

MaximoDev Java Library

**Version 1.2**

# Installation

## Deploy Classes

Copy class files into *[SMPDIR]\maximo\applications\maximo\businessobjects\classes*

Rebuild and redeploy *maximo.ear* file.

## Logging

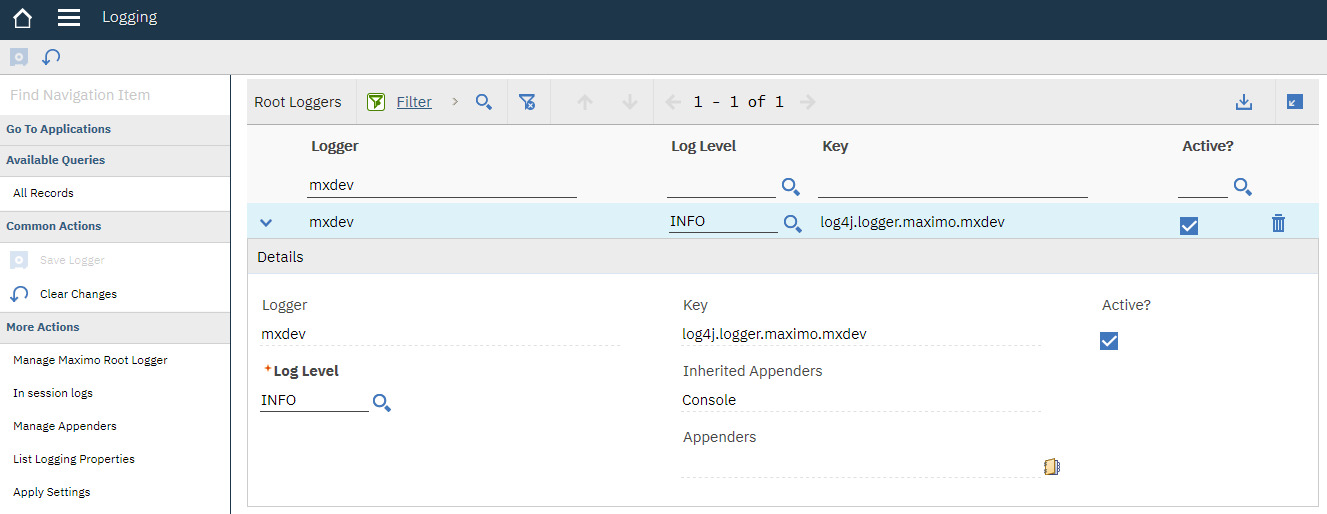
The cron task logs in ‘maximo.mxdev’ logger. If you need to debug your configuration you can configure the logger as follows.

Open the Logging application and create the following logger.

Logger: mxdev

Log Level: INFO

Key: log4j.logger.maximo.mxdev



# FlatFileProcess Cron Task

The FlatFileProcess Cron Task allows to perform few basic manipulation of CSV files to adapt their format to a Maximo-compliant structure.

It allows to:

* Reorder columns
* Skip rows from source files
* Insert one or to two headers to generate MIF file headers and rename column names
* Extract parts of the fields
* Parse dates and convert them to the correct format accepted by MIF

## Configuration

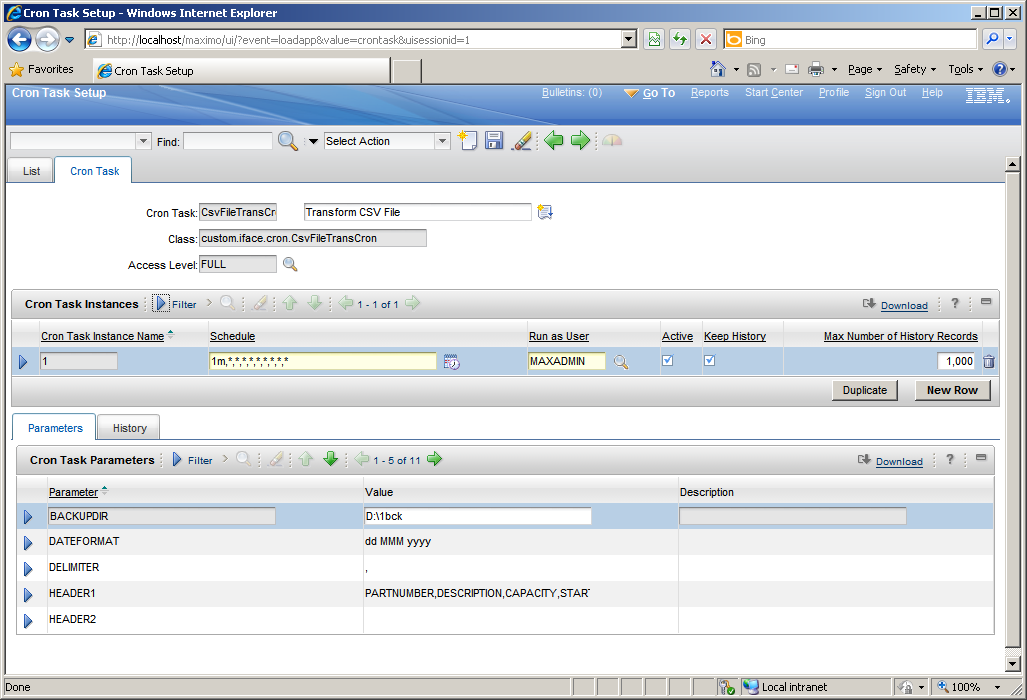
Go To > System Configuration > Platform Configuration > Cron Task Setup

Create a new Cron Task and set it up as follows:

* **Name**: FlatFileProcessCron
* **Description**: Transform CSV File
* **Class**: mxdev.iface.cron.FlatFileProcessCron

Create a cron task instance and configure parameters as follows.

* **INPUTDIR**: Directory where the files to be processed are located.
* **INPUTFILE**: Filter for files to be processed. For example setting it to ‘\*.csv’ will process all the files that have csv file extension in the input folder.
* **BACKUPDIR**: Files are moved in this folder once they are processed.
* **OUTPUTDIR**: Transformed files are stored here.
* **HEADER1**: Header row that will be written to the output files.
* **HEADER2**: A second (optional) row that will be written to the output files.
* **SOURCECOLUMNS**: Defines which columns from the source files are copied to the output files. Column numbers must be separated with commas. See below for details.
* **SKIPLINES**: Number of lines that are to be skipped from the input files.
* **TEXTQUALIFIER**: Character used to enclose text fields in input files.
* **DELIMITER**: Input files delimiter.
* **DATEFORMAT**: Allows to specify format of dates in the input file. Refer to the appendix for supported tags.
* **ESCAPEXML**: Converts XML predefined entities: &, ', <, >



The SOURCECOLUMNS parameter allows specifying the subset of columns to be transformed and their order. It also allows some processing. Let’s see some examples.

* **2,8,4** : Copies columns 2, 8 and 4 of the source file to the output file.
* **2,[Test]** : Copies column 2 and then put a constant value ‘Test’.
* **2,CURRDATETIME** :Copies column 2 and then put the current timestamp.
* **2,4.1-3** : Column 2 is copied as it is. Substring from character 2 to 3 of column 4 is copied. Note that index starts from 0 so 1 means 2nd character. The end index is exclusive to ‘3’ points to 4th character but it is not included in the output string.
* **2,4..DATE** : Column 2 is copied as it is. Column 4 is parsed as a date according to DATEFORMAT parameter.
* **2,4.13-24.DATE** : Column 2 is copied as it is. Substring from character 13 to 24 of column 4 is parsed as a date according to DATEFORMAT parameter.

## Date Formatting

|  |  |  |  |
| --- | --- | --- | --- |
| Letter | Date or Time Component | Presentation | Examples |
| G | Era designator | Text | AD |
| y | Year | Year | 1996; 96 |
| M | Month in year | Month | July; Jul; 07 |
| w | Week in year | Number | 27 |
| W | Week in month | Number | 2 |
| D | Day in year | Number | 189 |
| d | Day in month | Number | 10 |
| F | Day of week in month | Number | 2 |
| E | Day in week | Text | Tuesday; Tue |
| a | Am/pm marker | Text | PM |
| H | Hour in day (0-23) | Number | 0 |
| k | Hour in day (1-24) | Number | 24 |
| K | Hour in am/pm (0-11) | Number | 0 |
| h | Hour in am/pm (1-12) | Number | 12 |
| m | Minute in hour | Number | 30 |
| s | Second in minute | Number | 55 |
| S | Millisecond | Number | 978 |
| z | Time zone | General time zone | Pacific Standard Time; PST; GMT-08:00 |
| Z | Time zone | RFC 822 time zone | -0800 |

**Examples**

|  |  |
| --- | --- |
| Format |  |
| "yyyy.MM.dd G 'at' HH:mm:ss z" | 2001.07.04 AD at 12:08:56 PDT |
| "yyyyy.MMMMM.dd GGG hh:mm aaa" | 02001.July.04 AD 12:08 PM |
| "EEE, d MMM yyyy HH:mm:ss Z" | Wed, 4 Jul 2001 12:08:56 -0700 |
| "yyMMddHHmmssZ" | 010704120856-0700 |
| "yyyy-MM-dd'T'HH:mm:ss.SSSZ" | 2001-07-04T12:08:56.235-0700 |
| "yyyy-MM-dd'T'HH:mm:ss.SSSXXX" | 2001-07-04T12:08:56.235-07:00 |
| "YYYY-'W'ww-u" | 2001-W27-3 |

# Extended Communication Template

The Extended Communication Template allows to embed tables in emails generated by Maximo communication templates. These tables can be populated using simple relationship from the main object.

## Configuration

Java Handlers

COMMLOG.TEMPLATEID - mxdev.common.commlog.FldCommLogTmpltId

ESCCOMMLOG.TEMPLATEID - mxdev.common.commlog.FldCommLogTmpltId

### System properties

Go to System Configuration > System Properties and create the following entries

|  |  |  |  |
| --- | --- | --- | --- |
| Property name | Description | Global value | Default value |
| mxdev.commtmpl.texttable.tag | Prefix for text table | %TEXTTABLE% | %TEXTTABLE% |
| mxdev.commtmpl.htmltable.tag | Prefix for HTML table | %HTMLTABLE% | %HTMLTABLE% |
| mxdev.commtmpl.htmltable.tbstyle | HTML style for TABLE |  |  |
| mxdev.commtmpl.htmltable.thstyle | HTML style for TH |  |  |
| mxdev.commtmpl.htmltable.tdstyle | HTML style for TD |  |  |
| mxdev.commtmpl.htmltable.trstyle | HTML style for TR |  |  |

Select Live Refresh to load current values.

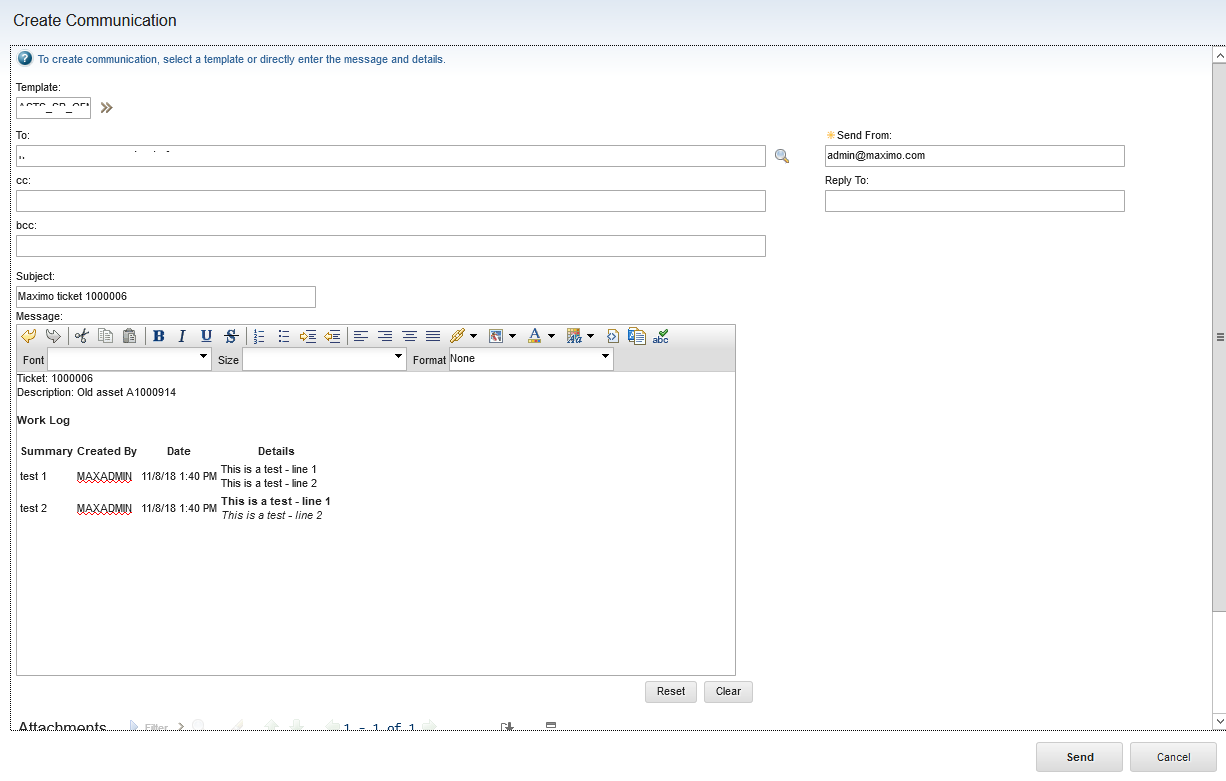
## How to

Child tables can be included in the communication template body using the following format:

* Starting tag – can be %HTMLTABLE% or %TEXTTABLE% depending if you want a well formatted table or a plain text one.
* Separator – is a ‘:’ character
* Relationship name – is the name of the relationship to the child object
* Field list – a list on attribute names (of the child object) separated by a ‘.’ Character.

For example, if you want to send an email that contains the information of a Work Order and you want to include the Work Log table you can set this text in the message

%HTMLTABLE%:UXWORKLOG.DESCRIPTION.CREATEBY.CREATEDATE.DESCRIPTION\_LONGDESCRIPTION

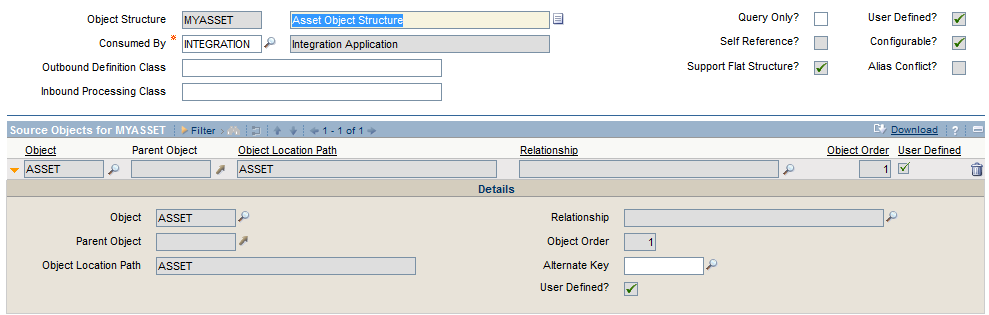


# CSV Exporter

In standard Maximo there is no way to schedule the invocation of an outbound interface in order to periodically export a CSV or XML file at a specific interval. This can be accomplished with a little custom Java class. I have used the Java code described in this post as a starting point to develop my own generic procedure. In this example I will show how periodically export in a CSV file all the assets that contains the word ‘pump’ in the description. The procedure can be adapted to periodically invoke any Publish Channel you like.

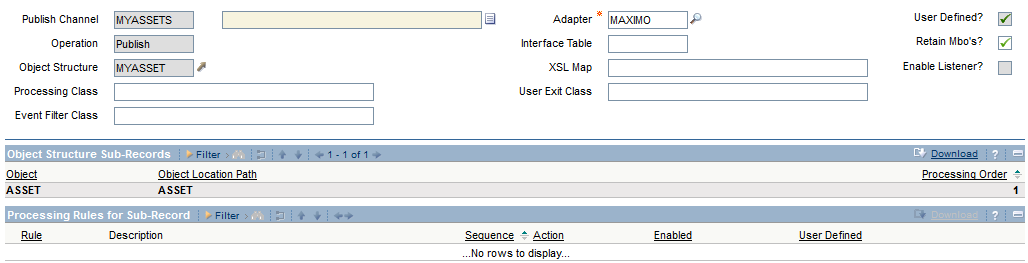
## Configuration

Create an Integration Object Structure MYASSET and add the ASSET object as source.

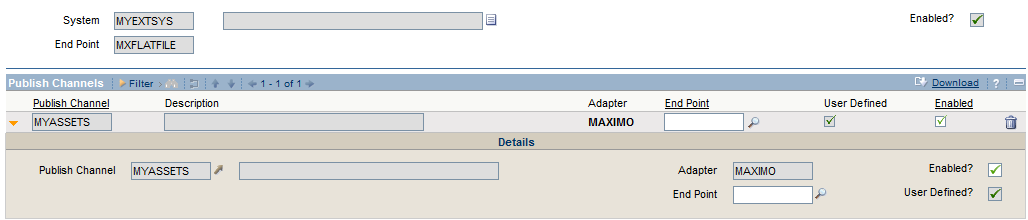


Remember to select ‘Support Flat Structure’ if you need CSV export.

Now create the Publish Channel MYASSETS.



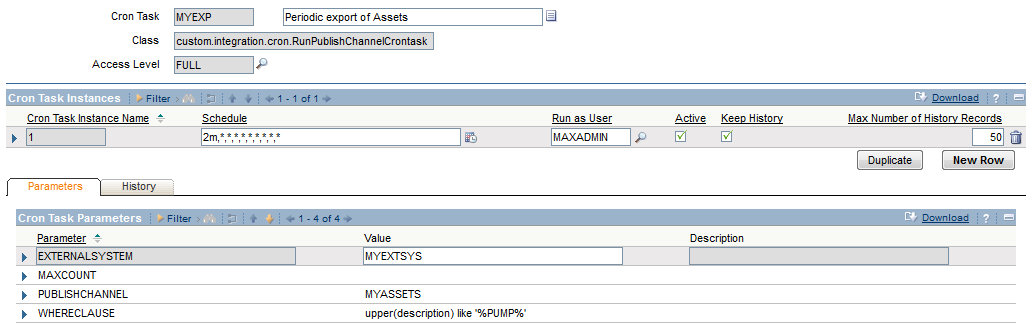
Create an External System (or reuse your own) and add the MYASSETS Publish Channel to it.



Enable publish channel and test that everything is ok using the ‘Data Export’ button. A CSV file should appear in few minutes in your MIF output folder (see mxe.int.globaldir system property). If you have problems check the JMS queues, JMSQSEQCONSUMER crontask and anything that may block the publish channel functionality.

### Cron Task

If everything is ok it’s now time to schedule the invocation of the Publish Channel. Go to System Configuration – Platform Configuration – Cron Task Setup and create the crontask instance setting mxdev.integration.cron.RunPublishChannelCrontask in the Class field.



Activate and reload the crontask instance with the Action – Reload Request command.

You should now see a new CSV file published every 2 minutes in the MIF output folder.