>/host=master/server=ec-server/core-service=platform-mbean/type=memory:read-attribute(name=heap-memory-usage)</code>	

#### JavaVmMemoryWidget

JavaVmMemoryWidget will display a continuously refreshing chart of the JavaVm memory allocations



#### Definition

Widget definition in CTRL\_DASHBOARD

WIDGET_CLASS
com.ec.frmw.jsf.dashboard.wg.JavaVmMemoryWidget

#### Parameters

No parameters necessary.

#### TableWidget

TableWidget will display a dataset in a table format.

INVENTORY VALUES BY COMPANY / PRODUCT / UOM							
PERIOD	CODE	OPENING QTY	MOVEMENT QTY	CLOSING QTY	UOM	CLOSING VALUE	CURRENCY
No records found.							

### Definition

Widget definition in CTRL\_DASHBOARD

WIDGET_CLASS
com.ec.frmw.jsf.dashboard.wg.TableWidget

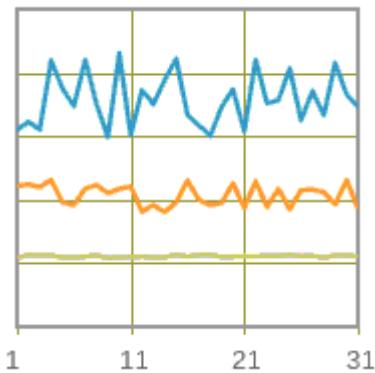
### Parameters

Parameters in CTRL\_DASHBOARD\_PARAM

Name	Description	Required	Possible values	Default
QUERY	Renderer-xml that the widget will fetch data from	Y	<renderer> element like in screen xml's.	
header	Widget header text	N	Any text	Empty
showHeader	Display column headers	N	EC boolean (Y,N)	Y

## LineChartWidget

LineChartWidget will display a dataset as line charts. It supports row based or column based series.



### Definition

Widget definition in CTRL\_DASHBOARD

WIDGET_CLASS
com.ec.frmw.jsf.dashboard.wg.LineChartWidget

**Parameters**

Parameters in CTRL\_DASHBOARD\_PARAM

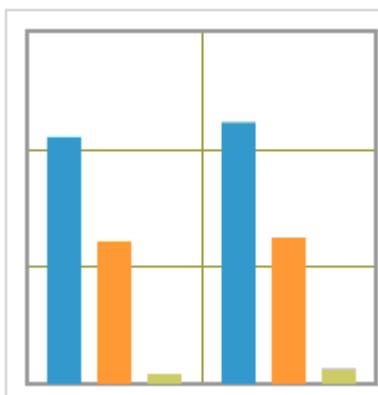
Name	Description	Required	Possible values	Default
QUERY	Renderer-xml that the widget will fetch data from	Y	<renderer> element like in screen xml's.	
header	Widget header text	N	Any text	Empty
SeriesMode	Defines if series are pr. row or pr. Column	Y	ROWS,COLUMNS	
ColSeriesNames	Datamodel columns defining individual series	Y if seriesMode=COLUMNS	Comma speareated list of datamodel columnnames.	
ColXColname	Datamodel column that contains X-axis values	Y if seriesMode=COLUMNS	Any text, that resolves to a datamodel columnname	
RowSeriesColName	Datamodel column that contains series identifier (f.eks OBJECT_CODE)	Y if seriesMode=ROWS	Any text, that resolves to a datamodel columnname	
RowXColName	Datamodel column that contains X-axis values	Y if seriesMode=ROWS	Any text, that resolves to a datamodel columnname	
RowYColName	Datamodel column that contains Y-axis values	Y if seriesMode=ROWS	Any text, that resolves to a datamodel columnname	
YaxisMinSeries1	Yaxis 1 Min	N	Any number	Dynamic
YaxisMaxSeries1	Yaxis 1 Max	N	Any number	Dynamic
YaxisMinSeries2	Yaxis 2 Min	N	Any number	Dynamic
YaxisMaxSeries2	Yaxis 2 Max	N	Any number	Dynamic
YaxisMinSeries...	There can be upto 9 series, set the number accordingly for configuration	N	Any number	Dynamic
YaxisMaxSeries...	There can be upto 9 series, set the number accordingly for configuration	N	Any number	Dynamic

ShowYaxisSeries1	Show Y axis 1	N	EC boolean (Y,N)	Y
ShowYaxisSeries2	Show Y axis 2	N	EC boolean (Y,N)	Y
ShowYaxisSeries...	There can be upto 9 series, set the number accordingly for configuration	N	EC boolean (Y,N)	Y
ShowXaxis	Show X axis	N	EC boolean (Y,N)	Y
ShowPointLabels	Show point Labels	N	EC boolean (Y,N)	Y
ShowMarker	Show markers	N	EC boolean (Y,N)	Y
LegendPosition	Legend Position	N	n,ne,e,se,s,sw,w,nw	e
Animate	Animate	N	EC boolean (Y,N)	N

### BarChartWidget

BarChartWidget will display a dataset as bar charts. It supports row based or column based series.

It allows stacking of bars and supports rendering some series as lines.



### Definition

Widget definition in CTRL\_DASHBOARD

WIDGET_CLASS
com.ec.frmw.jsf.dashboard.wg.BarChartWidget

### Parameters

Parameters in CTRL\_DASHBOARD\_PARAM, Uses the same configuration parameters as LineChartWidget, in addition these parameters can be used :

Name	Description	Required	Possible values	Default
BarChartStacked	Stack bars	N	EC boolean (Y,N)	N
ShowAsLineSeries1	Show series as lines instead of bar	N	EC boolean (Y,N)	N
ShowAsLineSeries2	Show series as lines instead of bar	N	EC boolean (Y,N)	N
ShowAsLineSeries...	There can be upto 9 series, set the number accordingly for configuration	N	EC boolean (Y,N)	N

## PieChartWidget

PieChartWidget can be used to display data in slices format. Once the configuration is done the widget will look like this.



### Definition

Widget definition in CTRL\_DASHBOARD

WIDGET_CLASS
com.ec.frmw.jsf.dashboard.wg.PieChartWidget

### Parameters

Parameters in CTRL\_DASHBOARD\_PARAM

Name	Description	Required	Possible values	Default
QUERY	Renderer-xml that the widget will fetch data from	Y	<renderer> element like in screen xml's.	
header	Widget header text	N	Any text	Empty
categoryColumn	Name of database column having category names.	Y	Any text, that resolves to a datamodel columnname	
categoryValueColumn	Name of database column having value for the respective category in a row.	Y	Any text, that resolves to a datamodel columnname	
showDataLabels	Set to 'Y' if user want show data labels on the slices of the pie.	N	EC boolean (Y,N)	Y
labelType	Type of label to place on the pie slices.	N	label,value,percent	label
fill	Whether to fill the slices.	N	EC boolean (Y,N)	Y



# BPM Technical Documentation

This documentation contains instructions and configuration examples of ecbpm shipped with EC 11.1.

- BPM Introduction
- Architecture
- Single Sign On
- Repositories
- Authoring Processes
- Process Execution and Management
- Business Action Invocation
- Process Action
- User Tasks and To-do List
- Samples
- API

# BPM Introduction

## Introduction

The Energy Components BPM (**ecbpm** in following texts) provides process automation functionality. It is an extension to EC and can be optionally deployed.

Current version of *ecbpm* utilizes jBPM, an open source process automation component as process engine. jBPM is a light-weight process executing engine, and in addition to that, it also provides a complete solution to process authoring, collaboration, management and process instance management. Following table shows the process automation component and versions used in EC releases:

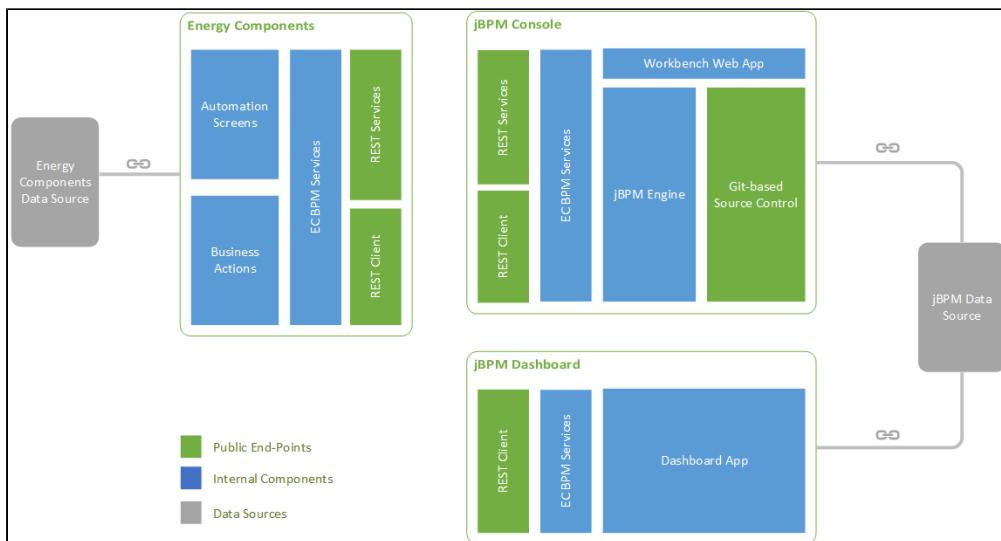
EC Release	Process Automation Component	Version	Supported Process Language
10.1 - 10.4	jBPM	3.2	JPDL
11.0	jBPM	6.1.0.Final	BPMN 2
11.1	jBPM	6.2.0.Final	BPMN 2

# Architecture

In this section, we describe the BPM architecture on EC 11.1. Architecture might change between releases, please refer to documentations from previous releases for earlier architecture.

The core of *ecbpmp* is a tailored version of the jBPM Console application, which shall be deployed standalone. All process definitions are maintained inside this application. Process instances runs by the process execution engine hosted by the application. To provide tailored functionality, for example invoking business actions, EC user integration, single-sign-on, etc., EC jBPM Console extension is deployed inside the jBPM Console application.

The following diagram shows the architecture of *ecbpmp*:



## Systems

EC BPM consists of three systems, Energy Components, jBPM Console and jBPM Dashboard.

### Energy Components

Energy Components is the main entrance for end-users to execute and manage process instances. All BPM functionalities should be accessed via either UI or API on EC, except for process authoring and process source code management.

Energy Components also participates in process execution when it comes to business action invoking. See section *Invoking Business Action In Process* for more information.

### Built-in Schedules

Following schedules are installed to support EC BPM, some of them may need to be configured and enabled before such functionality can be used.

Schedule	Description	Dependent Function
JbpmCallback	Handles callbacks from jBPM Engine, for example, invoking Business Action. This schedule needs to be configured and enabled before dependent function can be used. This schedule should only be used by BPM internally.	Invoking Business Action from process instances.

### jBPM Console

jBPM Console is the core of EC BPM. The system maintains all BPM data, including process source code and process instance information. BPM Engine is running inside the console. jBPM console is deployed as a standalone web application, and it may also be deployed to a separate server group than EC, see the user manual for more information.

Complete documentation on jBPM Console can be found at <https://docs.jboss.org/jbpm/v6.1/userguide/>.

It is suggested that use jBPM Console only for development purpose, as changes may come in future ecbpm releases.

### jBPM Dashboard

jBPM Dashboard is an extension to jBPM Console, providing process instance and task statistic and overview. jBPM Dashboard and jBPM Console should always be deployed to the same server group, as a hard requirement from jBPM Console. These two systems also share the same data source.

It is suggested that use jBPM Dashboard only for development purpose, as changes may come in future ecbpm releases.

### Services

Services are the internal functionality provider systems. They expose core API to UI and external systems. Services on different systems may also communicate with each other via REST services. To have BPM screens and APIs working properly, all BPM systems have to be up and running.

### Data Source

Energy Components do not access BPM data source directly, it gets or indirectly modifies data via REST services from jBPM Console or jBPM Dashboard.

jBPM Console and jBPM Dashboard are the direct users of BPM data source.

EC BPM data source is independent of EC data source. It is possible to deploy BPM data structure to a separate data source without EC data structure deployed. However, in current deployment scripts, the two data structures are always deployed together.

### Git-based Source Control

jBPM Console maintains a source control system supporting Git protocol. It enables using external IDEs to maintain process source code as well as team collaboration.

## Single Sign On

Single-sign-on across the three systems are automatically configured by the deployment script. See the user manual for more information.

Picketlink is the underlying component for single-sign-on. The version used in EC 11.0 is 2.6.1.Final.

### Roles

Picketlink supports federated SSO with SAML, which is currently implemented in EC. Such scenario has two roles:

- Identity Provider (IP) - The Identity Provider is the authoritative entity responsible for authenticating an end user and asserting an identity for that user in a trusted fashion to trusted partners (quoted from <http://docs.jboss.org/picketlink/2/latest/reference/html-single/>).
- Service Provider (SP) - The PicketLink Service Provider relies on the PicketLink Identity Provider to assert information about a user via an electronic user credential, leaving the service provider to manage access control and dissemination based on a trusted set of user credential assertions (quoted from <http://docs.jboss.org/picketlink/2/latest/reference/html-single/>).

The three systems in EC BPM are implemented as:

- Energy Components – Identity Provider
- jBPM Console, jBPM Dashboard – Service Provider

### Exposed Configuration Items

EC BPM has externalized some Picketlink configurations as system properties, these can be changed as needed in domain.xml in Wildfly.

Property Name	Description
picketlink.encrypt	“false” if no encryption is needed. Use “true” to turn on the encryption.
picketlink.keystore.url	The path to the Java Keystore for encryption. Required when picketlink.encrypt is set to “true”.
picketlink.keystore.pass	The keystore password. Required when picketlink.encrypt is set to “true”.
picketlink.signing.key.pass	The signing key password. Required when picketlink.encrypt is set to “true”.
picketlink.trust.domains.1	When multiple addresses are assigned to jBPM Console web application, this property can be used to add new address to trust list on IP.
picketlink.trust.domains.2	When multiple addresses are assigned to jBPM Console web application, this property can be used to add new address to trust list on IP.
picketlink.identity_provider.url	The public URL to EC. When implementing custom IP, this property can be used to direct to the new IP.

### Securing SAML Messages

As SAML are transmitted across applications as part of HTTP request, it is vulnerable to network attack. Picketlink supports using Java Keystore files to encrypt SAML messages.

#### Keystore File Generation

User tokens can be encrypted by providing a Java Keystore file. Java provides a key tool for key store generation.

The Java Keytool can be found under the bin directory on Java installation path. To execute the Keytool, following parameters are usually used:

Parameter Name	Description
-alias	The alias of the key
-keyalg	The encryption algorithm used to encrypt the key
-keystore	The path to the keystore file to be generated
-keysize	The size of the key

For example:

```
keytool -genkey -alias com.ec -keyalg RSA -keysize 2048 -keystore
ec.keystore
```

You may be asked to provide more information by the program during the generation process.

For more information, see <https://docs.oracle.com/javase/7/docs/technotes/tools/solaris/keytool.html>.

### Configuration

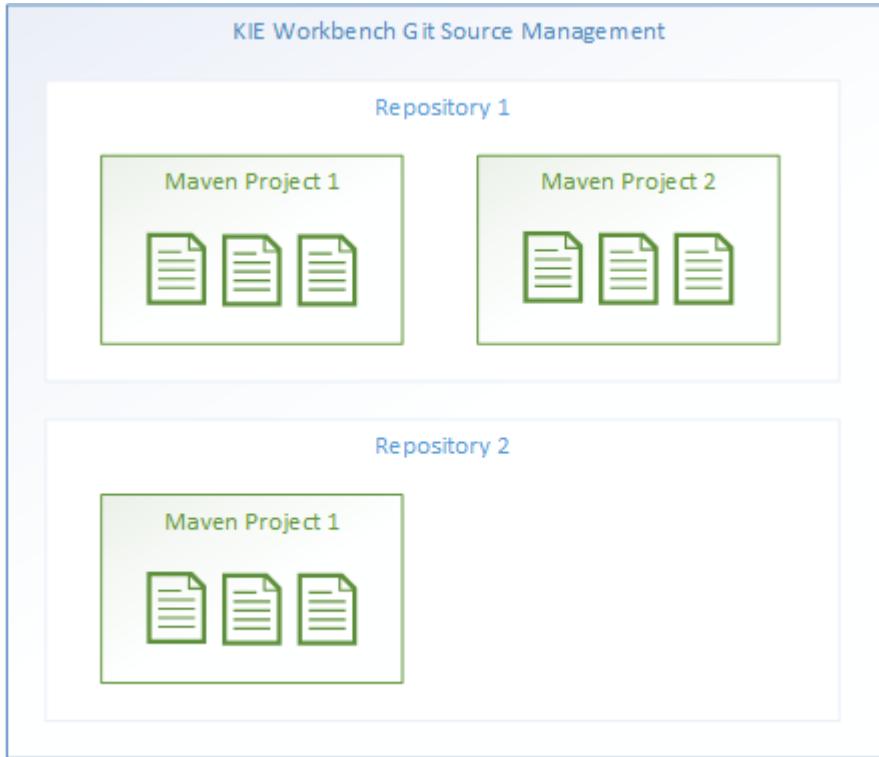
To enable EC for single-sign-on encryption with the generated key, following system properties in Wildfly's domain.xml file need to be configured:

Property Name	Description	Example
picketlink.encrypt	"false" if no encryption is needed. Use "true" to turn on the encryption.	false true
picketlink.keystore.url	The path to the Java Keystore for encryption. Required when picketlink.encrypt is set to "true".	C:\ec.keystore
picketlink.keystore.pass	The Keystore password. Required when picketlink.encrypt is set to "true".	
picketlink.signing.key.pass	The signing key password. Required when picketlink.encrypt is set to "true".	

# Repositories

## Repository, Project, Process

A simple KIE Workbench Source Management structure can be seen in the following diagram



## Repository

ecbpm ships with a Git repository server (hosted inside the Workbench) for process storing and management. All processes that need to be deployed to ecbpm have to be stored in a repository. Files in repositories are version-ed, change-traceable and shareable across a development team. Developers can also work on the same process file without affecting each other by using the "branch" feature.

The KIE Workbench provides basic repository management functionality. This document introduces the repository operations via the KIE Workbench. Advanced Git repository operations, such as, branch creation and merging, can be done via a Git client, and are not covered by this document.

## Project

A project is a Java compile unit resides in a repository. Projects are the direct container of Java source files and process files, in other words, process files are not allowed to be placed outside a project.

In ecbpm, projects use maven structure. Documentation on maven can be found at <http://www.apache.org/>.

Project is the smallest unit for process compile and deployment. To deploy a process, it must be compiled and deployed together with the whole project.

## Process

Process files contains work definition of a automation process. ecbpm requires a process file to be written using BPMN2 language. See website <http://www.bpmn.org> for introduction on BPMN2 standard.

Process files should be grouped into Projects and placed in src\main\resources directory.

When deployed, a process is identified by its ID and its project artifact ID. It is not allowed to have processes with same ID in a project, the compiler will error if such processes are found.

## Pre-shipped Repositories

ecbpm ships with two repositories as a start-up point:

- ecbpm-processes-sample - This repository contains sample projects for learning ecbpm. Files in this repository can be freely edited. To restore the original source files, delete the repository and restart Wildfly.
- Default - An empty repository as a start point. This repository can be used for new processes, users can also create their own repositories.

## Organizational Units

KIE Workbench provides repository identification feature called Organizational Units. A Organization Unit is basically a owner of repositories. As this feature currently is not linked to EC's user role, it does not mean much to assign repositories to different Organizational Units. However, it does help to label the ownership of repositories.

EC ships with a default Organizational Unit called "ec", this can be used by any repository.

To manage Organizational Units, use menu Authoring -> Organizational Units

## List all repositories

To list all repositories, use KIE Workbench menu Authoring -> Administration, Repositories -> List

The screenshot shows the KIE Workbench interface with the 'Repositories' tab selected in the top navigation bar. On the left, the 'File Explorer' sidebar shows a tree view with 'Repositories' expanded, containing 'ecbpm-example', 'ecbpm-processes-sample', and 'Default' branches. Each branch has sub-folders like '.sample' and 'readme.md'. The main area is titled 'RepositoryEditor' and displays three entries:

- ecbpm-example**: General Information [empty]. URL: git://localhost:9418/ecbpm-example. Available protocols: git ssh. Branch: master. Buttons: Update, Delete.
- ecbpm-processes-sample**: General Information [empty]. URL: git://localhost:9418/ecbpm-processes-sample. Available protocols: git ssh. Branch: master. Buttons: Update, Delete.
- Default**: General Information [empty]. URL: [empty]. Available protocols: [empty]. Branch: [empty]. Buttons: [empty].

## New repository

To create a new repository, use KIE Workbench menu Authoring -> Repositories -> New Repository

The screenshot shows the KIE Workbench interface with the 'Repositories' tab selected in the top navigation bar. On the left, the 'File Explorer' sidebar shows a tree view with 'Repositories' expanded, containing 'ecbpm-processes-sample' and '.sample' branches. A context menu is open over the 'Repositories' entry, listing 'List', 'Clone repository', and 'New repository'. The 'New repository' option is highlighted with a blue background.

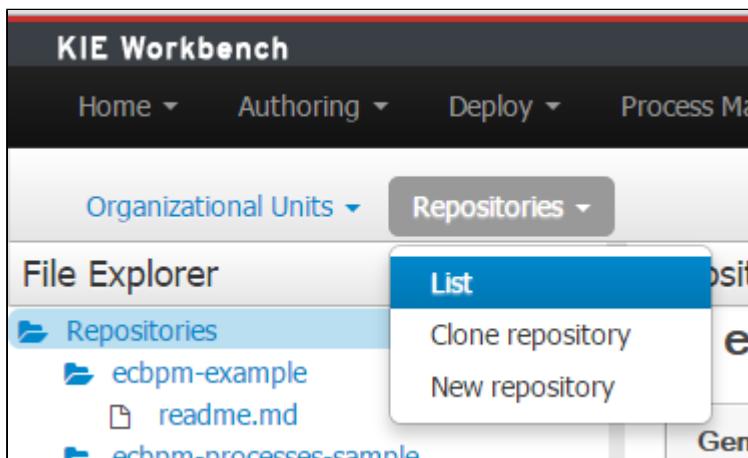
A new modal window shows up with fields

- Repository Name - The name of the repository. The name of the repository only contain alphabets, numbers and dashes "-".
- In Organizational Unit - The owner of this repository. "ec" can be used if no Organizational Unit is created.

Click Finish, the repository will be created. Note that the new repository will not show up in File Explorer until the page is refreshed.

## Deleting a repository

To delete a repository, use KIE Workbench menu Authoring -> Repositories -> List



This brings up the repository list. To delete a repository, click "Delete" button. Note that repository deletion is not recoverable, make sure all files are backed-up before making this operation.

## Clone a Report Repository into KIE Workbench

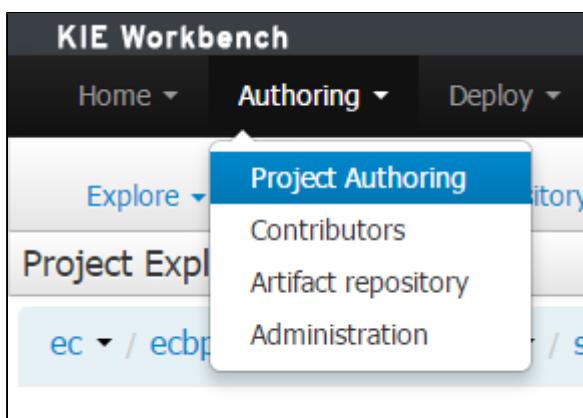
To clone a remote repository, e.g. a repository hosted on Github, into KIE Workbench, use link Authoring -> Administration -> Repositories -> Clone repository.

A model should show up with fields:

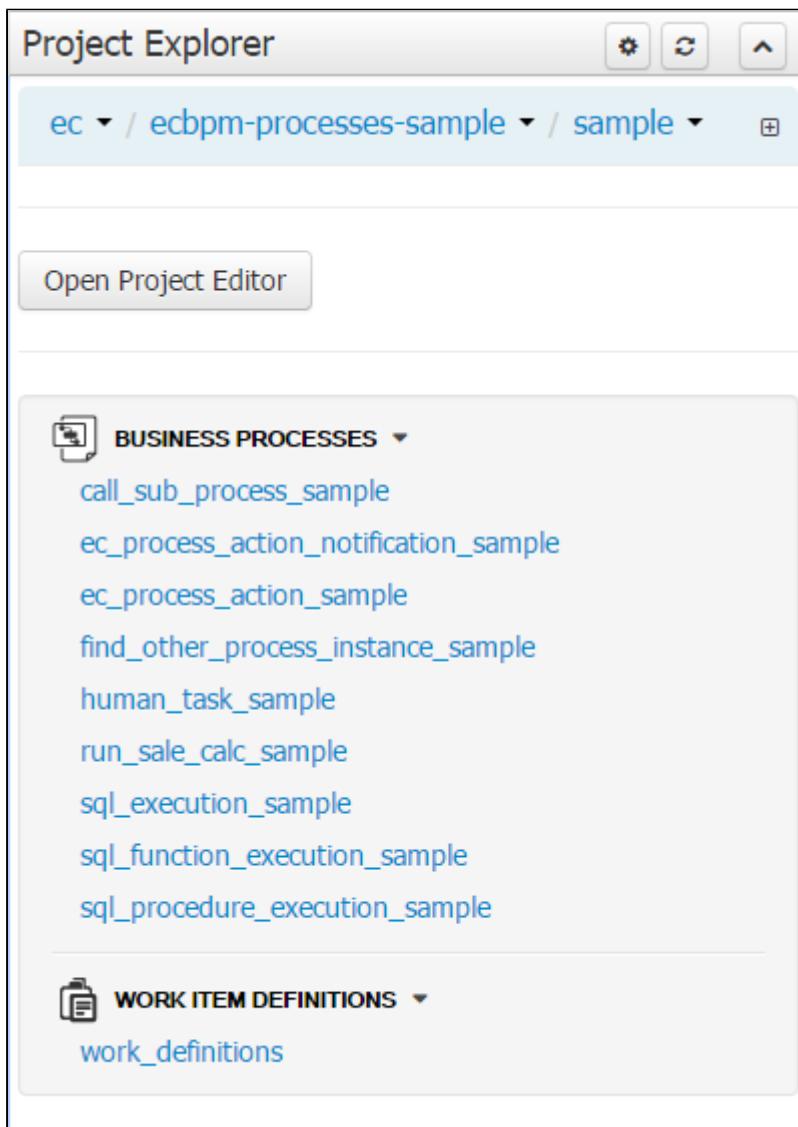
- Repository Name - The new name of the repository
- Organizational Unit - The owner of the cloned repository
- Git URL - The public Git URL where the remote repository is hosted
- User Name - If the remote repository is private, provides the user name for access
- Password - If the remote repository is private, provides the password for access

## Navigating through repositories

The KIE Workbench provides a Project Explorer and various designers for editing project files. To open the Project Explorer, use menu Authoring -> Project Authoring



On the left of the screen, the Project Explorer should be available, similar to the diagram below:



The project explorer has two views, Project View and Repository View. The screenshot shows above is Project View.

- Project View - Provides clean groups of files by file types, for example, Business Processes and Work Item Definitions
- Repository View - Shows file tree as it is stored in the repository

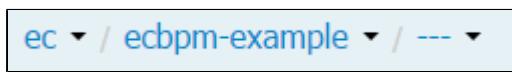


To switch between the views, click button .

The navigation bar on top shows the displaying project path, the three elements are Organizational Unit / Repository / Project.



For a repository that doesn't have a project, the navigation bar shows like

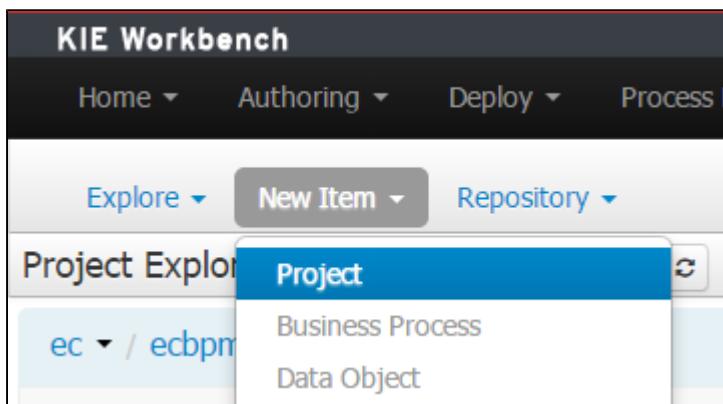


Following sections has instruction on how to create a new project.

## New Project

### Creation

To create a new project, follow menu Authoring -> Project Authoring, select the target repository for new project in Project Explorer, then use menu item New Item -> Project



A model window should show up with fields

- Project - The project name

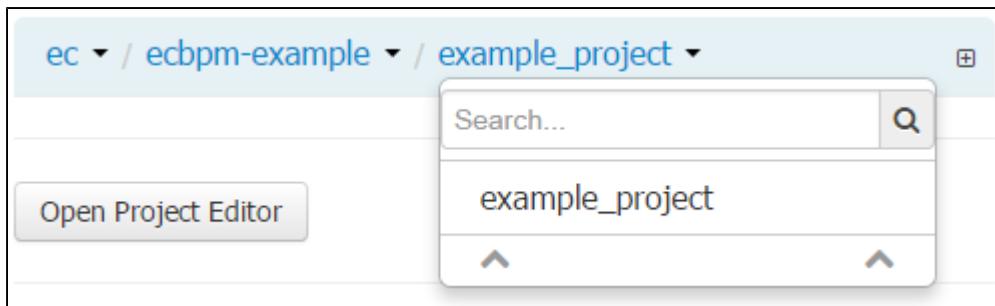
Click OK, New Project Wizard should show up

The screenshot shows the 'New Project' dialog box. On the left, there's a sidebar with a checkmark icon and 'New Project Wizard'. The main area has two sections: 'Project General Settings' and 'Group artifact version'. In 'Project General Settings', there's a 'Project Name' field containing 'example\_project' and a 'Project Description' field with a placeholder 'Insert a project description for documentation purposes ...'. In 'Group artifact version', there are three fields: 'Group ID' (containing 'ec'), 'Artifact ID' (containing 'example\_project'), and 'Version' (containing '1.0.0'). To the right of these fields are examples and help icons. At the bottom of the dialog are buttons for '< Previous', 'Next >', 'Cancel', and a blue 'Finish' button with a checked checkbox.

- Project Name - The project name
- Project Description - The project description
- Group ID - The group ID of the new project. Group ID is required for maven projects, which is to distinguish projects with the same name created by different parties. This is often named as `com.organization_name.project_name` or `com.organization_name`.
- Artifact ID - The artifact name of this project after it is compiled. An artifact is a result of compilation of a project, and usually a project has one artifact and the name is usually the same as the project name.
- Version - The version of the artifact. When a major change is done, it is usually required to change the version number of the project to reflect it. Note that ecbpm allows multiple versions to be deployed at the same time, and when a new version is deployed, a new Process Template should be created for that.

When the configuration is done, click Finish button.

The new project will not show up in the list immediately, click  button to refresh, then use the navigator to select the new project.

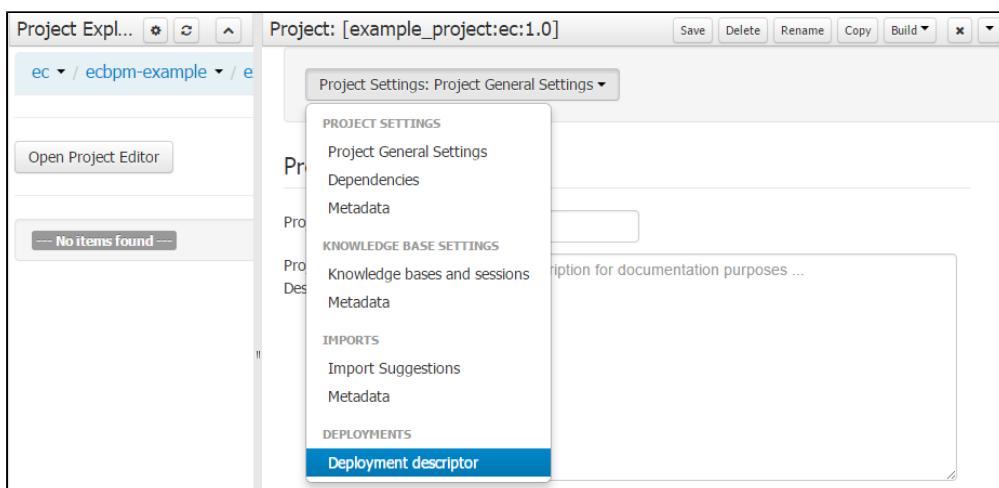


### Extra Configuration (Mandatory)

ecbpm requires some additional configuration to prepare the project. Follow sections below on how they can be done.

#### **Deployment Descriptor**

Open Deployment Descriptor by using Open Project Editor button -> Project Settings: Project General Settings (Drop-down) -> Deployment Descriptor



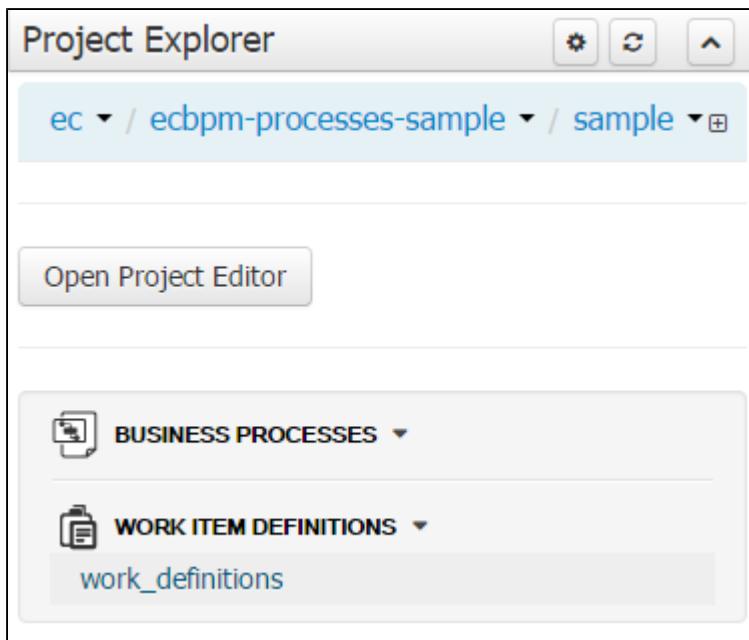
Following changes are required on this page:

1. Set *Runtime strategy* to **PER\_REQUEST**
2. Verify *Persistence unit name* is **org.jbpm.domain**
3. Verify *Persistence mode* is **JPA**
4. Verify *Audit mode* is **JPA**
5. Remove **all** items from Work Item handlers list
6. Click Save button on top

#### **Work Item Definition**

ecbpm ships with some Work Items for process to interacting with EC. To make them available for processes, a work item definition file needs to be created. This file can be copied from ecbpm-processes-sample project. Follow the following step to create the file.

1. Navigate to project ec/ecbpm-processes\_sample/sample
2. Verify the Project Explorer is in *Project View*
3. Open section Work Item Definitions
4. Click file "work\_definitions"



5. Copy all the content on the right to Clip Board
6. Navigate back to the new project
7. Click menu New Item -> Work Item definition, use any name (for example "work\_definitions") for the new item, and click OK
8. The new file should be created and open on the right.
9. Replace all its content with one in Clip Board
10. Click Save button

next

# Authoring Processes

## IDE Alternatives

When a project is ready, processes can be created in it. However, as the KIE Workbench hosts a GIT repository, it is possible to use a 3rd party IDE that support Git to edit processes.

### Inside KIE Workbench

The KIE Workbench provides a Web-based process IDE, which is sufficient for common processes as well as processes need interaction with EC. The designer interact directly with the underlying Git Repository and does commit and pushes in back-end. This documentation focuses on this designer, and other designers should follow the same principal when it comes to process authoring.

For more information on instructions of the editor, see <https://docs.jboss.org/jbpm/v6.2/userguide/chap-designer.html>.

### Eclipse and other 3rd Party IDEs

This section takes Eclipse as example on how to edit processes outside of the Workbench. Operations on other IDE varies but they should follow the same principal.

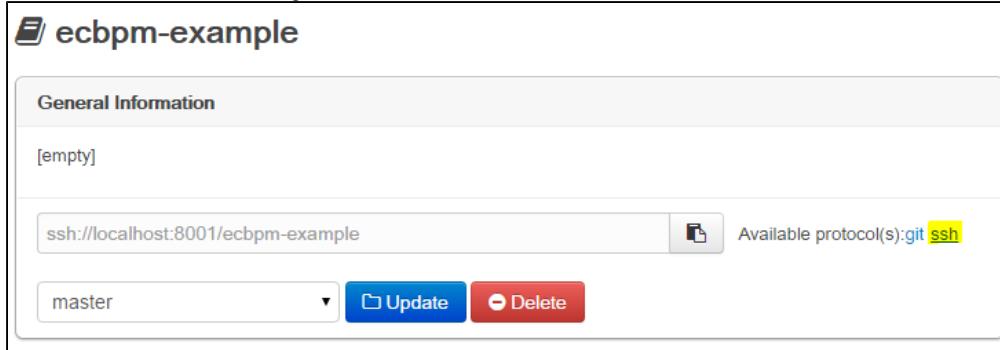
#### *Install jBPM designer plugin*

The jBPM project maintains an Eclipse designer plugin, which supports editing processes in Eclipse. Follow instructions on [https://docs.jboss.org/jbpm/v6.2/userguide/jBPM\\_Eclipse\\_JBPM.html](https://docs.jboss.org/jbpm/v6.2/userguide/jBPM_Eclipse_JBPM.html) to have it installed.

#### *Get Git access info*

Before cloning the repository in Eclipse, the Git address needs to be noted down. Follow the steps below.

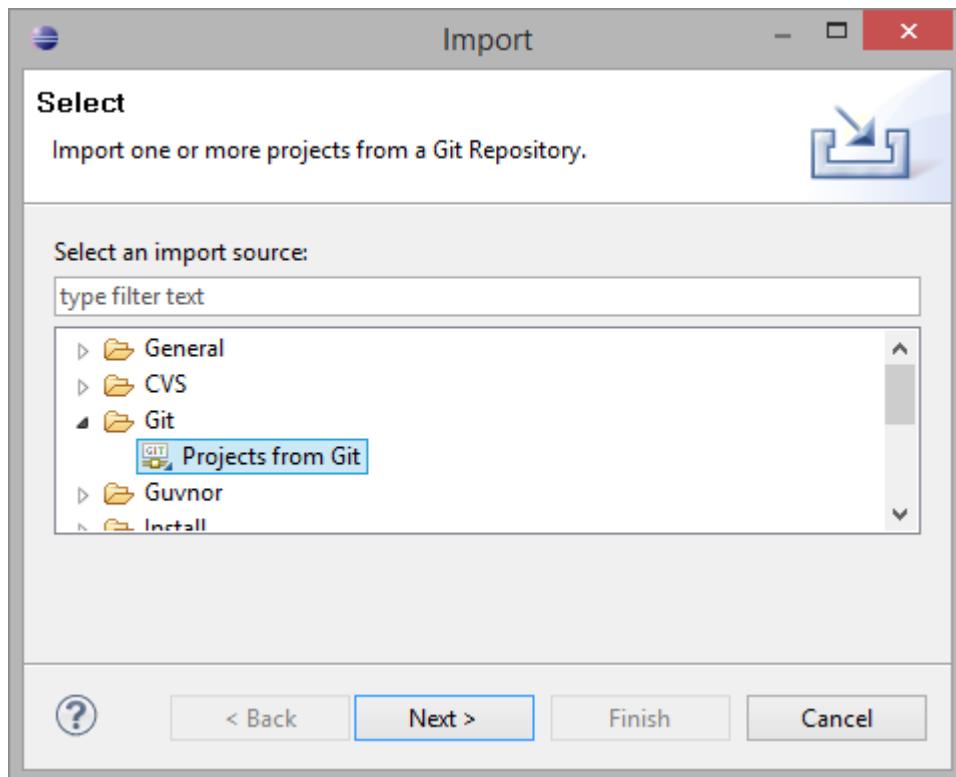
1. Open the repository list page via Authoring -> Administration, then Repositories -> List
2. Find the repository needs to be accessed in Eclipse
3. Click the "ssh" link beside the git address field



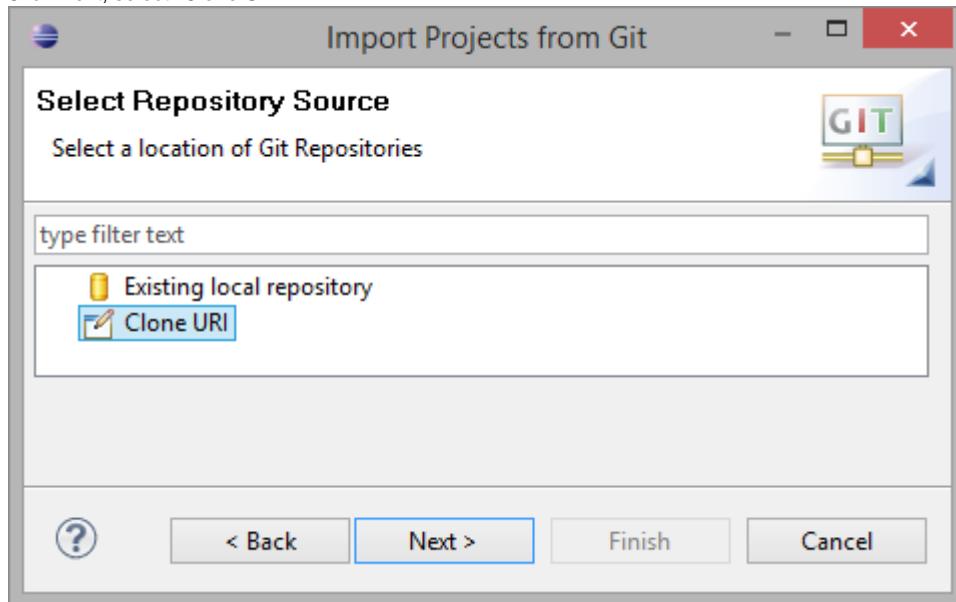
4. Copy the ssh address (in this case, it is "ssh://localhost:8001/ecbpm-example")

#### *Clone the repository in Eclipse*

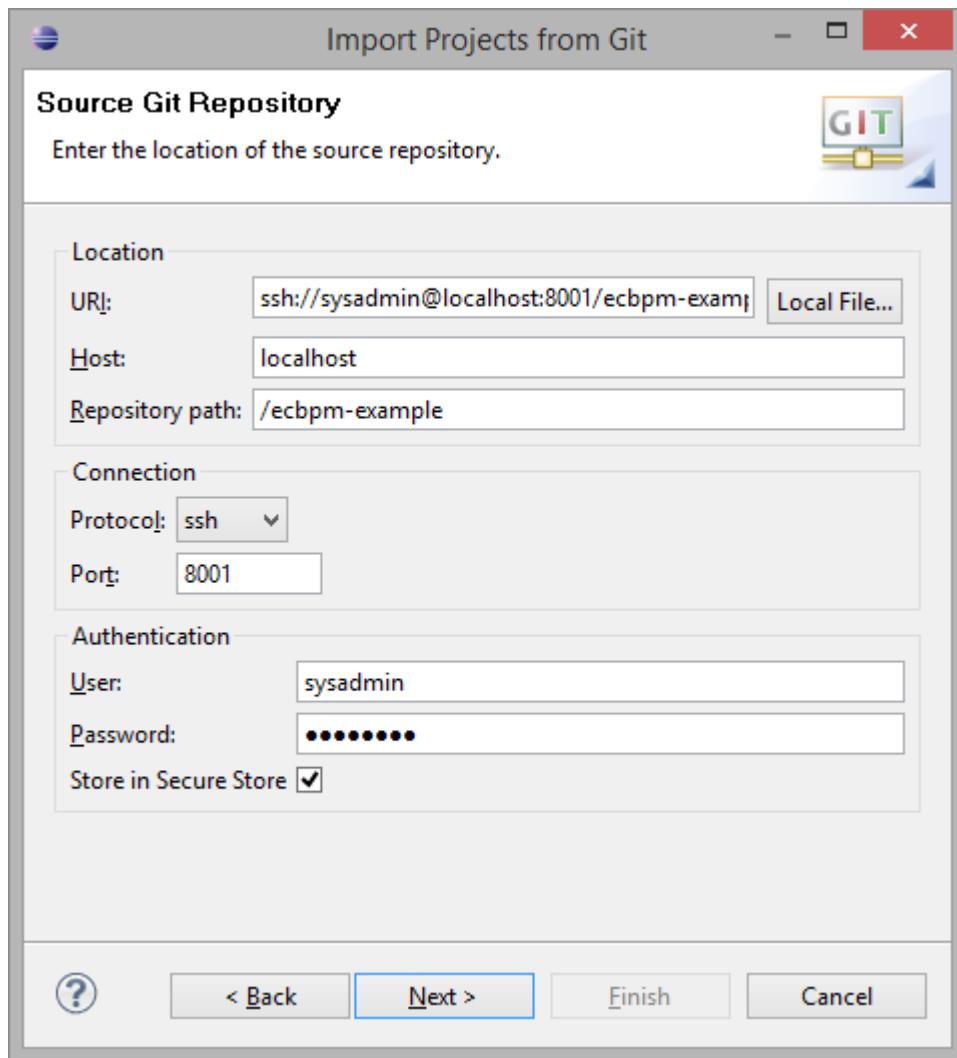
1. Open Eclipse, and use menu File -> Import..., which brings up the Import dialog. Choose Git -> Projects from Git as shown below:



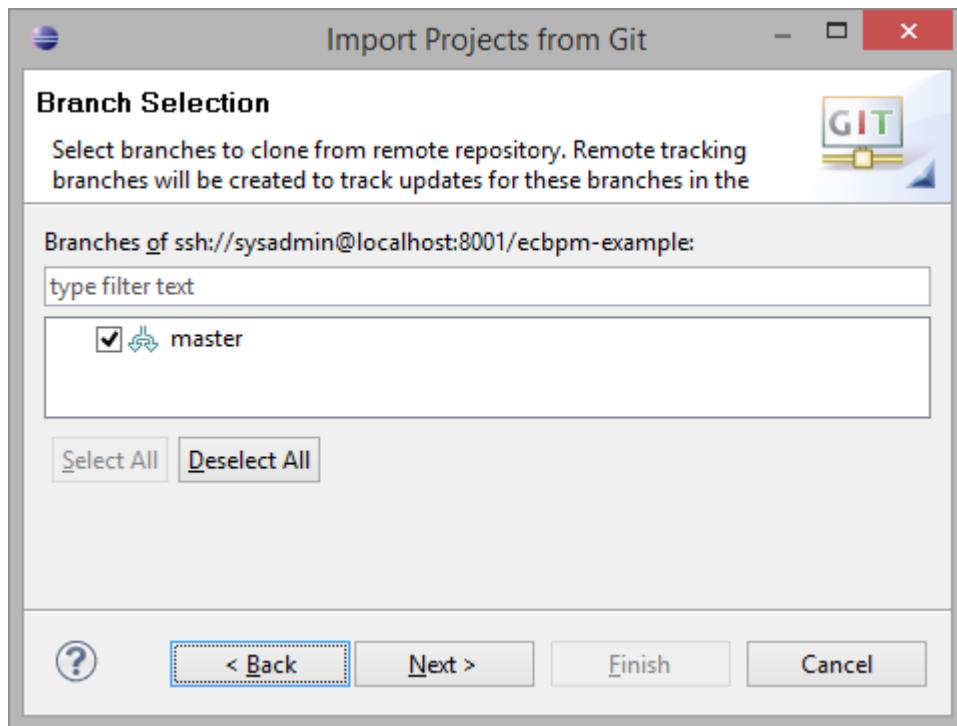
2. Click Next, select "Clone URI":



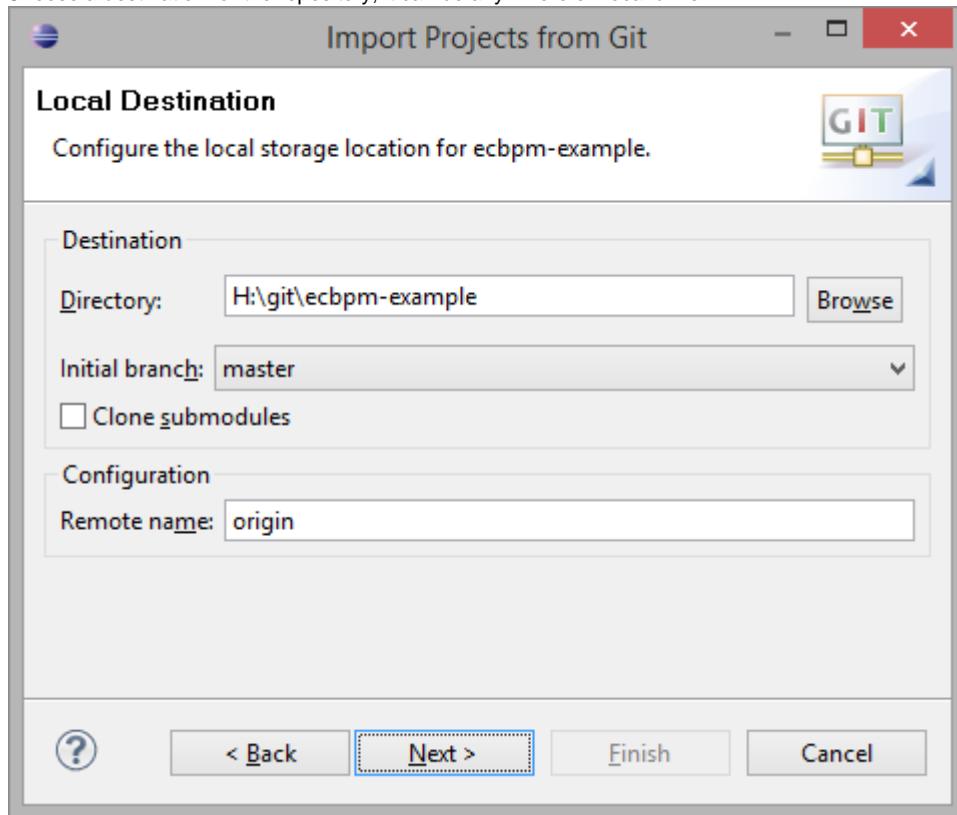
3. In next screen, paste the noted address in previous section to the URL field.  
Put in EC user name and password. The user has to have *JBPM.ADMIN* or *JBPM.DEVELOPER* user role.



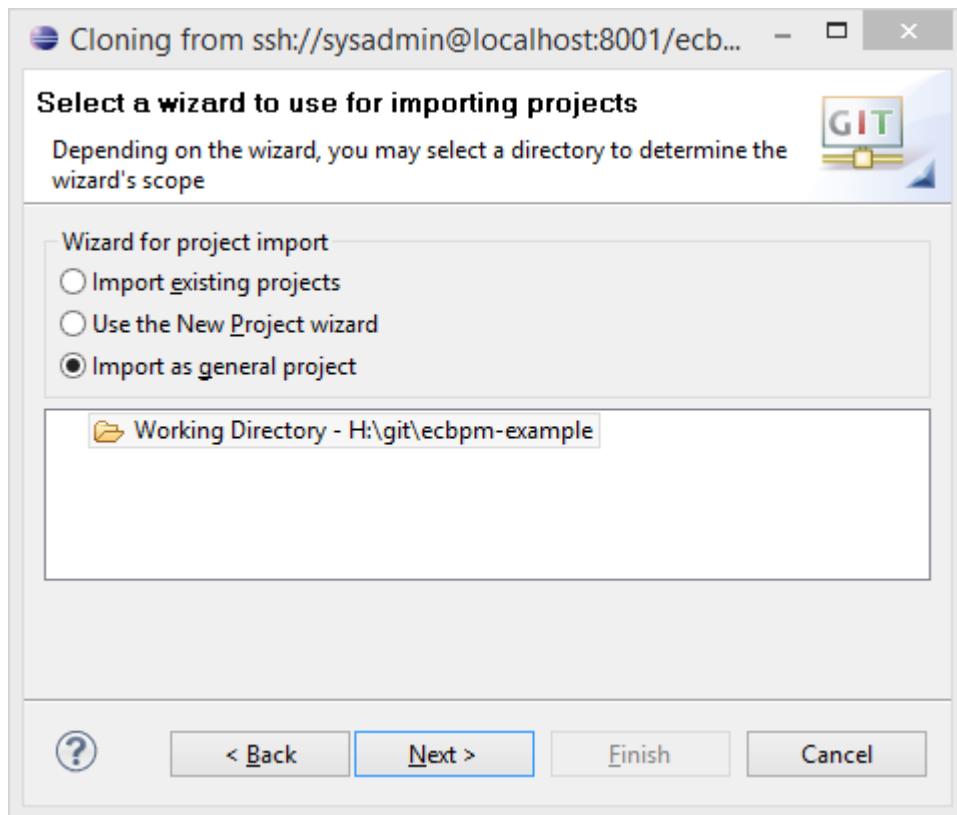
4. Click Next
5. Click Yes if it asks for permission to accept the server identity
6. Select the master branch, and click Next



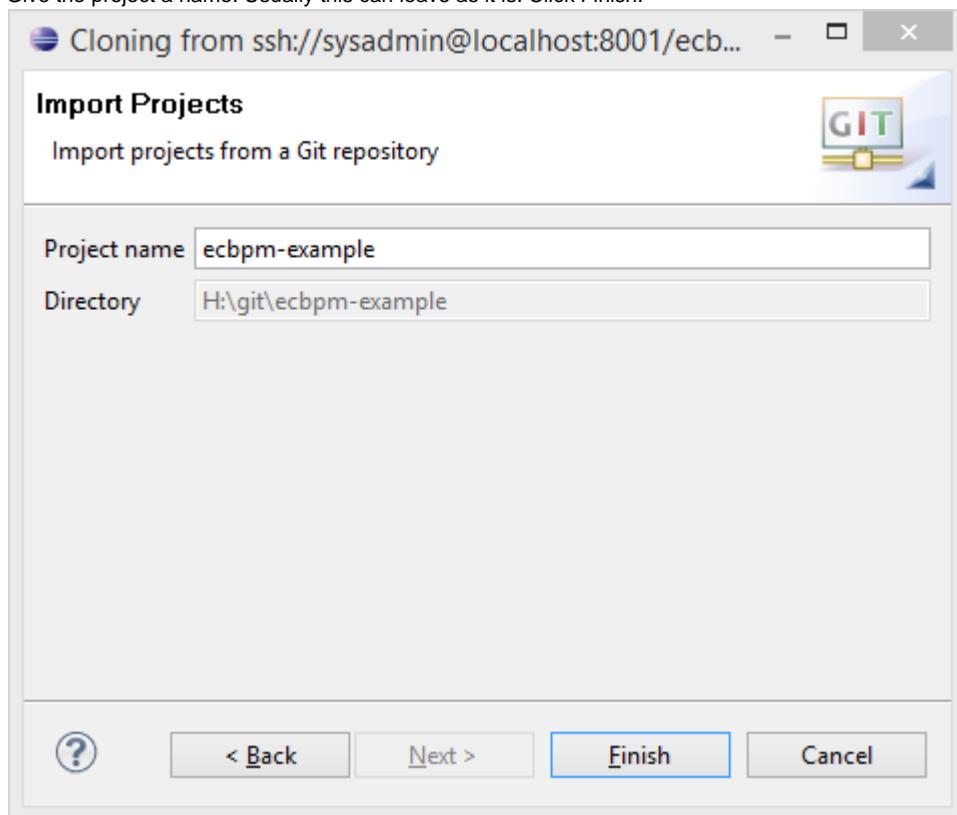
7. Choose a destination for the repository, it can be any where on local drive.



8. Click Next
9. Select "Import as general project", and click Next.

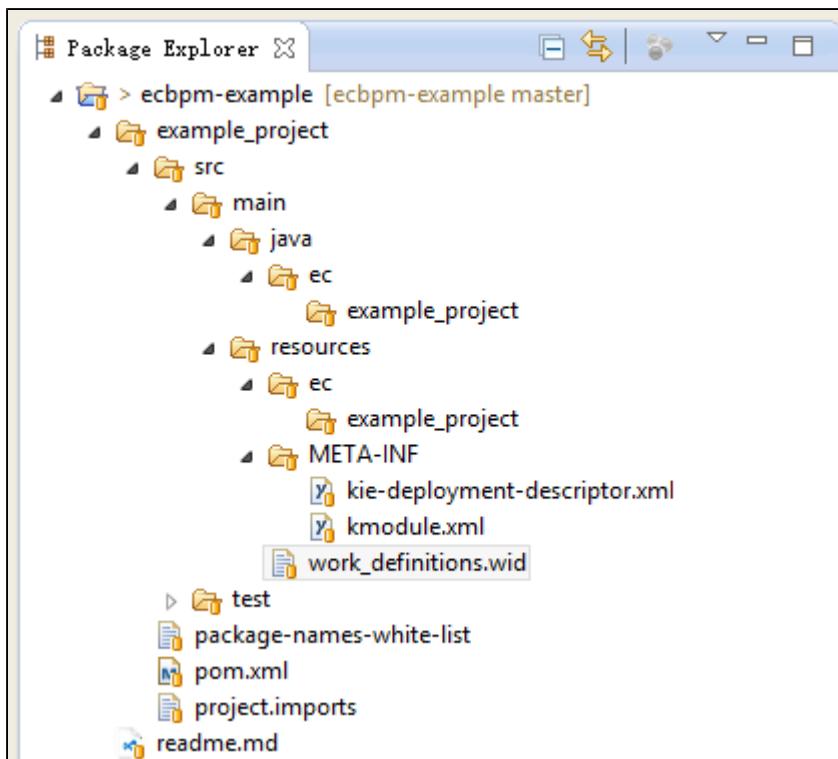


10. Give the project a name. Usually this can leave as it is. Click Finish.



11. Now a new project should be imported. If multiple projects exist in the repository, all of them will appear in the Package Explorer.

new



### **Commit and Push the Changes**

For how to pull, commit and push changes from and to the repository, see the [EGit Tutorial](#).

### **Which IDE to use**

Choosing of IDE is up to projects. Technically, as long as the project can be compiled and deployed to the process engine, the IDE is good to use. However, there are some notes on working with 3rd party IDE:

- Stick to only one IDE. As each IDE has its own way to read and render BPMN file, mix using multiple IDE may cause information loss or even permanently destroy a process file. We already found issues on editing Eclipse generated process files on Workbench.

## **Process Operations**

### **Create a Process**

A process has to be placed inside a project. To create a new project, first go to Project Authoring page via menu Authoring -> Project Authoring.

In Project Explorer, navigate to the project the new process should be created in.

Click menu New Item -> Business Process to bring up the new process dialog

## Create new Business Process

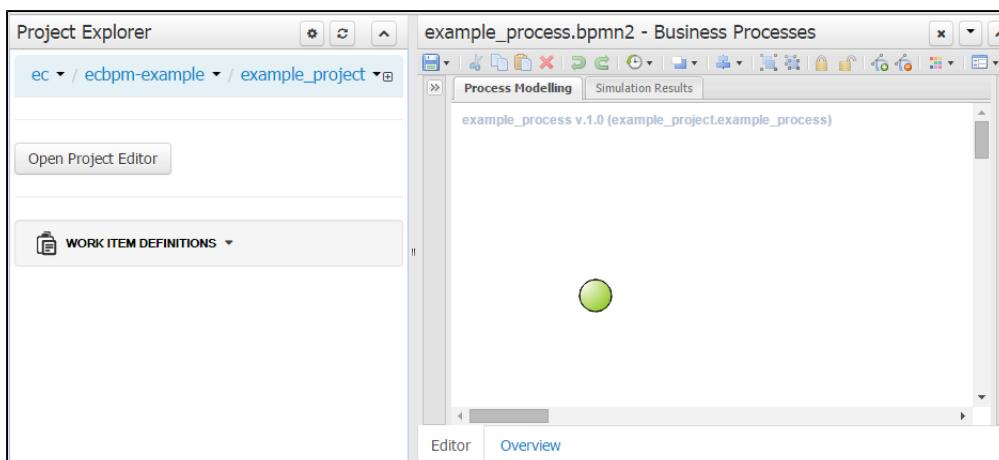
\* Business Process

Package

**Ok** **Cancel**

- Business Process - The name of the new business process
- Package - The package (directory) to place the process.

Click OK, the new process should be created and opened on the right.



Note that the new process does not show up in Project Explorer immediately. To see the new process file, click Refresh



Click the "<<" button on the right to bring up the process property:

new

Properties (BPMN-Diagram)	
Name	Value
<b>Core Properties</b>	
AdHoc	false
Executable	true
<b>Globals</b>	
ID	ec_process_action_sample
Imports	java.lang.Object default
Package	com.ec.bpm.processes.sample
<b>Process Inst...</b>	
Process Name	Simple EC Process Action Invocation
Variable Defi...	_action_state:Integer,_action_result_1:String,_action_r...
Version	1
<b>Extra Properties</b>	
Documentati...	This process shows how to invoke an EC Process Acti...
Target Name...	http://www.omg.org/bpmn20
TypeLanguage	http://www.java.com/javaTypes
<b>Simulation Properties</b>	
Base Currency	
Base time unit	minutes

Fill-in required property for the process:

- AdHoc - false. Currently ecbpm does not support AdHoc processes.
- Executable - true.
- Globals - The constants used by this process. Usually this is left blank.
- ID - The unique process Id in the project.
- Package - The package name of the process. This is not part of the process Id, therefore processes in the same project with same process Id but different package are still considered as a same process, which will fail the compilation.
- Version - This is only for documentation purpose. The jBPM engine now doesn't support process versioning.



After the information is updated, click the Save button on toolbar to save the changes.

## Upload a Process

To upload a process from local drive, first navigate to the project where to place it.

Via menu New Item -> Uploaded File, brings up the Create new Upload File dialog

## Create new Uploaded file

\* Uploaded file

Package  ▾

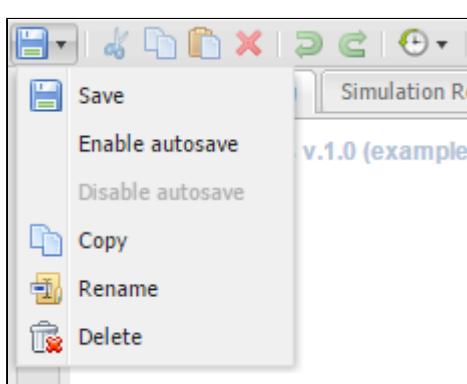
- Uploaded file - The new file name after uploading. The file name should end with .bpmn2 (case sensitive) in order for the designer to recognize it as a process.
- Package - The package (directory) to place the uploaded file.
- Choose File - Choose a local file to upload.

Click Ok to upload the selected file.

### Delete, Rename and Copy a Process

Process deletion, renaming and copying can be done via the process designer.

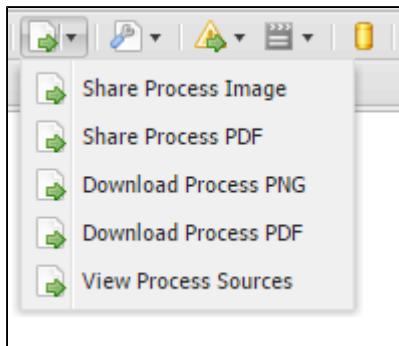
Open the process to operate, and on the designer, click the small triangle beside the save button , which brings up the operation list:



Choose the required operation.

### View Process Source

In the process designer, click the small triangle beside the export button , which brings up the operation list:

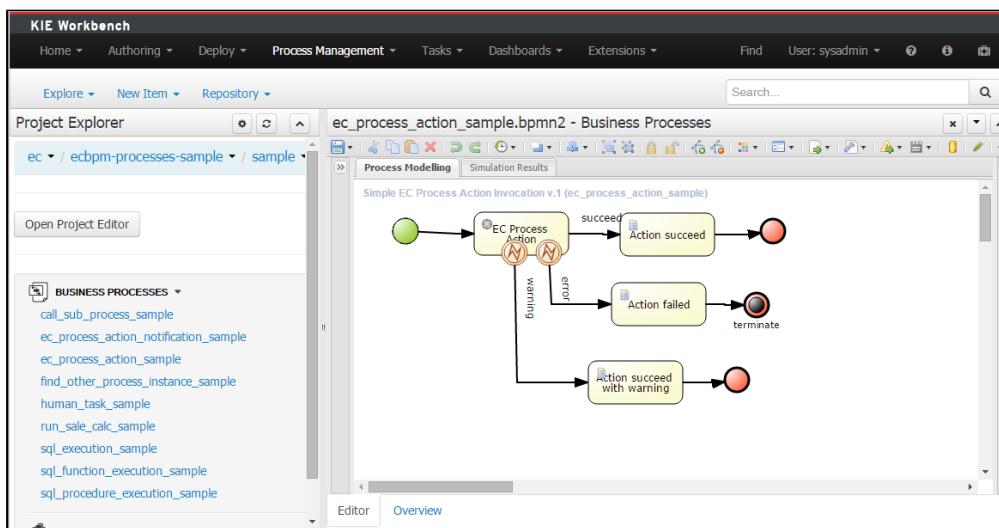


Clicking View Process Sources brings up the source viewer.

## Designing Processes

Process is designed using Business Process Management Notations version 2.0. The KIE Workbench provides most of the notations supported by the standard.

To open a process in designer, first go to Process Authoring page via menu Authoring -> Project Authoring. Navigate the Project Explorer to the project containing the process, and click the process to edit.

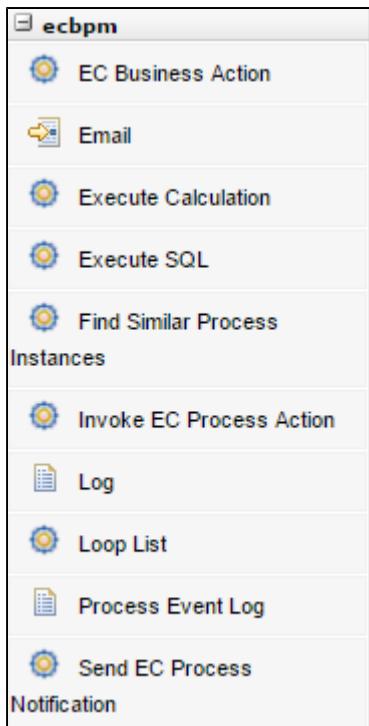


## The Object Library

The Object Library contains all notations available to be used in process. Click the ">>" button  on the left to bring the Object Library.



All ecbpm notations are placed inside the "ecbpm" folder. For instructions on their usage, refer to corresponding section.



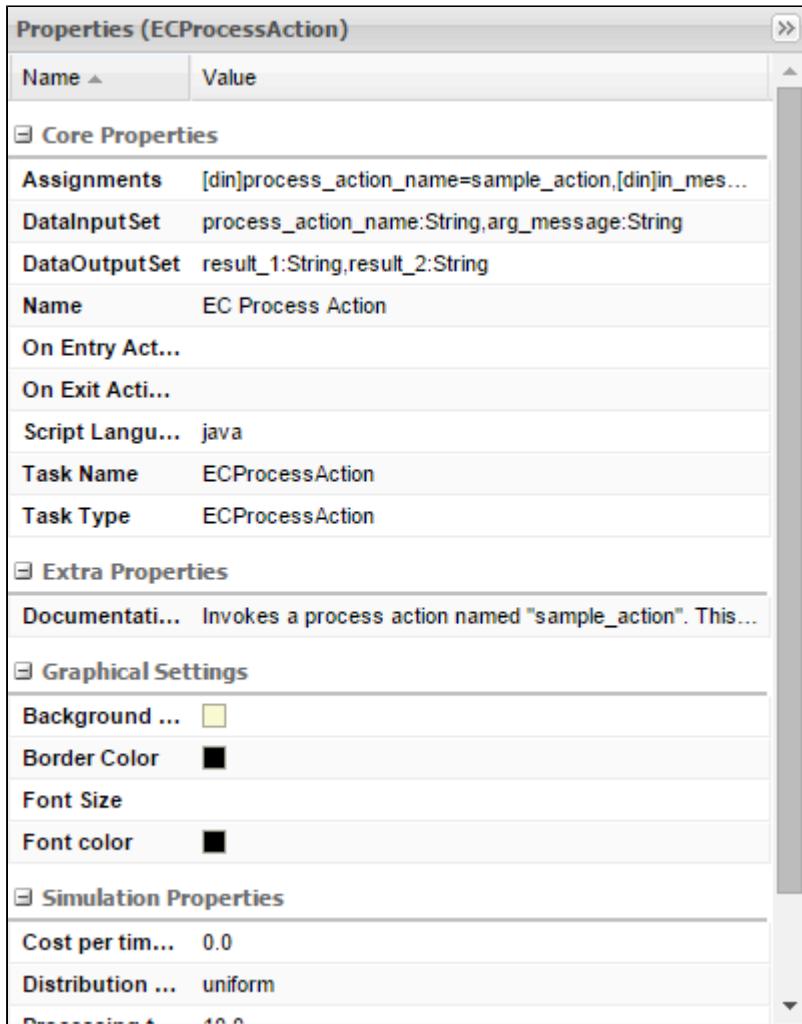
For other standard BPMN2 notations usages, refer to one of the following documentation:

- BPMN 2.0 Tutorial from Camunda - <https://camunda.org/bpmn/tutorial/>
- BPMN 2.0 References - <http://docs.camunda.org/7.3/api-references/bpmn20/>
- Book «Real-Life BPMN: Using BPMN 2.0 to Analyze, Improve, and Automate Processes in Your Company» - <http://www.amazon.com/Real-Life-BPMN-Analyze-Automate-Processes/dp/1502972328>
- BPMN notation poster - [http://www.bpmn.de/images/BPMN2\\_0\\_Poster\\_EN.pdf](http://www.bpmn.de/images/BPMN2_0_Poster_EN.pdf)

### The Properties Panel

The properties panel lists property of selected node. When no node is selected, it lists property of the process.

To bring up the Property Panel, click the "<<" button  on the right.



The screenshot shows the 'Properties (ECProcessAction)' dialog box. It contains several sections:

- Core Properties** section:
  - Assignments: [din]process\_action\_name=sample\_action,[din]in\_mes...
  - DataInputSet: process\_action\_name:String,arg\_message:String
  - DataOutputSet: result\_1:String,result\_2:String
  - Name: EC Process Action
  - On Entry Act... (disabled)
  - On Exit Act... (disabled)
  - Script Langu...: java
  - Task Name: ECProcessAction
  - Task Type: ECProcessAction
- Extra Properties** section:
  - Documentation: Invokes a process action named "sample\_action". This...
- Graphical Settings** section:
  - Background ...: yellow square
  - Border Color: black square
  - Font Size: (disabled)
  - Font color: black square
- Simulation Properties** section:
  - Cost per tim...: 0.0
  - Distribution ...: uniform

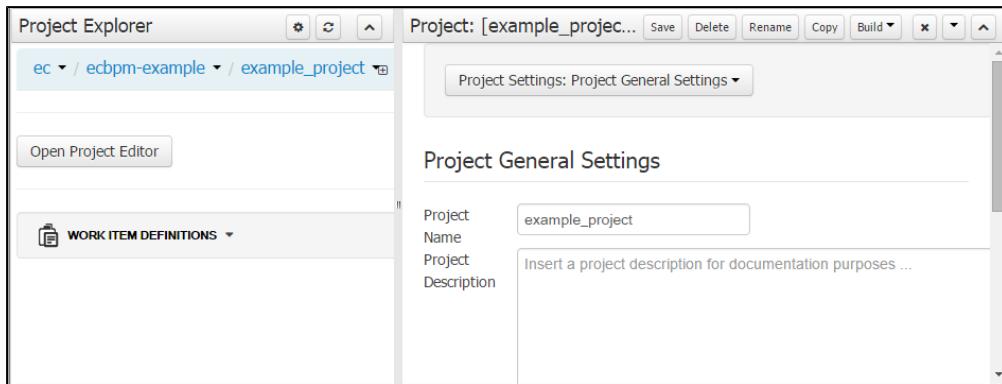
## Deploy Processes to Process Engine

### Compile and Deploy

Processes have to be compiled and deployed to process engine before they are accessible from EC. Processes are compiled and deployed together with its project. To compile and deploy a project, first go to the authoring page via menu Authoring -> Project Authoring.

Navigate the Project Explorer to the project.

Click button "Open Project Editor" to bring up the editor.



Click Build button to bring up the drop-down, the click Build & Deploy.

If deployment is successful, a green bar will show up on top. Otherwise, a red bar will appear. See the next section for troubleshooting.

### Viewing Deployed Projects

Via menu Deploy -> Process Deployments, brings the Deployed Units page. This page lists all projects currently being deployed in the process engine.

Deployed Units			
Deployment	Runtime strategy	Status	Actions
org.guvnor:guvnor-asset-mgmt-pr...	SINGLETON	Active	⊕ Ø
com.ec.frmw.bpm:ecbpm-sample:...	PER_REQUEST	Active	⊕ Ø

To find the project being deployed, look at the Deployment column. The ID in Deployment is made of three parts:

Group Id, Artifact Id, Version

For example, ID "com.ec.frmw.bpm:ecbpm-sample:11.1.RC4-SNAPSHOT" gives

- Group Id: com.ec.frmw.bpm
- Artifact Id: ecbpm-sample
- Version: 11.1.RC4-SNAPSHOT

Match these three parts with the project information, will find which project the deployment unit was built from.

### Compilation Troubleshooting

Due to an incompatible version component used in Wildfly, the Error Panel is not shown in KIE Workbench, which makes it tricky to find errors during compilation and deployment. However, error logs are still available in server log. One quick way to find compilation error in server log is to search for key word "compile".

Below lists some common errors happen in Compile and Deploy, they can be used as a check list before analyzing the server log.

- Active process instances found - Verify that all process instances are either completed or aborted from page Process Management -> Process Instances (no items list under Active tab)
- The project is already deployed - jBPM 6.2.0 by default does not support auto-undeployment. Therefore, if the project is already deployed, it needs to be undeployed. See section Project Undeployment for how to undeploy a project.
- No reusage of Process Id - Verify that all processes in the project have unique Id.
- No missing Process Id

## Project Undeployment

Via menu Deploy -> Process Deployments, brings the Deployed Units page. Find the deployment that needs to be undeployed, and click the icon .

Make sure there is no active process instance belong to the undeploying deployment, or the undeployment will fail.

## Viewing Deployed Processes

Screen jBPM Console -> Process Management -> Process Definitions lists all deployed processes. Clicking on one process shows its detailed information. Following items are used by EC to identity a process:

- Definition Id: the id of the process
- Deployment: name of the artifact that the process is defined in

## Process Execution and Management

Both EC and jBPM Console supports executing process instances. However, it is suggested to *always* execute processes from EC, as there will be additional information send from EC for access control. When executed from jBPM Console, it is not guaranteed the process instance will run as expected.

### Process Templates

A Process Template is an EC alias of a deployed process in process engine. Multiple Process Templates can point to the same process.

Process Template is the only way in EC to start a process instance. Therefore, to be able to execute a deployed process, it has to be registered in EC as a Process Template.

Process Templates by nature are also registered business actions. Created Process Templates can be found in EC's Business Actions screen. They can be scheduled using EC scheduler to be executed periodically.

### Creating Process Templates

To create Process Templates, go to screen EC -> Process Automation -> Process Template.

Process Template					Process Template	
Process Template	Deployment Id	Process Id	Functional Area			
Process Action Notification Sample	com.ec.frmv.bpm:ecbpms-sample:11.1.F	ec_process_action_notification_sample	EC			
Run Sale Calculation Sample	com.ec.frmv.bpm:ecbpms-sample:11.1.F	run_sale_calc_sample	EC			
SQL Execution Sample	com.ec.frmv.bpm:ecbpms-sample:11.1.F	sql_execution_sample	EC			
SQL Function Execution Sample	com.ec.frmv.bpm:ecbpms-sample:11.1.F	sql_function_execution_sample	EC			
To-do List Sample	com.ec.frmv.bpm:ecbpms-sample:11.1.F	human_task_sample	EC			

PARAMETERS					STATIC PARAMETERS	
Name	Type	Sub Type	Mandatory	Description	Name	Static Value
SIMULATE	Basic Type	String	<input type="checkbox"/>	<input type="text"/>	No records found.	
SKIP_CALC	Basic Type	String	<input checked="" type="checkbox"/>	<input type="text"/>		
TRANSACTIONMO	Basic Type	String	<input type="checkbox"/>	<input type="text"/>		

Click the Insert button  and select "Process Template" for a new record. A new row should appear in the list, fill-in the fields as described below:

Property Name	Description
Process Template	The display name of a process template.
Deployment Id	The Id of the deployed artifact that contains the process.
Process Id	The id of the process. This drop-down lists all deployed processes from the selected deployment (as in Deployment Id).
Functional Area	The functional area this process template belongs to. This can be used to restrict user access to a group of process templates.

### Parameters

To input values to process instance, corresponding process variable needs to be defined as Process Template Parameter. When executing a Process Template, the user will have a chance to provide values to Process Template Parameters, the parameter values are then sent to the new process instance variable with the same name (case-sensitive).



#### Caution

Note that it is not allowed to add undefined process variables to Parameters list, as it may cause process instances to fail during execution.

Click the Insert button  and select "Parameters" for a new record. A new row should appear in the parameter list, fill-in the fields as described below:

Property Name	Description
Name	Name of a process template parameter. This has to be the same (case-sensitive) as defined in process.
Type	Data type of a parameter.
Sub Type	Sub data type of a parameter.
Mandatory	Indicates whether a parameter is mandatory.
Description	Description of a parameter.

### Static Parameters

Static Parameters are Parameters, with exception that their values are provided on Process Template and can not be changed for each execution.



#### Caution

Note that it is not allowed to add undefined process variables to Static Parameters list, as it may cause process instances to fail during execution.

Click the Insert button  and select "Static Parameters" for a new record. A new row should appear in the static parameter list, fill-in the fields as described below:

Property Name	Description
Name	Name of a process template parameter. This has to be the same (case-sensitive) as defined in process.
Static Value	Value of a parameter.

### Before Process Template Execution

ecbpm uses EC schedule to provide internal services. Following schedules are required to be enabled based on processes to be executed.

#### Schedule "BpmSchedulerEnv"

Schedule "BpmSchedulerEnv" enables executing business actions via EC Scheduler. This is mandatory to enable this schedule before executing a process that uses either Business Action or Process Action functionality. It is recommended to always have this schedule enabled before running any processes.

This schedule is invoked on demand by process engine. When enabling this schedule, use following attributes:

Attribute	Value	Description
Schedule Type	Once	As this schedule is ran on demand, it doesn't need to be configured as auto-triggered.
Stateful	false	This schedule should not be stateful, as it should support multiple instances.
Pin To		Pin this schedule to a server for bpm related business action execution. This could be a dedicated server for load balancing.

#### Schedule "BpmEventInboundWatcher"

Schedule "BpmEventInboundWatcher" is responsible for checking EC side BPM inbound events. This schedule is not required to be enabled as long as EC Dataset tracing feature is not used in any process.

Based on the frequency events popup into the inbound, this schedule can be configured to run every minute, 5 minute, or in seconds. Configure the Schedule Type as "Cron" and set the Cron Expression to reflect the expected frequency.

## Execute a Process Template

Process Templates can be either executed manually or by the scheduler.

### Manual Execution

To manually execute a Process Template, go to EC screen Process Automation -> Process Execution.

Fill in the navigator fields:

Field Name	Description
Date	No actual use for now. This field comes alone with the Functional Area group model and is not used by other components, therefore the default value can be used.
Functional Area	The functional area the Process Template belongs to.
Process Template	The Process Template to execute.

Click Go to load the screen.

The screenshot shows the 'Process Execution' screen with the following details:

- Header:** Includes icons for download, refresh, add, delete, and search, followed by a star icon.
- Filter:** Fields for Date (set to 2015-12-03), Functional Area (set to EC), and Process Template (set to Run Sale Calculation). A green arrow button is to the right of the Process Template field.
- Parameters:** A table showing registered parameters:
 

Name	Value	Description
SIMULATE		
SKIP_CALC	N	
TRANSACTIONMODE		
- Buttons:** 'START PROCESS' (green button) and 'REFRESH' (green button).
- Execution History:** A section with two tabs: 'TRIGGER HISTORY (TODAY)' and 'PROCESS INSTANCES (TODAY)'. The 'TRIGGER HISTORY (TODAY)' tab is selected, showing the message 'No records found.'

The screen shows Parameters registered on Process Template, as well as Execution History section.

The Execution History section contains two tabs:

Tab	Description												
Trigger History (Today)	<p><b>TRIGGER HISTORY (TODAY)</b> <b>PROCESS INSTANCES (TODAY)</b></p> <table border="1"> <thead> <tr> <th>Name</th> <th>Status</th> <th>Schedule Time</th> <th>Start Time</th> <th>Duration</th> <th>Node</th> </tr> </thead> <tbody> <tr> <td>Run Sale Calculation</td> <td>OK</td> <td>2015-12-04 12:40</td> <td>2015-12-04 1:00:00.11</td> <td></td> <td>WL2006560</td> </tr> </tbody> </table> <p><b>REFRESH</b> <b>GET DETAILED LOG</b></p> <p>Shows today's trigger history. A trigger is a business action carried out by EC Scheduler to make sure a process instance is created on process engine. After process instance is created, the trigger is completed with status OK. Triggers do not monitor process instance status and always return immediately when a process instance is created on process engine.</p> <p>To view trigger log, select a trigger history item and click Get Detailed Log button.</p>	Name	Status	Schedule Time	Start Time	Duration	Node	Run Sale Calculation	OK	2015-12-04 12:40	2015-12-04 1:00:00.11		WL2006560
Name	Status	Schedule Time	Start Time	Duration	Node								
Run Sale Calculation	OK	2015-12-04 12:40	2015-12-04 1:00:00.11		WL2006560								
Process Instances (Today)	<p><b>TRIGGER HISTORY (TODAY)</b> <b>PROCESS INSTANCES (TODAY)</b></p> <table border="1"> <thead> <tr> <th>ID</th> <th>State</th> <th>Process Template</th> <th>Process Definition</th> <th>Start</th> <th>End</th> </tr> </thead> <tbody> <tr> <td>383</td> <td>Completed</td> <td>Run Sale Calculation Sample</td> <td>Run Sale Calculation Sample</td> <td>2015-12-04 11:40</td> <td>2015-12-04 11:40</td> </tr> </tbody> </table> <p><b>REFRESH</b></p> <p>Shows process instances executed today.</p>	ID	State	Process Template	Process Definition	Start	End	383	Completed	Run Sale Calculation Sample	Run Sale Calculation Sample	2015-12-04 11:40	2015-12-04 11:40
ID	State	Process Template	Process Definition	Start	End								
383	Completed	Run Sale Calculation Sample	Run Sale Calculation Sample	2015-12-04 11:40	2015-12-04 11:40								

To execute a process template, first fill in parameter values, at least mandatory ones, then click Start Process button. As process execution is ran asynchronously, the history area may not show new items immediately. Click Refresh to refresh the data for new items.

### By Scheduler (Automatically and Periodically)

To configure a Process Template to be executed automatically by the scheduler, go to EC screen Configuration -> Scheduler -> Schedules.

Click the Insert button  and select "Schedule" for a new record. A new row should appear in the schedule list, fill-in the fields as described below:

Property Name	Description
Name	Display name of a schedule.
Description	Description of a schedule.
Functional Area	The functional area this schedule belongs to. This can be used to restrict user access to a group of process templates.

Save the changes. Select the newly created schedule (if not selected by default).

Switch to Business Action Tab.

Click the Insert button and select "Business Action" for a new record. A new row should appear in the Business Action list, fill-in the fields as described below:

Property Name	Description
Action	In the drop-down, it lists all registered Business Actions. Find the Process Template name in the list.
Description	Description of the Process Template.
Sequence#	The order of this Process Template to execute when multiple Process Templates are added to one schedule. If only one template is added, give 0.
Isolate Trans	This will not affect the execution of a Process Template, as the transaction boundary is maintained by the ecbpm server rather EC.

Screen EC -> Process Automation -> Process Execution provides way to create new process instances manually. To configure scheduled process execution, use screen EC -> Configuration -> Scheduler -> Schedules.

So far, the schedule is configured. To finalize the set up, complete the configuration in Schedule tab, and enable the schedule in Details tab as normal schedules. For more information, refer to EC User Manual.

## Managing Process Instances

### Viewing Process Instances

Process instances can be managed via EC screen Process Automation -> Process Overview.

Navigator provides common filters for process instances, as described below:

Field Name	Description
From date	Start of process instance daytime range (including the date itself)
To date	End of process instance daytime range (excluding the date itself)
Date Param	The process instance daytime parameter/variable. This variable indicates where to pull process instance daytime from. If leaving blank, process instance start time will be used.
Functional Area	Functional area of the process templates process instances was created from.
Process Template	The process template from which process instances was created. If leaving blank, all process instances falls into the selected Functional Area will be queried.

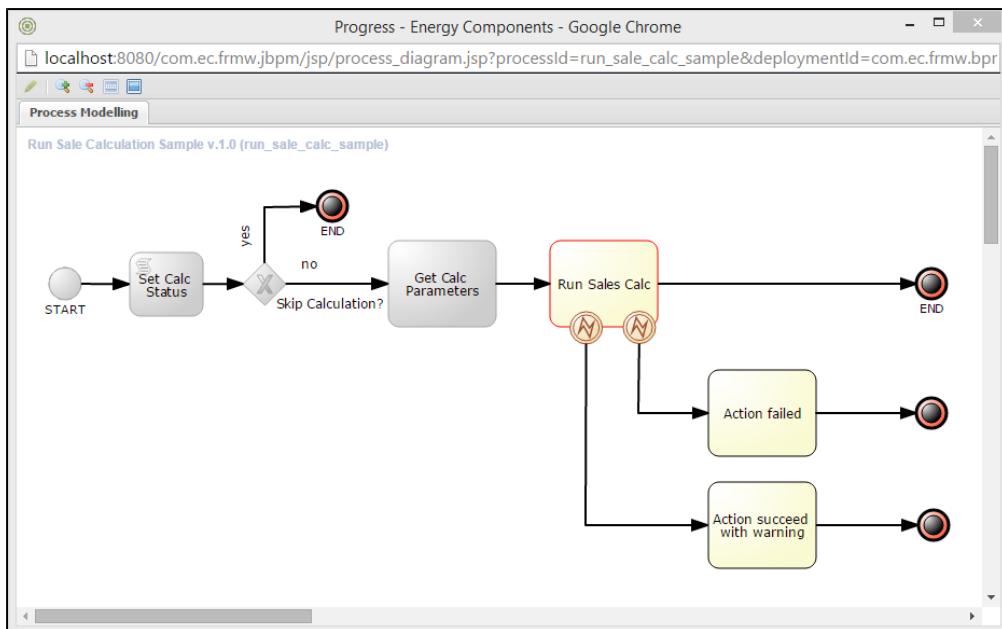
Click Go, the screen will load with queried process instances.

The screenshot shows the 'Process Overview' interface. At the top, there are filters for 'From date' (2015-12-01), 'To date' (2015-12-05), 'Functional Area' (EC), 'Process Template' (Run Sale Calculation), and 'Date param'. Below the filters is a table titled 'PROCESS INSTANCES' with columns: Id, State, Process Template, Process Definition, Start, and End. The table lists 36 process instances, all in 'Completed' state, with various start and end times. To the right of the table is a 'LOGS' section showing a timeline of events with columns: Time, Node, and Message. The log entries show the progression of activities like 'Run Sales Calc' and 'Get Calc Parameters' for each instance. At the bottom of the interface are buttons for 'VIEW DIAGRAM', 'TOGGLE SELECTION', 'ABORT', and 'DELETE'.

### Viewing Process Instance Status

Refer to section "Viewing Process Instances" for how to query process instances.

Select a process instance from the list, and click button View Diagram. A new browser window will show up with process diagram and colored nodes indicating current process instance status.



Nodes are colored according to current execution status,

- Grayed - Executed/completed nodes
- Red-bordered - Executing nodes
- Black-bordered - nodes have not been executed

To refresh the diagram, press F5 or other browser quick key for refreshing web pages.

### Aborting Process Instances

Refer to section "Viewing Process Instances" for how to query process instances.

A process instance must be in Active state to be aborted.

Process abortion supports bulk operation. First, tick the check-boxes on process instances to be aborted, then click Abort button. The screen will reload with new process instance status. Aborted process instances will have a new status called "Aborted".

## Deleting Process Instances

Refer to section "Viewing Process Instances" for how to query process instances.



### Caution

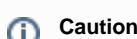
Process Instance Deletion is not recoverable, all data, including Process Event Logs, jBPM internal process instance and node logs, are deleted permanently.

A process instance must be in Completed or Aborted state to be deleted.

Process deletion supports bulk operation. First, tick the check-boxes on process instances to be deleted, then click Delete button. The screen will reload with new process instance status. Aborted process instances will have a new status called "Aborted".

## Automatically Deleting Process Instances (Process Instance Clean Up)

ecbpm supports automatically process instance clean up via EC Scheduler. Pre-defined schedule "BpmProcessInstanceCleanUp" can be configured for this purpose.



### Caution

Process Instance Deletion is not recoverable, all data, including Process Event Logs, jBPM internal process instance and node logs, are deleted permanently.

Schedule "BpmProcessInstanceCleanUp" is based on business action "RunAppBusinessAction - ProcessInstanceCleanUpAction". If more than one clean up schedule is required (in cases like deleting instances from process 1 each 5 min, while deleting instances from process 5 every 10 min), new schedule can be created using the same business action.

Schedule "BpmProcessInstanceCleanUp" supports deleting instances from a certain process, from all processes of a given deployment, or from all processes. The schedule also supports deleting process instances by age (based on daytime, i.e., instance start time or a specified date parameter).

The schedule only deletes process instances of state Completed and Aborted, active ones are not affected.

To configure the schedule, go to EC screen Configuration -> Scheduler -> Schedules, select schedule "BpmProcessInstanceCleanUp", and switch to "Business Action" tab.

The screenshot shows the EC Scheduler configuration interface. The top navigation bar has tabs for SCHEDULE, BUSINESS ACTION, SCHEDULE, and MONITOR. The BUSINESS ACTION tab is selected. The main area displays the configuration for the "BpmProcessInstanceCleanUp" schedule. It includes sections for BUSINESS ACTION, PARAMETERS, MACRO, and FIXED PARAMETERS. The BUSINESS ACTION section lists the action "BpmProcessInstance" with description "Deletes ecbpm process instances" and sequence number 1. The PARAMETERS section contains fields for "deployment\_id", "process\_daytime\_since", "process\_daytime\_var", and "process\_id". The MACRO section shows "No records found". The FIXED PARAMETERS section shows "Sclass\_name\$ com.ec.bpm.ext.ec.actions.ProcessInstanceCleanUp". A green "RUN NOW" button is visible at the top right of the SCHEDULE tab.

Fill-in parameters as described below:

Parameter	Description
deployment_id	Indicates which deployment the deleting process instance should belong to. This parameter is used together with process_id, when either one is left blank, all process instances are queried.
process_id	Indicates which process the deleting process instance should belong to. This parameter is used together with deployment_id, when either one is left blank, all process instances are queried.
process_daytime_since	Indicates the youngest age of process instances to be deleted. When leaving blank, process instance daytime will not be considered on filtering.
process_daytime_var	Indicates the name of process instance daytime variable. When leaving blank, process instance start time will be used as process instance daytime.

To filter process instances with additional process variable values, add new parameters with name format "process\_param\_{process\_var\_name}".

Following are some examples on parameter configuration:

	deployment_id	process_id	process_daytime_since	process_daytime_var
Deletes process instances started before 5 min ago			5	
Deletes process instances of deployment_a/process_a, and started before 5 min ago	deployment_a	process_a	5	
Deletes process instances of deployment_a/process_a	deployment_a	process_a		
Deletes process instances of last month's processing period			21600	processing_period
Deletes all process instances				

## Business Action Invocation

ecbpm supports invoking EC Business Actions from processes. Business Actions are ran on EC by scheduler in asynchronous manner. Process instance waits for business action completion and continues.

### Use in Process

#### Input and Output Variables

Business Action invocation activity is implemented as a BPMN Service Task with name "BusinessActionCall".

The activity has following pre-defined input parameters:

Name	Type	Object Type	Optional	Description
ActionClass	Input	String	No	The full name of the business action class to invoke.

To send arguments to a business action, new input parameters need to be added. The new parameters should follow following rules:

1. Parameter name should follow the format "arg\_" + argument name. For example, to assign value to a business action argument "CONTRACT\_ID", the input parameter should be named as "arg\_CONTRACT\_ID"
2. The parameter name is case sensitive. The prefix "arg\_" needs to be in small cases. Business action argument name needs to follow its own definition.

To receive results sent from business actions, following output parameters can be added:

Name	Type	Object Type	Optional	Description
ec.extension.ba.text	Output	String	Yes	The text returned from business action
ec.extension.ba.status	Output	String	Yes	The status returned from business action
ActionResult	Output	com.ec.frmw.jbpm.services.action.ActionResult	Yes	<b>(Obsoleted)</b> Holds the result object returned from business action.

To receive additional result returned from the business action (via method UserEventResponse.setAttribute()), add an output variable with the attribute name (case-sensitive).

#### Error Handling

ecbpm interpreters business action's non-successful return state (status > 0, i.e. WARNING and ERROR) as boundary error signals. When status WARNING or ERROR is returned from a business action, the corresponding signal is thrown at the activity. The two signals are defined as below:

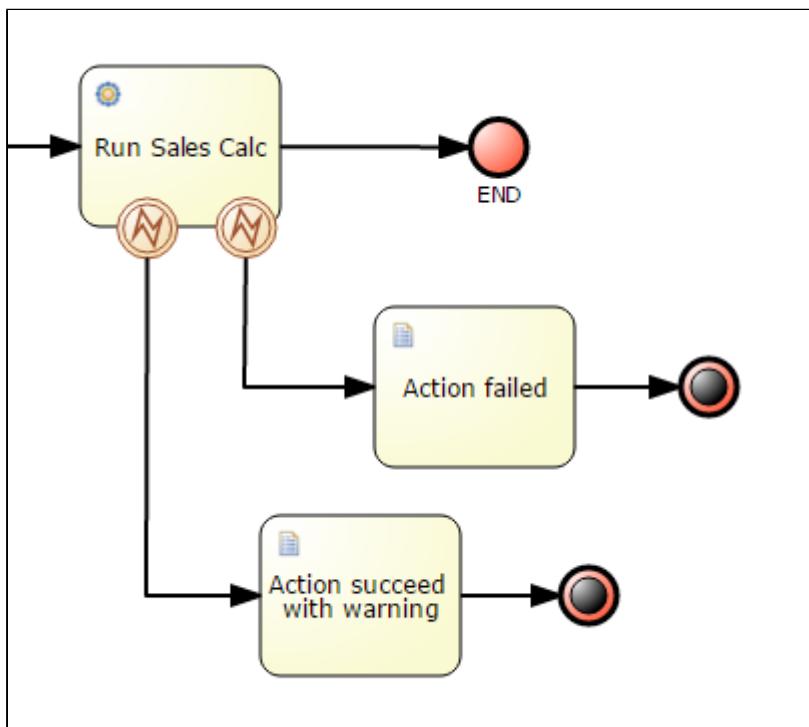
Signal Name	Signal/Event Type	Description	Event Object Content
ecbpm_action_error	Error	Thrown when business action throws exception or returns state "ERROR"	A map containing either the exception doc content, or business action activity output variables, for example "ec.extension.ba.text" and other user event response attributes. To access, first map the object to a process variable, then use MVEL expression like #{error_obj["ec.extension.ba.text"]} to access the value.
ecbpm_action_warning	Error	Thrown when business action returns state "WARNING"	A map containing business action activity output variables, for example "ec.extension.ba.text" and other user event response attributes. To access, first map the object to a process variable, then use MVEL expression like #{error_obj["ec.extension.ba.text"]} to access the value.

Error type catching intermediate events should be attached to the activity in order to capture the signals.



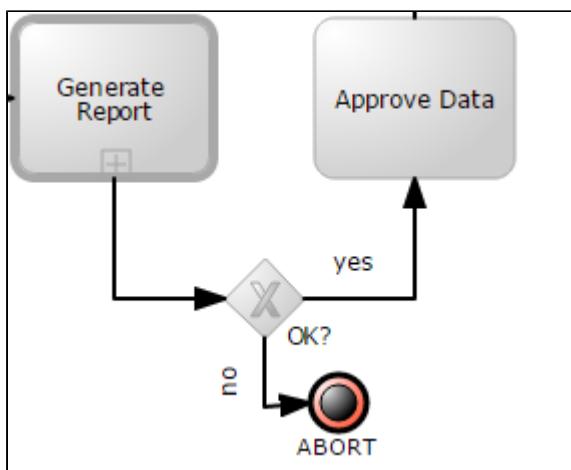
#### Caution

When a business action errors or returns with warning, the process instance will not progress unless the two signals are handled. To fallback to the old way of handling business action status (with a gateway, which is obsoleted), see the following section.



## EC 11.0 Mode

On EC 11.0, business action status are handled by a Gateway, like below

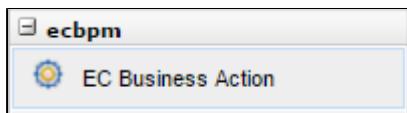


By default, this way of error handling is not supported on EC 11.1. Process instance will hang if events are not handled. However, it is still possible to configure ecbpm to use the old error handling mode. To achieve that, system property (in domain.xml) ecbpm.ext.jbpmcons.process\_action.ignore\_ba\_state needs to be sat to "true".

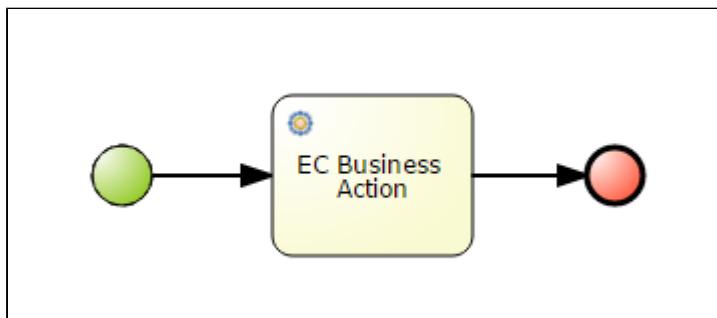
## Example

### Adding the Activity

The activity is available in Object Library of KIE Workbench Designer under category "ecbpm".



To add it to the process, drag it onto the canvas, and add proper connection to preceding and succeeding nodes.



To change the node name, double click on the node and update the name.

### Configuring the Activity

Before configuring the activity variables, one needs to know business action class name and what parameters the business action requires. To get this, either refer to EC Documentation, or the Business Action source code.

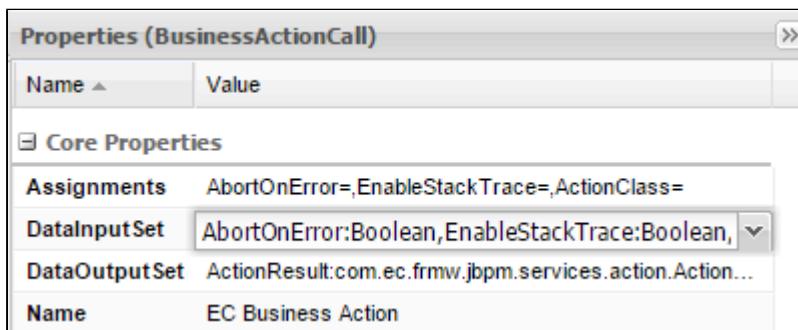
In this example, business action "com.ec.SampleBusinessAction" is used with following specification:

- Class Name: com.ec.SampleBusinessAction
- Input Parameters (*via UserEvent.getParameters()*): PARAM\_1, PARAM\_2
- Output Parameters (*via UserEventResponse.setAttribute()*): RESULT\_1, RESULT\_2
- Business action text is returned

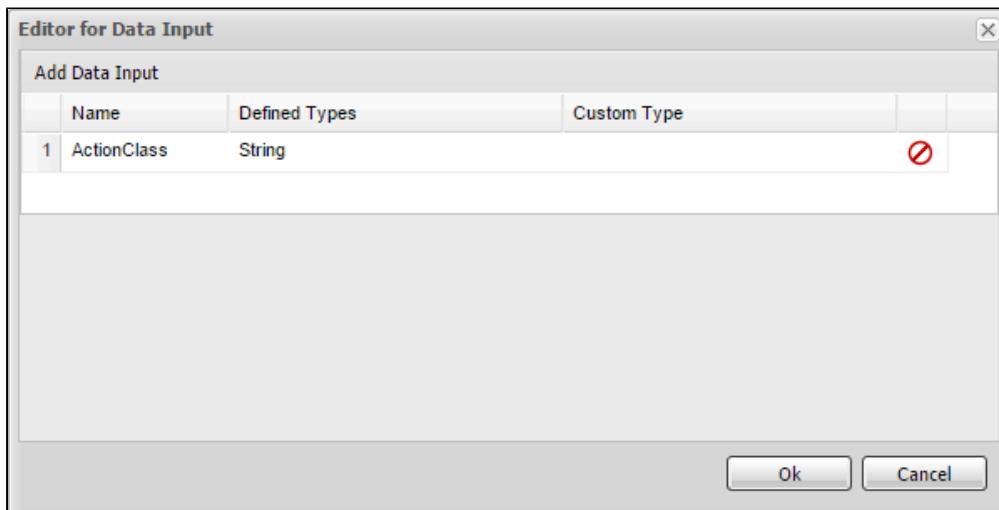
### ***Input Parameters Registration***

All required business action input parameters need to be registered in the Data Input Set. This can be done via the

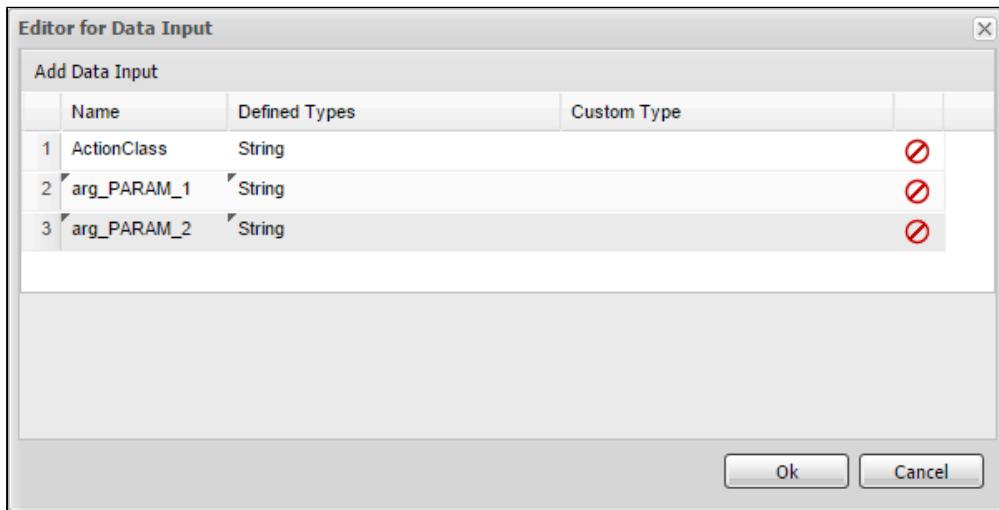
Property Panel.



Click the down arrow button to bring up the configuration dialog:

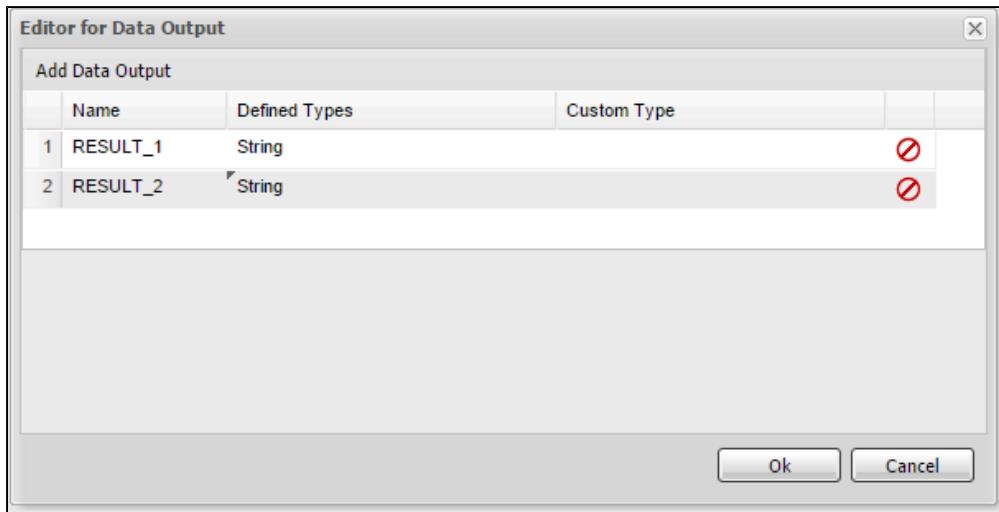


Click button Add Data Input to add parameters for business action. Note that all parameter should be named using format "arg\_{param\_name}".



### ***Output Parameters Registration***

To receive output attributes, register them in the Data Output Set. The name should be the same as provided by the business action (case sensitive). This can be done via the Property Panel.

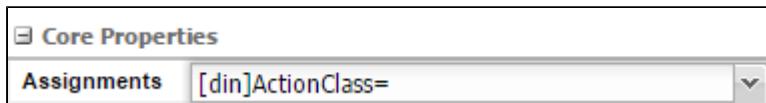


**Caution**

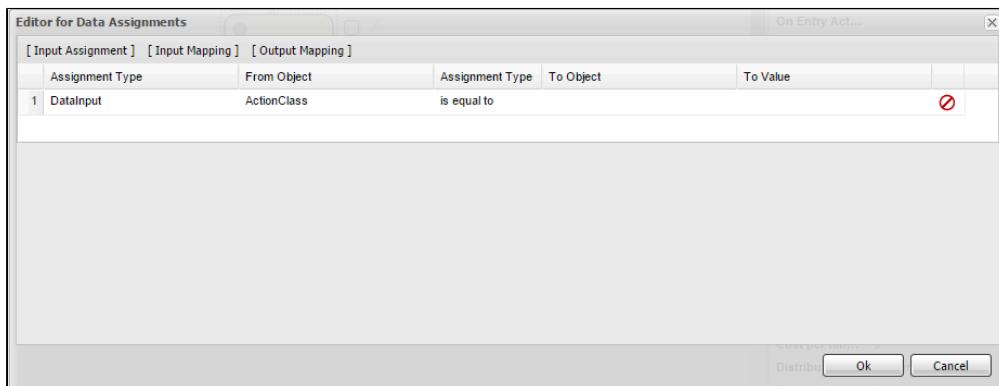
Parameter data type should be the same as provided in the Business Action. Usually this is String. However, if you are unsure about the actual type, or the data type is dynamic during run-time, leave Defined Types empty, and put "java.lang.Object" in Custom Type.

### Assign Input Parameter Values

To assign values to input parameters, open the Properties panel, and click the triangle button on Assignments row:



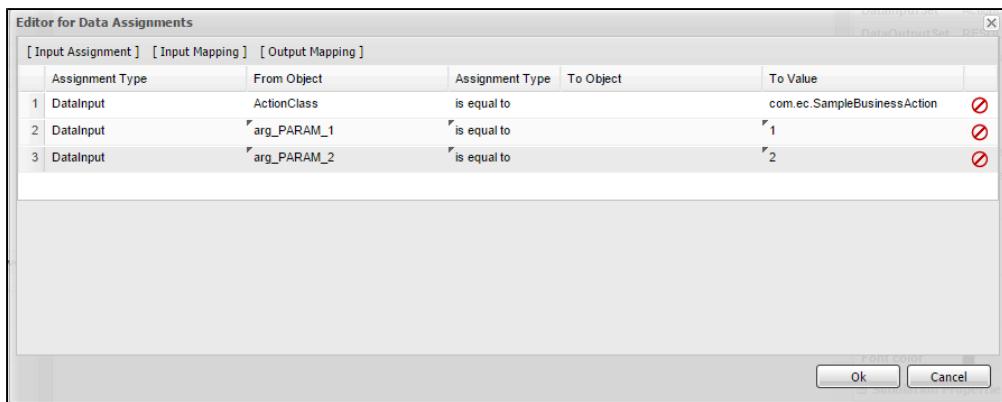
This should show the Assignments dialog:



There are two ways to do input assignment:

1. Input Assignment - Manually input a static value to the variable.
2. Input Mapping - Copy value from a process variable to the input parameter.

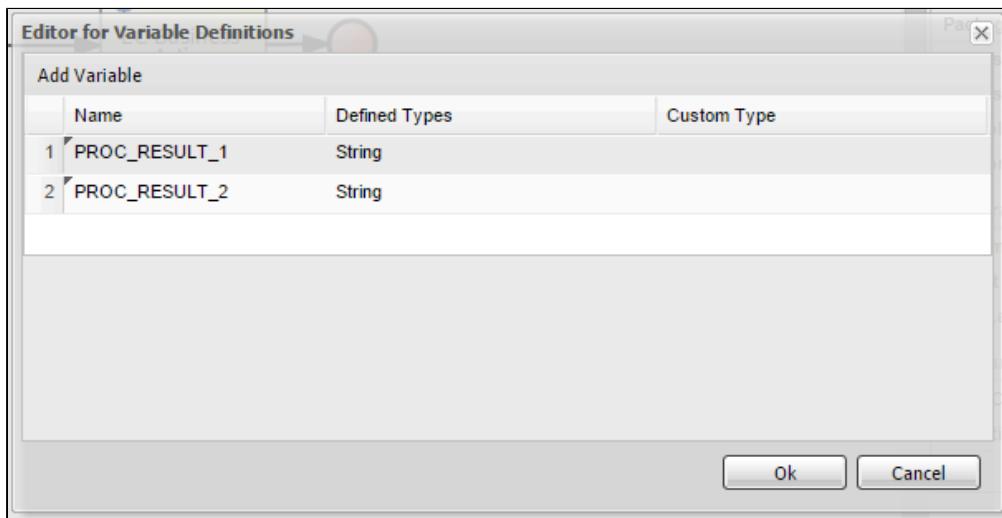
In this example, all assignments are Input Assignments, with following data:



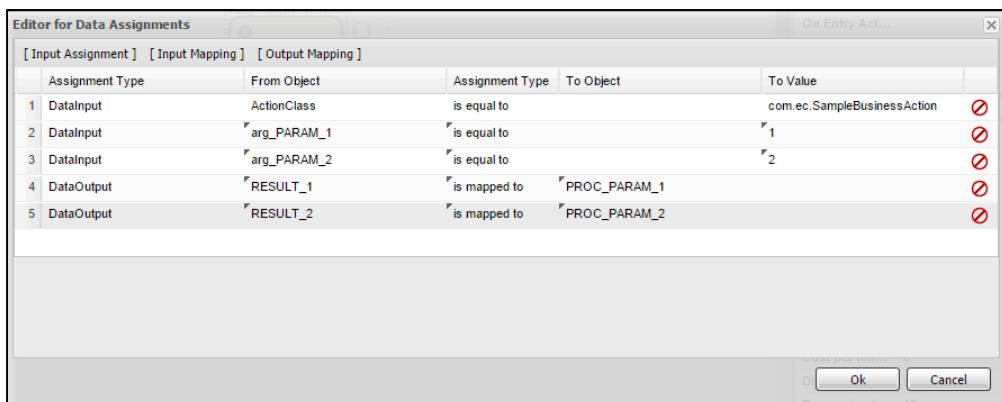
### Assign Output Parameter Values

To store the output parameter values to a process variable, assignments need to be made.

First, define a process variable for storing the parameters. Click on blank canvas, and on Properties panel, open the Variable Definition dialog (by clicking the drop-down arrow on that property row):



Select the Business Action node again, and open the Assignments dialog. Create two new Output Mapping as below:

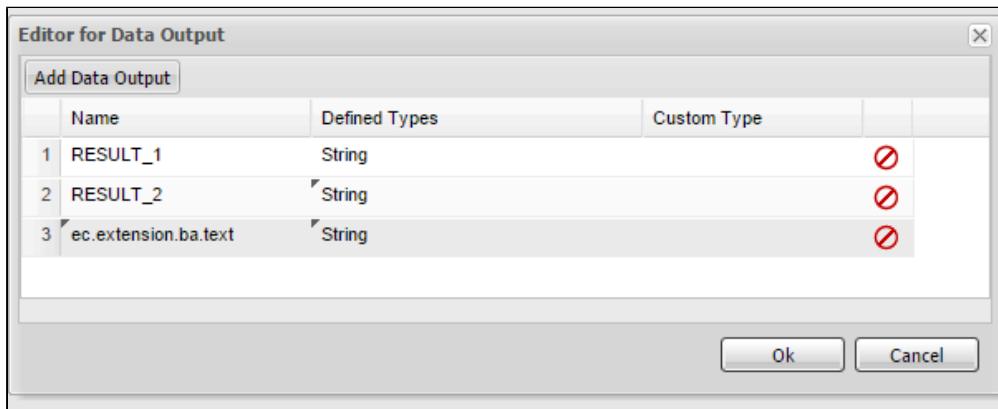


Now the two output variables are mapped to process variables.

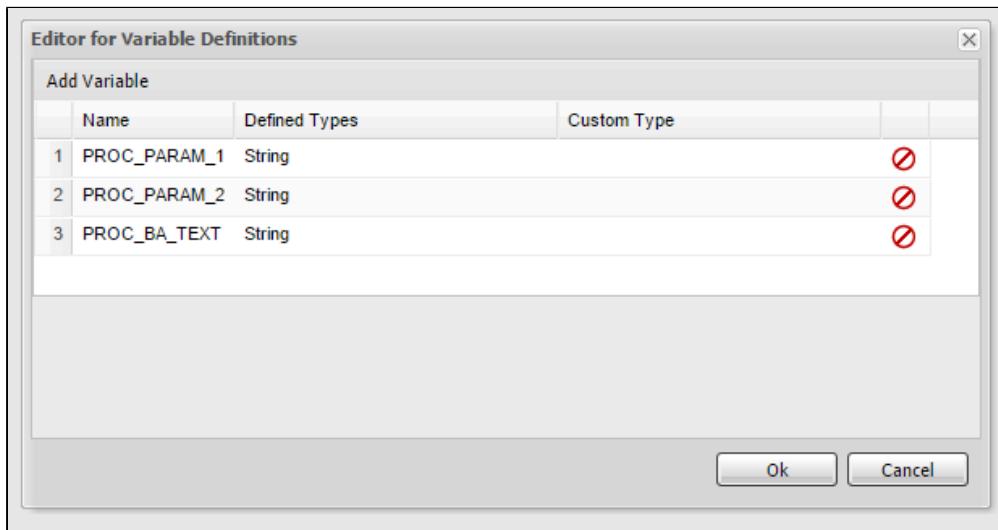
### Map Business Action Return Text

ecbpm supports store business action return text to process variable. To do this, a Output Mapping for node output variable "ec.extension.ba.text" needs to be created.

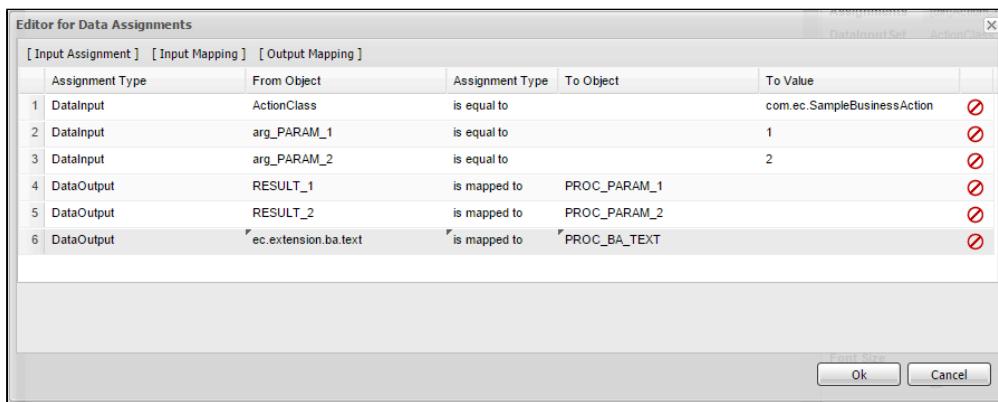
First, register it in the Data Output Set (see instruction above):



Then register a new process variable (see instruction above):

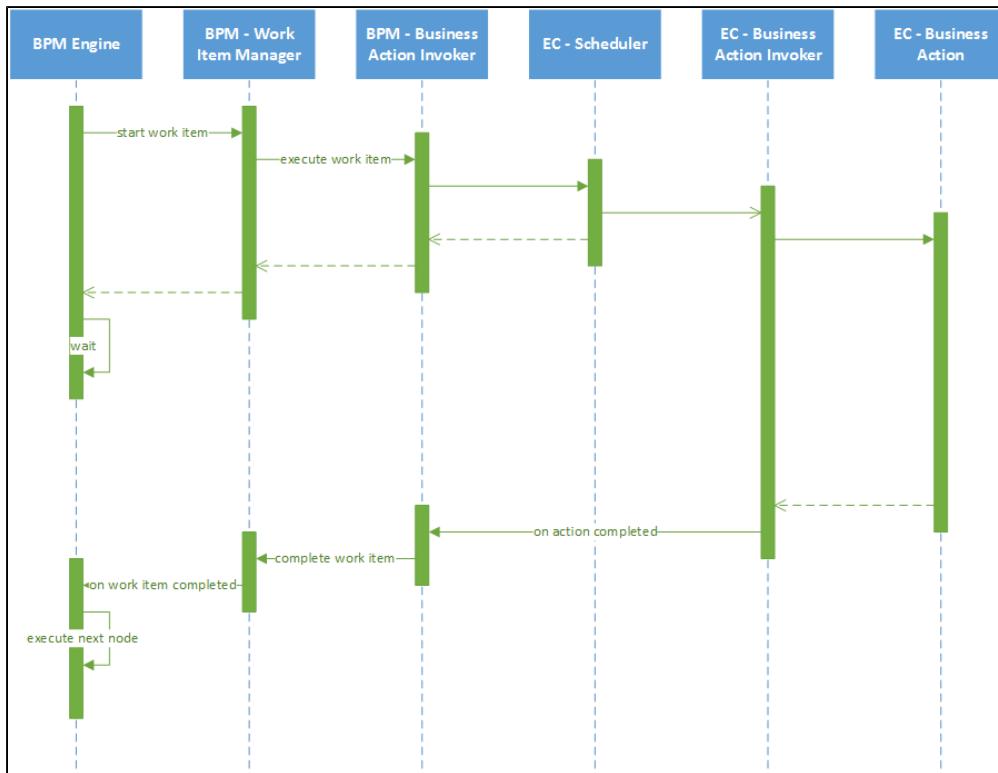


Finally, create a new Output Mapping in Assignments property:



## Internal Workflow

Following sequence diagram shows the internal workflow of one time business action invocation:



1. BPM Engine arrives at the business action work item node.
  2. BPM Engine calls Work Item Manager to start the work item.
  3. Work Item Manager calls registered work item handler – EC's business action asynchronous invoker – to invoke the business action. *Note the name "Business Action Invoker" is a simplified term in this document to avoid confusion, representing a group of classes that help achieves business action invocation.*
  4. Business action invoker notifies EC about the invocation via REST service.
  5. EC starts schedule "BpmSchedulerEnv" to start the business action. The call chain returns, and BPM Engine starts waiting for a complete callback.
  6. EC business action invoker is triggered by the scheduler, and executes the target business action synchronously.
  7. Business action execution finished (succeeded or failed), the invoker notifies the other invoker on BPM side.
  8. BPM Work Item Manager is notified that the work item has been completed.
  9. BPM Engine is notified that the work item has been completed.
  10. BPM Engine starts executing the next node.

## Process Action

Process Actions are actions being carried out by EC and triggered on demand from process engine. Process Actions are invoked by EC Scheduler via the BpmSchedulerEnv schedule, with the same manner as Business Action invocations.

A process action can be configured to have one or several action handlers. An action handler could be a business action or a generic action handler. If a process action has more than one action handler, users needs to define the execution order of the action handlers. And the action handlers could be configured as totally independent, meaning that all the action handlers can get the required handler parameters from the process, or the succeeding action handler could be configured to take the output of preceding action handler as input. Note that a process action is recommended to has no more than one business action as action handler.

Business actions/handlers inside a process action is invoked in given order. Parameters being provided by a process instance is input to the first business action, whose output serves input to the next business action. Business actions and handlers are chained to each other in this manner.

Although it is possible to chain multiple business actions together inside one process action, it is suggested to have all business actions in one process action for the same goal. The purpose to use Process Action instead of Business Action, is to achieve the following:

- Reuse of existing Business Actions
- Transform input or output of a business action
- Wrapping a business action with "pre" and "post" actions

### Defining Process Actions

In order to use process action in process, they need to be defined on EC Process Action Screen. The screen is located at EC -> Process Automation -> Process Action.

Action Name	Description
sample_process_action1	
sample_process_action2	
{AddNotificationAction}	
{ExecuteCalculationAction}	
{ExecuteEcIsSourceAction}	
{ExecuteEcIsTransformAndLoadAction}	
{ExecuteSqlAction}	
{SendReportAction}	

ACTION HANDLER				
Sort Order	Enabled	Class Name	Name	Comment
1	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.CreateProcessLogH	WriteProcessLog1	Write log before business ac
2	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	SqlExecution	business action example
3	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.CreateProcessLogH	WriteProcessLog2	Write result to process log

HANDLER PARAMETER OVERWRITE				
Parameter Name	Mapping Type	Mapping Value	Comments	
create_process_log.log.desi	Manually input	▼ Before sql execution		

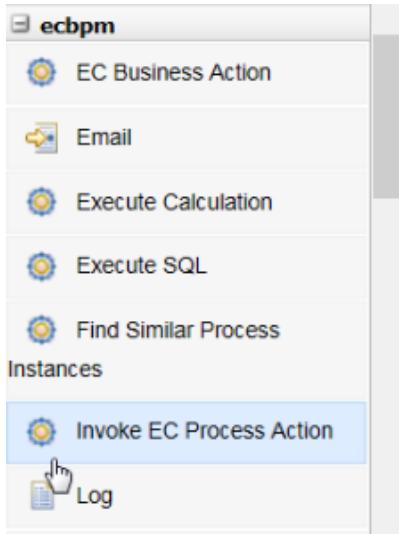
The screen contains three parts, as described below:

Component	Description
Action	The process action is identified by Action Name column, thus the action name need to be unique
Action Handler	Users need to define which action handlers to perform and the execution order when the corresponding process action is invoked. An action handler could be a business action or a generic process action handler
Handler Parameter Overwrite	<p>This section is optional. If the parameter of the action handler is given in the Handler Parameter Overwrite section, it would overwrite the default parameter provided by ecbpm. Default parameter values are provided in output-as-input manner.</p> <p>The mapping types of the parameter are categorized into four types described as following:</p> <ul style="list-style-type: none"> <li>• Default (no mapping): The action handler parameter is given by the process (to first handler) or output from previous handler (preceding handlers).</li> <li>• From an input parameter: The parameter is mapped to an output parameter of the previous action handler.</li> <li>• From an initial input parameter: The parameter is mapped to an initial input parameter given to the node in the process.</li> <li>• Manually input: The parameter value is given by manually input from the screen.</li> </ul>

## Use in Process

### Adding the Activity

"Invoke EC Process Action" activity can be found in Object Library panel, under the "ecbpm" group:



### Input and Output Parameters

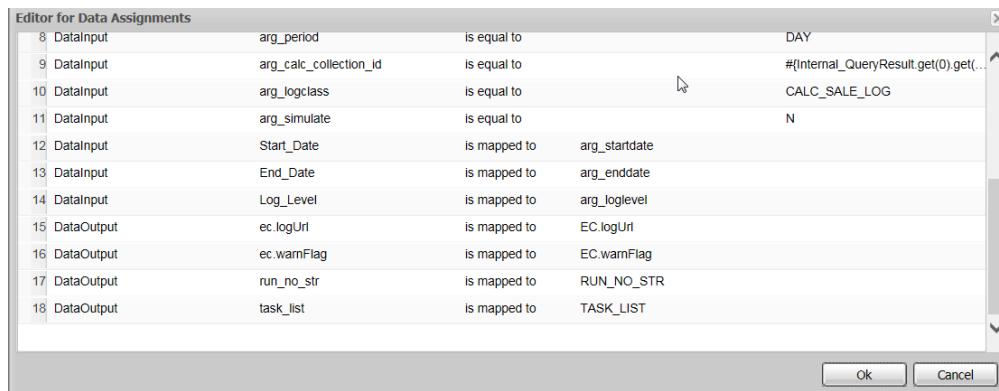
To use the node, drag it to the process canvas. The node has following pre-defined parameters:

Name	Type	Object Type	Optional	Description
process_action_name	Input	String	No	The name of the process action to invoke, which is defined on EC Process Action screen.

To send arguments to a process action, action handler input parameters need to be added. The input parameters should follow following rules:

1. Parameter name should follow the format "arg\_" + argument name. For example, to assign value to a process action handler argument "CONTRACT\_ID", the input parameter should be named as "arg\_CONTRACT\_ID"
2. The parameter name is case sensitive. The prefix "arg\_" needs to be in small cases. Action handler argument name needs to follow its own definition.

The result returned by the process action invoke is a Map<String, Object> containing the key and value of output parameters. To reference the values in the result object, it needs to be mapped to a process variable.



## Error Handling

Error handling of process action is very similar to business action, please refer to the "Error Handling" of business action chapter.

## Example - Wrapping Business Action Execution with Logs

This example illustrates how to write process event logs before and after a business action execution.

### Define the Process Action

Here we define a process action named "sample\_process\_action1" which is configured to have three action handlers.

Action Name	Description
sample_process_action1	
sample_process_action2	
{AddNotificationAction}	
{ExecuteCalculationAction}	
{ExecuteEcIsSourceAction}	
{ExecuteEcIsTransformAndLoadAction}	
{ExecuteSqlAction}	
{SendReportAction}	

**ACTION HANDLER**

Sort Order	Enabled	Class Name	Name	Comment
1	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.CreateProcessLogH	WriteProcessLog1	Write log before business action execution
2	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	SqlExecution	business action example
3	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.CreateProcessLogH	WriteProcessLog2	Write result to process log

**HANDLER PARAMETER OVERWRITE**

Parameter Name	Mapping Type	Mapping Value	Comments
create_process_log.log.description	Manually input	Before sql execution	

Add the configurations as described below:

**Action Handler 1**

The process action first writes process log to the screen before any business action is executed, the log description can be configured to describe the current process status or any useful information for users. The handler parameter mapping type of the first action handler is "Manually input" which means the input parameter value is given by the manually input from screen.

Action Handler Configuration

Sort Order	Enabled	Class Name	Name	Comments
1	Y	com.ec.bpm.ext.ec.handlers.CreateProcessLogHandler	WriteProcessLog1	Write log before business action execution

Handler Parameter Overwrite Configuration

Parameter Name	Mapping Type	Mapping Value	Comments
create_process_log.log.description	Manually input	Before sql execution	

**Action Handler 2**

The second action handler in the example is a business action example, here it simple calls a SQL function as configured.

**ACTION HANDLER**

Sort Order	Enabled	Class Name	Name	Comment
1	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.CreateProcessLogH	WriteProcessLog1	Write log before business action execution
2	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	SqlExecution	business action example
3	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.CreateProcessLogH	WriteProcessLog2	Write result to process log

**HANDLER PARAMETER OVERWRITE**

Parameter Name	Mapping Type	Mapping Value	Comments
SQL	Manually input	call ec_t_basis_user.surnam	
SqType	Manually input	function	

Action Handler Configuration

Sort Order	Enabled	Class Name	Name	Comments
2	Y	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	SqlExecution	business action example

Handler Parameter Overwrite Configuration

Parameter Name	Mapping Type	Mapping Value	Comments
SQL	Manually input	call ec_t_basis_user.surname('sysadmin')	
SqlType	Manually input	function	

**Action Handler 3**

The third action handler is also a "CreateProcessLogHandler", the handler input parameter mapping type is "From an input parameter" which means it is mapped to the output parameter of the previous action handler named "Sql Execution". And it will write the result returned by the "Sql Execution" handler to the process log on the screen (Process Overview Screen).

**ACTION HANDLER**

Sort Order	Enabled	Class Name	Name	Comment
1	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.CreateProcessLogH	WriteProcessLog1	Write log before business ac
2	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	SqlExecution	business action example
3	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.CreateProcessLogH	WriteProcessLog2	Write result to process log

**HANDLER PARAMETER OVERWRITE**

Parameter Name	Mapping Type	Mapping Value	Comments
create_process_log.log.description	From an input parameter	FUNCTION_RETUR	

Action Handler Configuration

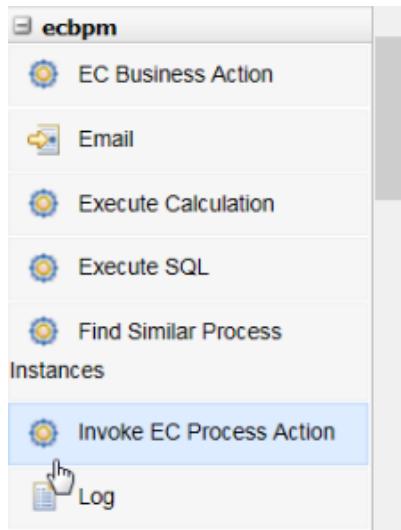
Sort Order	Enabled	Class Name	Name	Comments
3	Y	com.ec.bpm.ext.ec.handlers.CreateProcessLogHandler	WriteProcessLog2	Write result to process log

Handler Parameter Overwrite Configuration

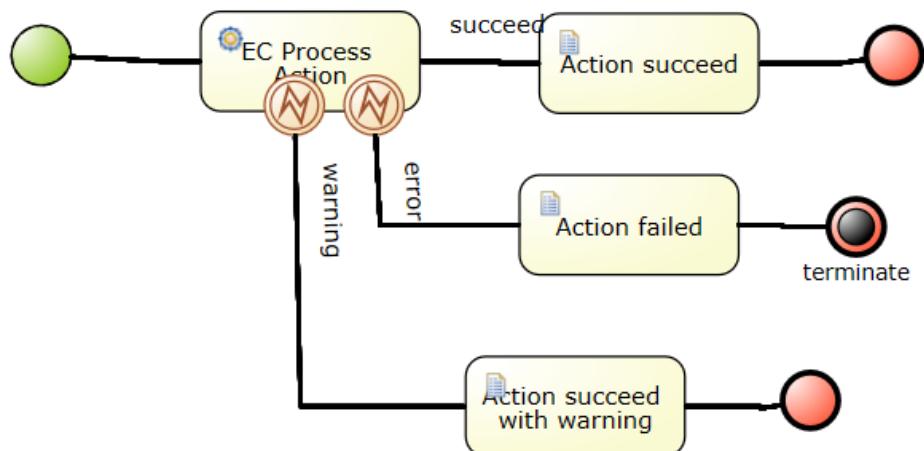
Parameter Name	Mapping Type	Mapping Value	Comments
create_process_log.log.description	com.ec.bpm.ext.ec.handlers.CreateProcessLogHandler	WriteProcessLog2	Write result to process log

**Adding the Activity**

The activity is available in Object Library of KIE Workbench Designer under category "ecbpmp".



To add it to the process, drag it onto the canvas, and add proper connection to preceding and succeeding nodes.



To change the node name, double click on the node and update the name.

### Configuring the Activity

#### **Data Input Set**

Click on the activity, add following parameters to Data Input Set.

Variable	Data Type
<b>process_action_name</b>	String
arg_SQL	String
arg_SqlType	String
arg_create_process_log.log.description	String

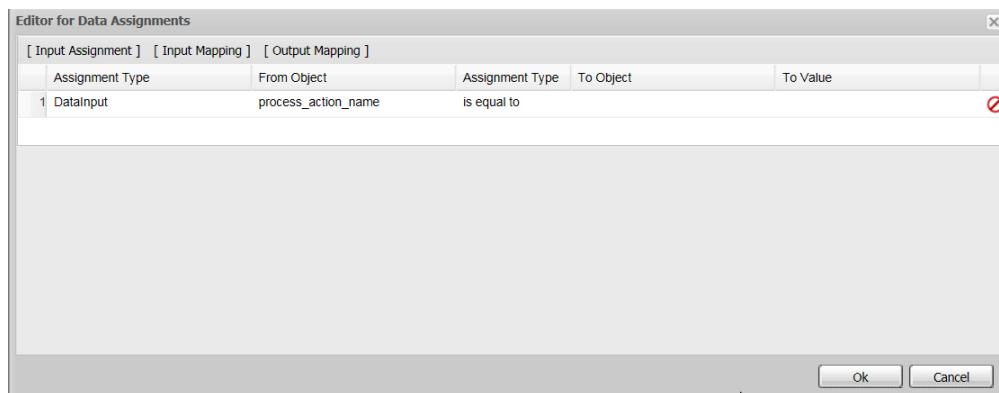
#### **Data Output Set**

Click on the activity, add following parameters to Data Output Set.

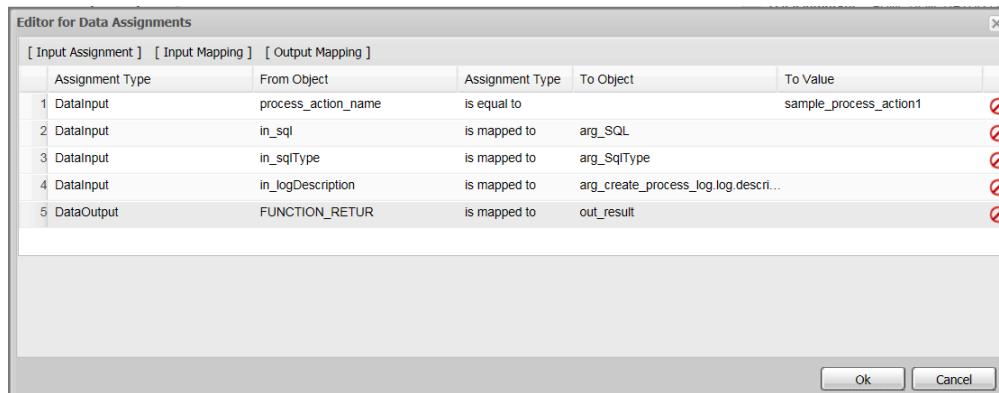
Variable	Data Type
FUNCTION_RETUR	java.lang.Object

### Data Assignments

To assign values to input parameters, open the Properties panel, and click the triangle button on Assignments row, this should show the Assignments dialog:



In this example, the input assignments and output assignments are configured as following:



### Executing the Process

Build and compile the process, create a new process template on EC with the process Id.

Execute the process via EC screen Process Automation -> Process Execution. More information, refer to section Process Execution and Management.

Process event logs is available via screen Process Automation -> Process Overview screen. Process Action results are available through the event log, as shown in screenshot below (row 5 and 6). Process Event Log are also been created as configured in Process Action (row 4 and 7).

The screenshot shows a process instance list and a detailed log view. The process instance list shows one entry: '72 Completed' for 'ec\_process\_action\_sample'. The log view shows the following entries:

Time	Node	Message
2015-12-07 13:39		Process instance completed (Completed)
2015-12-07 13:39	Action succeed	Completed activity 'Action succeed'
2015-12-07 13:39	Action succeed	Started activity 'Action succeed'
2015-12-07 13:39	EC Process Actor	Completed activity 'EC Process Action'
2015-12-07 13:39	EC Process Actor	System supervisor
2015-12-07 13:39	EC Process Actor	Before sql execution
2015-12-07 13:39	EC Process Actor	Started activity 'EC Process Action'
2015-12-07 13:39		Process instance started

Below the log, there are tabs for 'MESSAGE' and 'DATA'. The 'MESSAGE' tab shows a single message: 'System supervisor'.

## Example - Wrapping Business Action with Pre and Post User Exists

The second example shows how to leverage process action to perform pre and post actions for business action in process.

### Define the Process Action

Here we define a process action named "sample\_process\_action2" which is configured to have three action handlers.

Action Name	Description
sample_process_action1	
sample_process_action2	
{AddNotificationAction}	
{ExecuteCalculationAction}	
{ExecuteEcisSourceAction}	
{ExecuteEcisTransformAndLoadAction}	
{ExecuteSqlAction}	
{SendReportAction}	

### ACTION HANDLER

Sort Order	Enabled	Class Name	Name	Comment
1	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.CreateProcessLogH	WriteProcessLog	Write Process Log
2	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	PreAction	Pre-action before execute bi
3	<input checked="" type="checkbox"/>	com.ec.frmw.bs.calc.engine.CalcAction	RunCalculation	business action
4	<input checked="" type="checkbox"/>	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	PostAction	Post-action before execute t

### HANDLER PARAMETER OVERWRITE

Parameter Name	Mapping Type	Mapping Value	Comments
SQL	Manually input	call ecxx_sample.pre_action	
SqType	Manually input	procedure	

Add the configurations as described below:

#### Action Handler 1

"WriteProcessLog": Record current process status or any useful information to the process log.

#### Action Handler Configuration

Sort Order	Enabled	Class Name	Name	Comments
1	Y	com.ec.bpm.ext.ec.handlers.CreateProcessLogHandler	WriteProcessLog	Write Process Log before executing business action
2	Y	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	PreAction	Pre-action before execute business action
3	Y	com.ec.frmw.bs.calc.engine.CalcAction	RunCalculation	business action
4	Y	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	PostAction	Post-action before execute business action

Handler Parameter Overwrite Configuration

Parameter Name	Mapping Type	Mapping Value	Comments
create_process_log.log.description	Manually input	Write process log before calculation	

**Action Handler 2**

"PreAction": Perform pre-action for the following business action, such as querying "jobid" for the calculation business action.

Action Handler Configuration

Sort Order	Enabled	Class Name	Name	Comments
2	Y	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	PreAction	Pre-action before execute business action

Handler Parameter Overwrite Configuration

Parameter Name	Mapping Type	Mapping Value	Comments
SQL	Manually input	call ecxx_sample.pre_action	
SqlType	Manually input	procedure	

**Action Handler 3**

"RunCalculation": Calculation business action, it takes the output of "PreAction" as input for parameter "jobid".

Action Handler Configuration

Sort Order	Enabled	Class Name	Name	Comments
3	Y	com.ec.frmw.bs.calc.engine.CalcAction	RunCalculation	business action

Handler Parameter Overwrite Configuration

Parameter Name	Mapping Type	Mapping Value	Comments
SKIP_CALC	Manually input	N	
jobid	From an input parameter	PROCEDURE_RETUR	

**Action Handler 4**

"PostAction": Perform post-action to the output of the "RunCalculation" action handler.

#### Action Handler Configuration

Sort Order	Enabled	Class Name	Name	Comments
4	Y	com.ec.bpm.ext.ec.handlers.ExecSqlHandler	PostAction	Post-action after execute business action

#### Handler Parameter Overwrite Configuration

Parameter Name	Mapping Type	Mapping Value	Comments
SQL	Manually input	call ecxx_sample.post_action	
SqIType	Manually input	procedure	

#### **Data Assignments**

To assign values to input parameters, open the Properties panel, and click the triangle button on Assignments row, this should show the Assignments dialog:

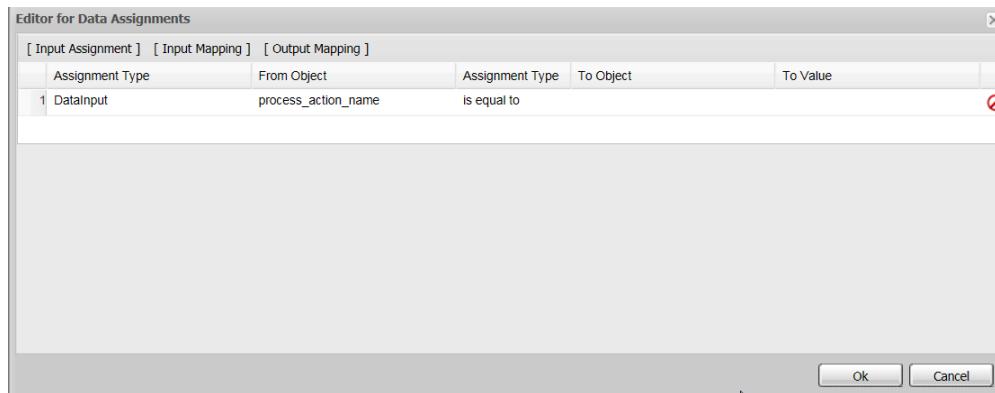


image2015-12

-8 9-44-32.png

In this example, the input assignments and output assignments are configured as following:

Editor for Data Assignments

[ Input Assignment ] [ Input Mapping ] [ Output Mapping ]

	Assignment Type	From Object	Assignment Type	To Object	To Value	
1	DataInput	process_action_name	is equal to	arg_sample_process_action2	sample_process_action2	✗
2	DataInput	in_logdescription	is mapped to	arg_create_process_log.log.description		✗
3	DataInput	in_sqType	is mapped to	arg_SqType		✗
4	DataInput	in_sql	is mapped to	arg_SQL		✗
5	DataInput	in_startDate	is mapped to	arg_startdate		✗
6	DataInput	in_endDate	is mapped to	arg_enddate		✗
7	DataInput	inLogLevel	is mapped to	arg_loglevel		✗
8	DataInput	arg_context	is equal to	EC_SALE_SA		✗
9	DataInput	internal_result	is mapped to	arg_jobid		✗
10	DataInput	arg_jobcode	is equal to	SS1_CNTR_CALC		✗
11	DataInput	arg_jobname	is equal to	SS1 Contract Calculation		✗
12	DataInput	arg_period	is equal to	DAY		✗
13	DataInput	arg_logclass	is equal to	CALC_SALE_LOG		✗
14	DataInput	arg_simulate	is equal to	N		✗
15	DataOutput	PROCEDURE_RETUR	is mapped to	internal_result		✗
16	DataOutput	ec.logUrl	is mapped to	out_EC.logUrl		✗
17	DataOutput	ec.warnFlag	is mapped to	out_EC.warnFlag		✗
18	DataOutput	run_no_str	is mapped to	out_RUN_NO_STR		✗
19	DataOutput	task_list	is mapped to	out_TASK_LIST		✗

Properties /FCProcessActions.htm

Ok Cancel

## Executing the Process

Build and compile the process, create a new process template on EC with the process Id.

Execute the process via EC screen Process Automation -> Process Execution. More information, refer to section Process Execution and Management.

## Predefined Process Actions

ecbpm ships with several predefined process actions, as shown below.

It is not suggested to change the name of these process actions, as they are being used by following ecbpm activities:

- Execute SQL
- Send EC Process Notification
- Execute Calculation

Process Action	Input Parameters	Output Parameters
{AddNotificationAction}	<ul style="list-style-type: none"> <li>• Description - (data type: String) Notification description</li> <li>• NotificationName - (data type: String) Notification name</li> <li>• ResponsibleRole - (data type: String) Role that is responsible for this notification</li> <li>• ResponsibleUser - (data type: String) User that is responsible for this notification</li> <li>• Severity - (data type: String) The severity of this notification (when type is Alarm)</li> <li>• Type - (data type: String) The type of this notification</li> </ul>	
{ExecuteCalculationAction}	<ul style="list-style-type: none"> <li>• AllocNetworkCode - (data type: String)</li> <li>• CalcCode - (data type: String)</li> <li>• ProdDay - (data type: String)</li> <li>• SKIP_LOG - (data type: String)</li> <li>• context - (data type: String)</li> <li>• logclass - (data type: String)</li> <li>• period - (data type: String)</li> <li>• simulate - (data type: String)</li> </ul>	<ul style="list-style-type: none"> <li>• EC.logUrl - (data type: String)</li> <li>• EC.warnFlag - (data type: String)</li> <li>• RUN_NO_STR - (data type: String)</li> <li>• CALC_STATUS - (data type: String)</li> <li>• EC_CalcEngine_User_Msg - (data type: String)</li> </ul>
{ExecuteSqlAction}	<ul style="list-style-type: none"> <li>• SQL - (data type: String) The SQL/function/procedure to be ran</li> <li>• SqlType - (data type: String) SQL type, see following table for options</li> </ul> <p>Available SQL Types:</p> <ul style="list-style-type: none"> <li>• select - Indicates the given SQL is a multi-column-multi-row select statement.</li> <li>• select_single_col - Indicates the given SQL is a single-column-single-row select statement</li> <li>• procedure - Indicates the given SQL is a procedure call, for example "call ecdp_business_function.AddBF List(null, null)".</li> <li>• update - Indicates the given SQL is an update statement.</li> <li>• function - Indicates the given SQL is a function call, for example "call ec_t_basis_user.surname('sysadmin')".</li> </ul>	<ul style="list-style-type: none"> <li>• SqlExecutionResult - (data type: Map&lt;Object&gt;)</li> </ul> <p>Members of the map SqlExecutionResult:</p> <ul style="list-style-type: none"> <li>• SQL_RETUR - (data type: List&lt;Map&gt;) Contains the rows returned by the select statement. Each row is a Map object, keyed by the column name.</li> <li>• SQL_RETUR - (data type: Integer) The rows being affected.</li> <li>• SQL_SINGLE_RETUR - (data type: Object) The single return value from the select statement.</li> <li>• FUNCTION_RETUR - (data type: Object) The return value of the function.</li> <li>• PROCEDURE_RETUR - (data type: String) Hard-coded string "NOEXCEPTION".</li> </ul>
{SendReportAction}	<ul style="list-style-type: none"> <li>• REPORT_NO - (data type: Integer/String) Number of the report to be sent via EC MHM.</li> </ul>	

## User Tasks and To-do List

It is usually a case that part of a automated process requires manually done. In BPMN, such work is also modeled as an activity, called User Task. The work flow for User Tasks are the same as Service Tasks, which is, process engine generates user tasks, assign them to groups or users according to process definition, and then progress the process instance once the task is done.

ecbpm supports user tasks supposed to be taken by EC users. Active and completed tasks can be viewed and operated through EC To-do List screen. Additionally, ecbpm also supports displaying formatted task information (input variables) with EC Data and UI integration. This section describes how User Task can be used in ecbpm.

The screenshot shows the ECBPM Todo List interface. At the top, there are tabs for 'AVAILABLE', 'COMPLETED', and 'EXCEPTIONAL'. Below this is a table with columns: Id, Task name, Status, Process Template, Assignee, Preassigned Users, Preassigned Groups, Created Date, and Due Date. The table contains several rows of task data. Below the table are sections for 'DESCRIPTION' and 'DETAILS', which show the task description 'Place an order by filling out the parameters' and its skipability status 'true'. At the bottom, there is a 'TASK INPUT' section with fields for 'Your Name' (set to 'supervisor'), 'Contract' (set to 'COSL\_INVENTORY'), 'Item Name' (set to 'Oil'), and 'Quantity' (set to '1'). Below these fields are buttons for 'REFRESH', 'COMPLETE TASK', 'STOP TASK', 'RELEASE TASK', and 'EDIT USERS'.

## Use In Process

### The activity

BPMN uses the notation below for User Tasks



This task can be found under Tasks category in the Object Library.

### Task Subject

Task subject is the short description of a task, and it will be shown to the task taker on To-do List screen. Due to jBPM doesn't support value injection (the #{} format) on Name property, ecbpm introduces its own task subject variable, named "ec.extension.task.subject". If not defined, the Name property is displayed instead. See the description of the variable below.

Variable	Type	Data Type	Description
ec.extension.task.subject	Input	String	Task subject. If not given, Name property on User Task activity is used.

### Task Description

ecbpm reads Comments property as task description (because of the implementation of jBPM).

### Task Assignment

Task assignment is done during process design time. A task can be pre-assigned to a set of users (called actors in jBPM) or groups. Pre-assigned tasks will show-up in target user's To-do list.

Following properties indicates who a task is pre-assigned to:

Parameter	Data Type	Description	Example
Actors	List	List of pre-assigned users for a task. When the task is triggered, only the selected users can see the task. EC login Id should be used.	sysadmin
Groups	List	List of pre-assigned user roles for a task. When the task is triggered, only users from selected user roles can see the task. EC user role Id should be used.	SYST.ADM

### Task Attributes

Task attributes serve as additional information for task takers, and usually contains process instance dependent values. Defined attributes are listed on To-do List screen for task takers to read.

#### ***Value Assignment***

Task attributes are provided as node Input Data (Input Variable). They are normal input variables as used on other activities, therefore their values can be assigned either by Input or Variable Mapping. See previous sections or related jBPM documents for how to assign values to input variables.

#### ***Formatting***

As attributes are visible on EC's To-do List screen, ecbpm provides support on variable formatting using EC presentation attributes, for example, viewhidden, vieweditable, PopupUrl, etc.. Presentation attributes are provided in JSON format, via variable "ec.extension.variableMeta".

Variable	Type	Data Type	Description
ec.extension.variableMeta	Input	String	Task attributes presentation meta. This variable contains EC presentation attribute for all Input Variables defined on the User Task activity.

Variable value should use the following format:

```
[  
    'task_attribute_name_1': [  
        'viewtype':'text',  
        'viewlabel':'Your Name',  
        'viewrow':'1',  
        'datarequired':'true',  
  
        'PopupQueryURL':'/com.ec.frmw.co.screens/query/get_user_popup.xml',  
        'PopupLayout':'/com.ec.frmw.co.screens/layout/user_popup.xml',  
        'PopupDependency':'',  
        'PopupReturnColumn':'2',  
        'PopupWidth':'250',  
        'PopupHeight':'300',  
        'PopupCache':'true'  
    ],  
    'task_attribute_name_2': [  
        'viewtype':'text',  
        'viewlabel':'Contract',  
        'viewrow':'2',  
        'datarequired':'true'  
    ]  
]
```

## Task Actions

Task actions are additional actions a task taker can perform on EC to help finishing the task. Task actions should be designed as utilities, for task takers to complete the task easier, rather a mandatory step, as task actions are not ran unless triggered by the task taker. This is to make sure the task taker always have full control on what is done for his/her task.

Defined actions are displayed as *buttons* on EC To-do List screen.

### Configuration

Task actions are defined in User Task activity's input data (variable), via variable "ec.extension.task.actions". The value is provided as a map in Json format.

Variable	Type	Data Type	Description
ec.extension.task.actions	Input	String	<p>Defined actions. Actions are defined together in a map with its display name as key. Each action has following attributes:</p> <ul style="list-style-type: none"> <li>• name - Action display name</li> <li>• type - Type of the action, see following sections for available options</li> <li>• description - Action description</li> <li>• key - Action operation key. The usage of this attribute is dependent on action type</li> <li>• order - Display order of action</li> <li>• parameters - The parameters to the action</li> </ul>

Variable value should use the following format:

```
[
  '{name}': [
    'type':'{action_type}' ,
    'description':'{action_description}' ,
    'key':'{action_key}' ,
    'order':{action_order} ,
    'parameters': [
      '{param1_name}':'{param1_value}' ,
      '{param2_name}':'{param2_value}' ,
      ...
    ]
  ,
  '{name}': [
    'type':'{action_type}' ,
    'description':'{action_description}' ,
    'key':'{action_key}' ,
    'order':{action_order} ,
    'parameters': [
      '{param1_name}':'{param1_value}' ,
      '{param2_name}':'{param2_value}' ,
      ...
    ]
  ]
]
```

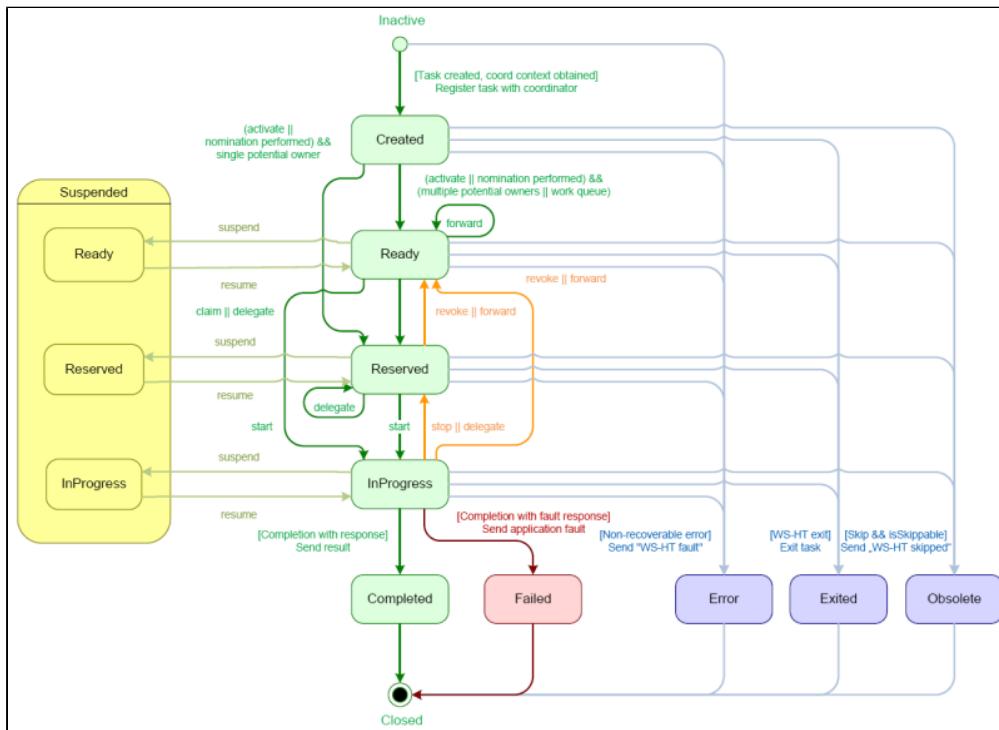
### Action Types

Following types of actions are supported:

Action Type	Description	Key	Parameters
ec_url	Shows a EC screen in a modal (popup) window.	The relative URL of a EC screen, for example "/FrontController/com.ec.fr mw.co.screens/maintain_access_users"	Parameters given to this type of action are sent to the screen via URL

## Task Workflow

User tasks follows a pre-defined workflow, as displayed below (from <http://docs.jboss.org/jbpm/v6.1/userguide/jBPMTaskService.html>):



1. Created – Task is created
2. Ready/Ready for Claim – Task is ready to be assigned
3. Reserved – Task has been assigned to a user
4. In Progress – The assignee is now working on the task
5. Completed – The task has been completed by the assignee

Task workflow operations are displayed as buttons on To-do List screen, for example:

REFRESH

COMPLETE TASK

STOP TASK

RELEASE TASK

Buttons are displayed according to task status. See below sections on how a task should be processed.

## Processing Tasks in EC

It is suggested to access tasks within EC in order to provide better UI integration. EC screen for task operation is EC -> Process Automation -> To-do List.

The screenshot shows the 'Todo List' screen with the following interface elements:

- Header:** Shows icons for back, forward, search, and filter, followed by the title 'Todo List'.
- Filter Bar:** Includes tabs for 'AVAILABLE', 'COMPLETED', and 'EXCEPTIONAL'.
- Table:** A grid displaying task details with columns: Id, Task name, Status, Process Template, Assignee, Preassigned Users, Preassigned Groups, Created Date, and Due Date. The table contains four rows of data.
- Description Area:** A large text area labeled 'DESCRIPTION' containing the task description 'Place an order by filling out the parameters'. Below it is a 'DETAILS' section with a 'Skipable' checkbox set to 'true'.
- Task Input Area:** A form labeled 'TASK INPUT' with fields for 'Your Name' (set to 'supervisor'), 'Contract' (set to 'COSI\_INVENTORY'), 'Item Name' (set to 'Oil'), and 'Quantity' (set to '1').
- Action Buttons:** At the bottom are buttons for 'REFRESH', 'COMPLETE TASK', 'STOP TASK', 'RELEASE TASK', and 'EDIT USERS'.

When a process instance arrives at a User Task activity, a To-do List Item is created. The items is visible to either all users or pre-assigned users on this screen.

The To-do List screen categorizes tasks into three types:

- Available – all on going tasks pre-assigned or assigned to current user, or pre-assigned to a role the current user has.
- Completed – all tasks completed by current user. Note it only shows completed tasks in active process instances.
- Exceptional – all exceptional tasks assigned to current user. Exceptional tasks includes failed, error, exited and obsolete tasks. Note it only shows completed tasks in active process instances.

The screen lists task attributes (Task Details) and task form (Task Input) of the selected task.

Task actions are listed on the bottom of the screen as defined in process.

When a task is in progress, it is possible to edit the Task Input values and save. After completing a task, it is not possible to update the values anymore.

Command buttons are available to on going tasks according to task state, clicking on the buttons will take a task to a new state.

Button	Description	Available for state	Next state
Claim Task	Assign a task to current user	Ready for Claim	Reserved
Release Task	Release a task assigned to current user	Reserved	Ready for Claim
Start Task	Start working on a task	Reserved	In Progress
Stop Task	Stop working on a task	Work In Progress	Reserved
Complete Task	Complete a task	In Progress	Completed

## Samples

ecbpm shipped with some sample processes for process authors to start with. All sample processes can be found in repository "ecbpm-processes-sample" under Organizational Unit "ec".

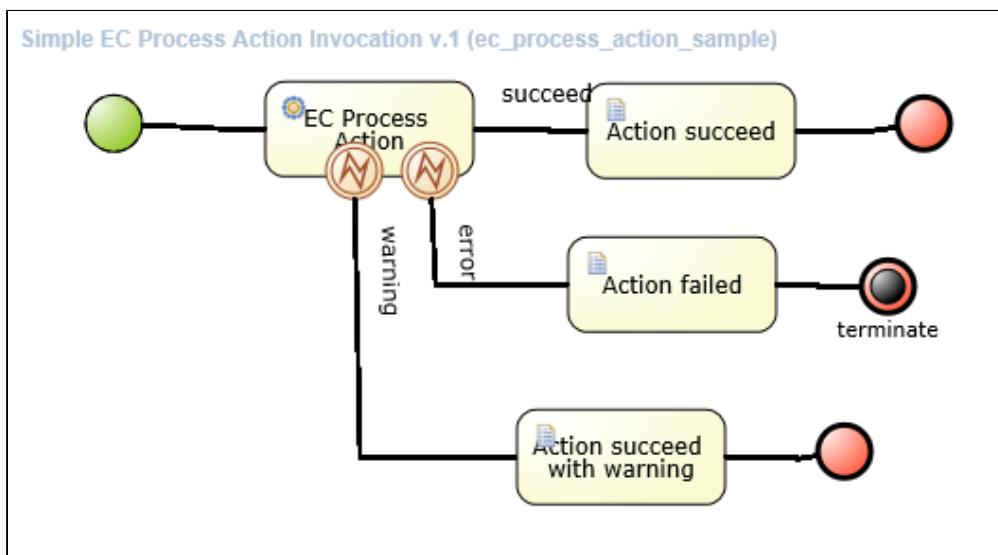


The existence of sample repository is checked during Wildfly startup, and will be automatically created if not found. Therefore, to re-create the sample repository, delete the repository from KIE Workbench and restart the Wildfly application server.

Sample processes are not deployed by default, and no Process Templates are created for them. To execute sample processes in EC, see section "Process Execution and Management" for instruction. For process template parameters, see the following sections.

This section introduce each sample process and their parameters.

### Simple EC Process Action Invocation



This sample demonstrates EC Process Action invocation and error handling. The process action used by this sample is called "sample\_action", which does not exist on EC by default. Therefore, when executing this process, the action would fail, and task Action failed will be triggered.

#### Process Id

ec\_process\_action\_sample

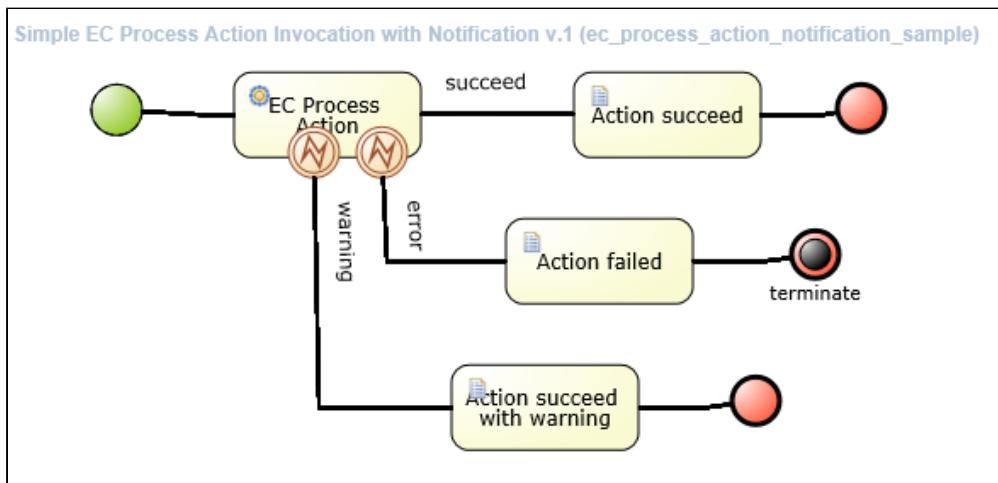
#### Process Nodes

Type	Name	Description
Task	EC Process Action	A service task that invokes specified process action on EC.
Catching Event	warning	A boundary catching event listens for WARNING return values from business action.
Catching Event	error	A boundary catching event listens for ERROR return value or process action exception.
Task	Action succeed	Called when the process action executed successfully. This task adds a process event log, which can be viewed on Process Overview screen.
Task	Action failed	Called when the process action returned ERROR or threw an exception. This task adds a process event log.
Task	Action succeed with warning	Called when the process action returned WARNING. This task adds a process event log.

#### Process Template Parameters

Name	Type	Sub Type	Mandatory	Description

#### Simple EC Process Action Invocation with Notification



This sample demonstrates sending Process Notification during EC Process Action invocation. The process action used by this sample is called "sample\_action", which does not exist on EC by default. Therefore, when executing this process, the action would fail, and task Action failed will be triggered.

Process notification definitions are defined on activity EC Process Action via input variable "ec.extension.notifications".



##### Caution

Process Notification feature is only for back-capability purpose. More change will come to implementation and configuration in future versions.

On completion of this process, two notification will be created, one on invocation of EC Process Action, and one post the invocation. Process Notifications can be viewed via EC screen Process Automation -> Process Notifications.

Process Notifications									
From Date	To Date	Notification Type	Acknowledged						
2015-12-7	2015-12-09		<input type="checkbox"/>						
Notification	Type	Severity	Responsible User	Responsible Role	Process Template	Process Definition	Node Name	Created Time	Acknowledged
Error from EC Process	Alarm	Information	sysadmin	revenue	ec_process_action_no	ec_process_action_no	EC Process Action	2015-12-07 11:26	<input type="checkbox"/>
Notification from node	Status	Information	sysadmin		ec_process_action_no	ec_process_action_no	EC Process Action	2015-12-07 11:26	<input type="checkbox"/>
Error from EC Process	Alarm	Information	sysadmin	revenue	ec_process_action_no	ec_process_action_no	EC Process Action	2015-12-07 11:26	<input type="checkbox"/>
Notification from node	Status	Information	sysadmin		ec_process_action_no	ec_process_action_no	EC Process Action	2015-12-07 11:26	<input type="checkbox"/>
Error from EC Process	Alarm	Information	sysadmin	revenue	ec_process_action_no	ec_process_action_no	EC Process Action	2015-12-07 11:23	<input type="checkbox"/>
Notification from node	Status	Information	sysadmin		ec_process_action_no	ec_process_action_no	EC Process Action	2015-12-07 11:23	<input type="checkbox"/>
Acknowledge comment									
This is a notification									
Description									

### Process Id

ec\_process\_action\_notification\_sample

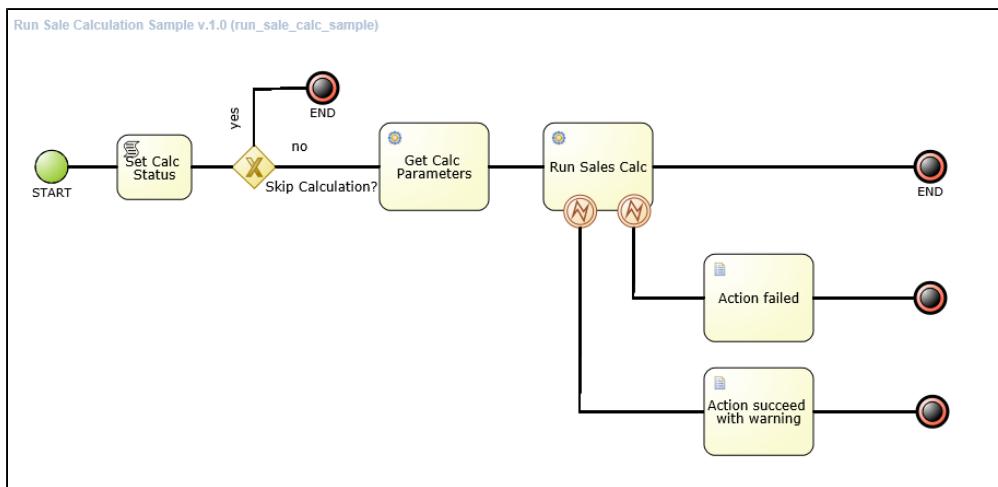
### Process Nodes

Type	Name	Description
Task	EC Process Action	A service task that invokes specified process action on EC. Process notifications are created pre and post the activity invocation.
Catching Event	warning	A boundary catching event listens for WARNING return values from business action.
Catching Event	error	A boundary catching event listens for ERROR return value or process action exception.
Task	Action succeed	Called when the process action executed successfully. This task adds a process event log, which can be viewed on Process Overview screen.
Task	Action failed	Called when the process action returned ERROR or threw an exception. This task adds a process event log.
Task	Action succeed with warning	Called when the process action returned WARNING. This task adds a process event log.

### Process Template Parameters

Name	Type	Sub Type	Mandatory	Description

### Run Sale Calculation Sample



This sample demonstrates invoking a sale calculation using EC Business Action Invocation activity and error handling. This process executes business action "com.ec.frmw.bs.calc.engine.CalcAction", with job "SS1 Contract Calculation".

"Get Calc Parameters" executes a SQL on EC to convert specified object code to object ID, which is required by the business action.

#### Process Id

run\_sale\_calc\_sample

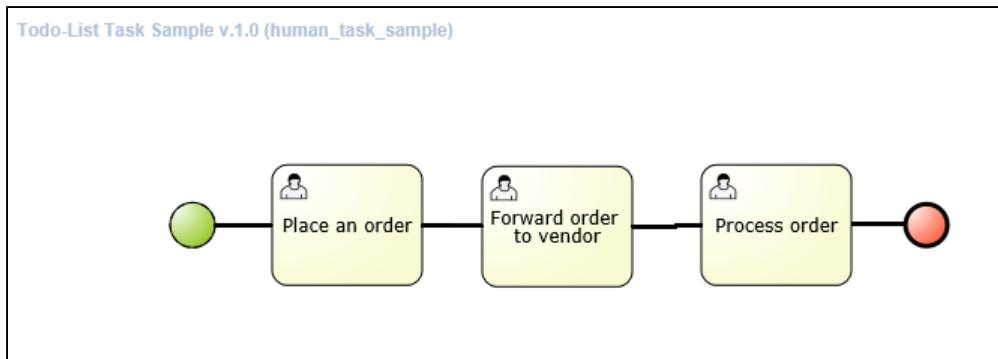
#### Process Nodes

Type	Name	Description
Task	Set Calc Status	Executes Java code on process engine to set process variable CALC_STATUS to 0. This node does no affect to later process nodes, it is only for demonstration purpose.
Geteway	Skip Calculations?	Gateway that look at the input process variable "SKIP_CALC" for if sales calculation should be skipped.
Task	Get Calc Parameters	Executes process action "{ExecuteSqlAction}" on EC with SQL "SELECT EC_CALC_COLLECTION.object_id_by_uk('SS1_CNTR_DAY') as calc_collection_id, ec_calculation.object_id_by_uk('SS1_CNTR_CALC') as job_id FROM DUAL". The SQL converts calculation job code to object Id.
Task	Run Sales Calc	Executes business action "com.ec.frmw.bs.calc.engine.CalcAction" with given parameters.
Catching Event	warning	A boundary catching event listens for WARNING return values from business action.
Catching Event	error	A boundary catching event listens for ERROR return value or process action exception.
Task	Action failed	Called when the process action returned ERROR or threw an exception. This task adds a process event log with returned message from calculation.
Task	Action succeed with warning	Called when the process action returned WARNING. This task adds a process event log with returned message from calculation.

#### Process Template Parameters

Name	Type	Sub Type	Mandatory	Description
Start_Date	Basic Type	Date	Yes	Start date of the calculation
End_Date	Basic Type	Date	Yes	End date of the calculation
Log_Level	EC Code Type	CALC_LOG_LEVEL	Yes	Calculation log level
SKIP_CALC	Basic Type	Boolean	Yes	Indicates whether the calculation should be skipped

#### Todo-List Task Sample



This sample demonstrates User Tasks creation. There are three tasks required to be executed in order.

#### Process Id

human\_task\_sample

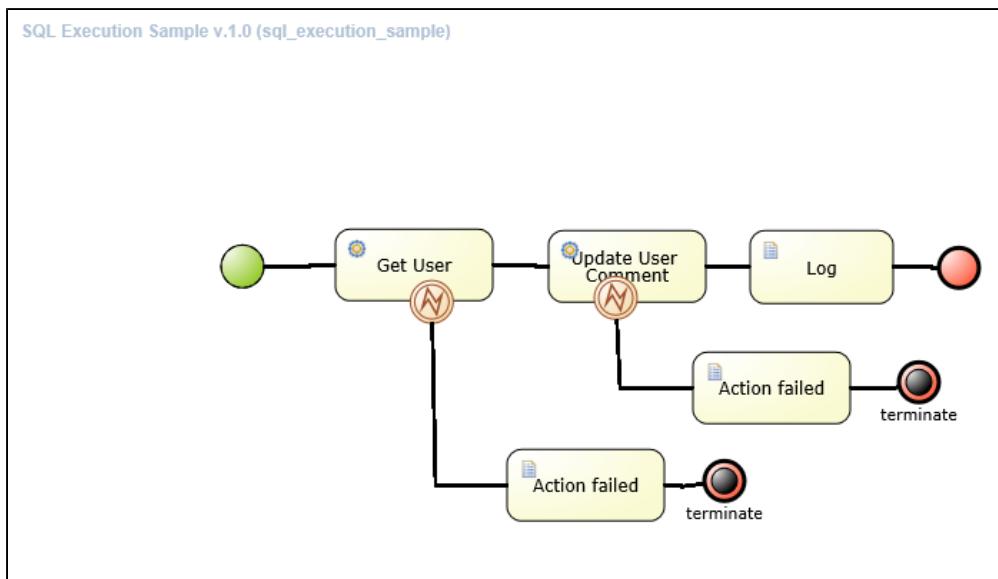
#### Process Nodes

Type	Name	Description
Task	Place an order	A user task activity. This task contains four fields for task takers to fill-in, two drop-downs, one free-text string fields, and one number fields. All fields are decorated using EC presentation meta via the meta variable.
Task	Forward order to vendor	This task takes all the four previous task results as input, which is displayed under Task Details section on To-do List screen. It contains one free-text comment field for task taker to fill-in.
Task	Process order	This task takes all previous task results as input, and a check-box field as output. See this task for how to show a check-box on To-do List's Task Result section.

#### Process Template Parameters

Name	Type	Sub Type	Mandatory	Description

#### SQL Execution Sample



This sample demonstrates executing a SELECT and a UPDATE statement on EC database.

The process first queries USER\_ID from T\_BASIS\_USER table for specified user\_id, then updates TEXT\_4 on T\_BASIS\_USER with queried data.

#### Process Id

sql\_execution\_sample

#### Process Nodes

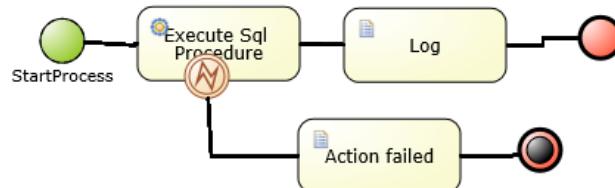
Type	Name	Description
Task	Get User	Executes query "select USER_ID, SURNAME, GIVEN_NAME from t_basis_user where user_id = '#{userid}'" on EC for specified userid.
Task	Action failed (1)	Triggered when "Get User" failed.
Task	Update User Comment	Executes query "update t_basis_user set TEXT_4 = 'name: #{internal_getUserResult[0].SURNAME} #{internal_getUserResult[0].GIVEN_NAME}' where user_id = '#{internal_getUserResult[0].USER_ID}'" on EC.
Task	Action failed (2)	Triggered when "Update User Comment" failed.
Task	Log	Writes a log to BPM server log.

#### Process Template Parameters

Name	Type	Sub Type	Mandatory	Description
userid	Basic Type	String	Yes	User Id to query and update.

#### Execute SQL Procedure Sample

## Execute Sql Procedure Sample v.1.0 (sql\_procedure\_execution\_sample)



This sample demonstrates executing a procedure on EC database.

Procedure being executed is ecdp\_business\_function.AddBFLList(null, null).

#### Process Id

sql\_procedure\_execution\_sample

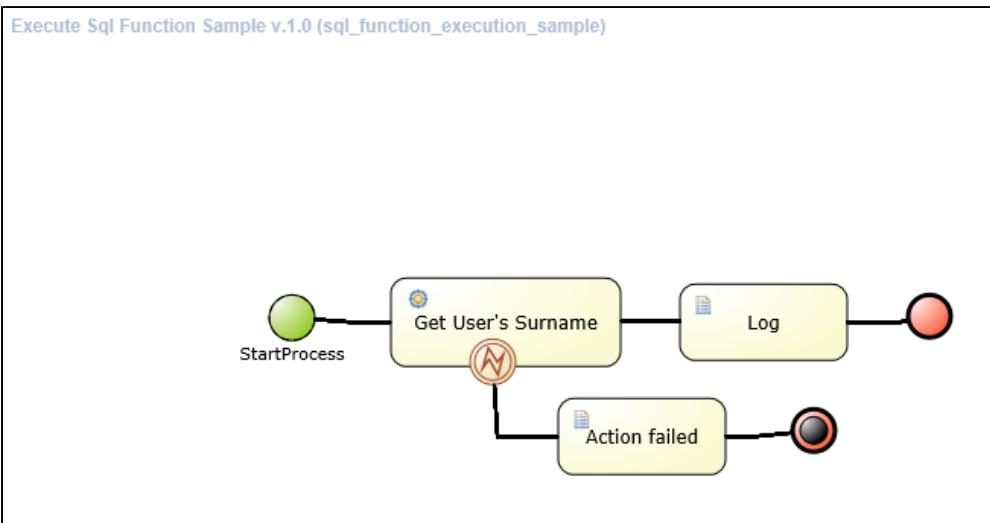
#### Process Nodes

Type	Name	Description
Task	Execute Sql Procedure	Executes procedure ecdp_business_function.AddBFLList(null, null) on EC.
Task	Action failed	Triggered when "Execute Sql Procedure" failed.
Task	Log	Writes a log to BPM server log.

#### Process Template Parameters

Name	Type	Sub Type	Mandatory	Description

#### Execute SQL Function Sample



This sample demonstrates executing a function on EC database.

Function being executed is `ec_t_basis_user.surname('sysadmin')`.

#### Process Id

`sql_function_execution_sample`

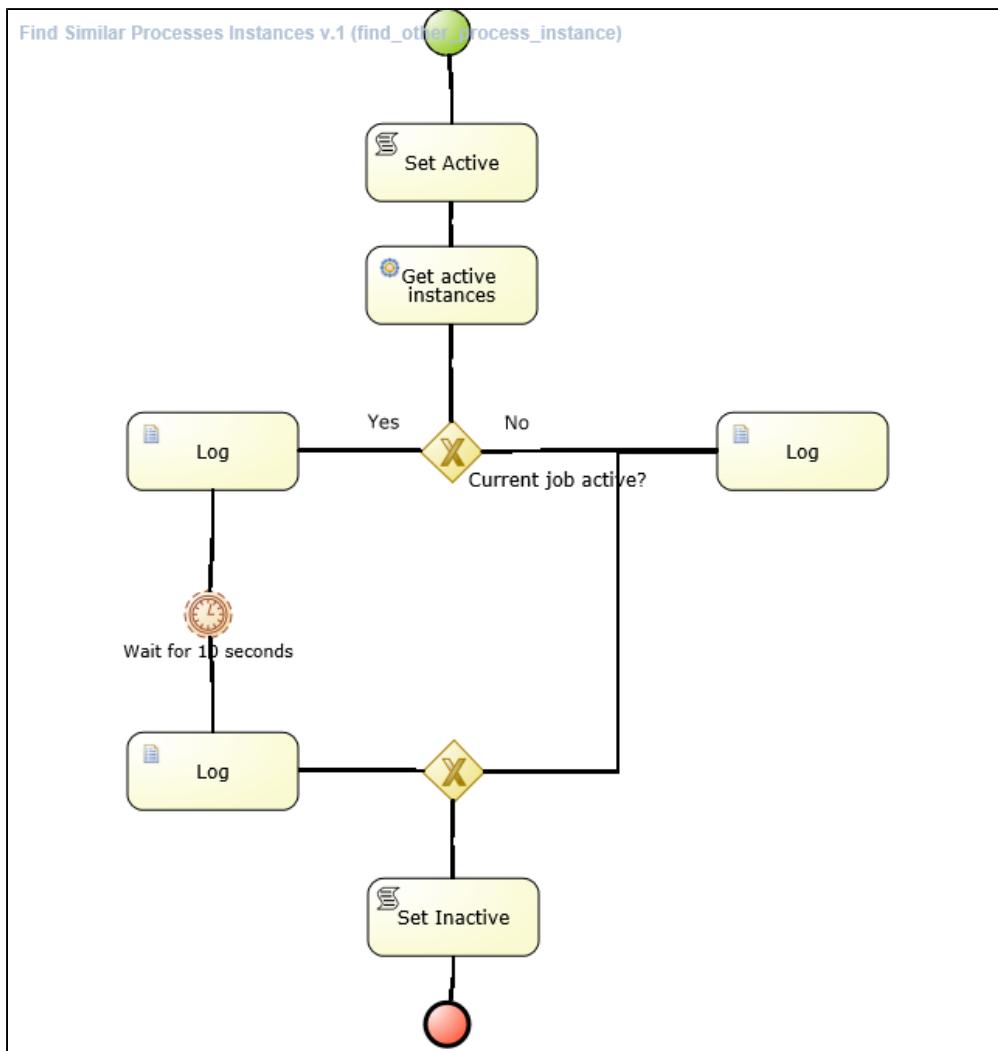
#### Process Nodes

Type	Name	Description
Task	Get User's Surname	Executes function <code>ec_t_basis_user.surname('sysadmin')</code> on EC. The process action stores the result in variable <code>FUNCTION_RETUR</code> , which is mapped to process variable <code>Internal_GetSurnameResult</code> .
Task	Action failed	Triggered when "Get User's Surname" failed.
Task	Log	Writes a log to BPM server log.

#### Process Template Parameters

Name	Type	Sub Type	Mandatory	Description

#### Find Similar Processes Instances



This sample demonstrates finding active process instances in process by a process variable value.

The process demonstrates a case where a single instance of a process is allowed. Singleton is achieved by checking existence of process instances with its variable "InternalProcessState" sat to "ACTIVE". If any instance found, the new process instance will exit immediately. The 10 seconds wait in the process is only for testing purpose, which ensures a process instance to be active for at least 10 seconds, and new instances will exit without waiting. In real-life usages, the timer event can be replaced with actual work ought to be carried out by the process.

#### Process Id

find\_other\_process\_instance

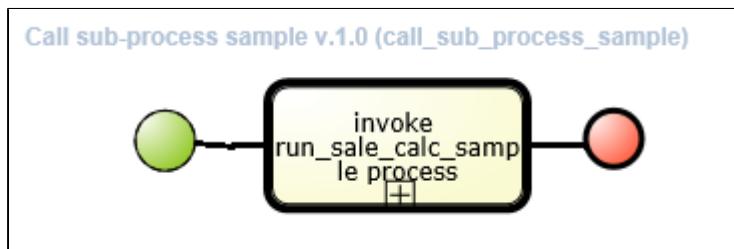
#### Process Nodes

Type	Name	Description
Task	Set Active	Sets process variable "InternalProcessState" to "ACTIVE", so it can be queried by "Get active instances" node on newer instances.
Task	Get active instances	Finds out process instances (excluding current one) that have process variable "InternalProcessState" set to "ACTIVE".
Gateway	Current job active?	A gateway determines whether to exit current instance or continue with actual process job.
Catching Event	Wait for 10 seconds	A timer catching event that makes the process instance hang for 10 seconds, to simulate "actual process job".
Task	Set Inactive	Sets process instance variable "InternalProcessState" to "INACTIVE", so new process instances do not exit without executing "actual process jobs".

#### Process Template Parameters

Name	Type	Sub Type	Mandatory	Description

#### Call sub-process sample



This sample demonstrates calling another process as a sub-process.

The sub-process being called is the "Run Sale Calculation Sample" (run\_sale\_calc\_sample). Variables to the sub-process are mapped from parent process. The sub-process runs synchronously with its parent, which means parent has to wait for its completion before progressing to next node.

#### Process Id

call\_sub\_process\_sample

#### Process Nodes

Type	Name	Description
Task	invoke run_sale_calc_sample process	Invokes process "Run Sale Calculation Sample" (run_sale_calc_sample) as a sub-process, and waits for its completion.

### Process Template Parameters

Name	Type	Sub Type	Mandatory	Description
Start_Date	Basic Type	Date	Yes	Start date of the calculation
End_Date	Basic Type	Date	Yes	End date of the calculation
Log_Level	EC Code Type	CALC_LOG_LEVEL	Yes	Calculation log level
SKIP_CALC	Basic Type	Boolean	Yes	Indicates whether the calculation should be skipped

## API

Currently ecbpm does not open any API to third party code, any usage to ecbpm internal interfaces, including ICommandHandler, IGenericCommandHandler, etc., are not supported and might not be working in next release. The only API supported by ecbpm is EC's Business Action interface (BusinessAction), all process action handlers defined by third party should implement this interface rather the Command Handler.

The ecbpm team is working on infrastructure heavily and its internal API is expected to change in next releases. However, all documented functionality and process meta variables will be supported. No internal APIs are open to third party code before they are finalized.

When it comes to customized Work Item Handlers, KIE's deployment descriptor can be used. However, as ecbpm is trying to isolate jBPM as much as possible, customized Work Item Handlers might not be suggested or supported in the future, but currently they can still be implemented if it is really needed. It is suggested that all third-party code should be implemented either as Script Task, or Business Actions.