
Administrator GUIDE

MES DATA RETRIEVAL JMP ADD-IN

Version 2.0

2023/11/17

TABLE OF CONTENTS

TABLE OF CONTENTS	2
INTRODUCTION.....	2
SYSTEM CONFIGURATION REQUIREMENTS	2
CODE ARCHITECTURE	4
1.1 MAIN SCRIPT TO RUN THE ADD-IN	4
1.2 FOLDER CONFIG	4
1.3 FOLDER INCLUDE.....	6
1.3.1 ATTACHED SCRIPTS	7
1.3.2 RESSOURCES	8
1.4 FOLDER DOC	12
1.5 FOLDER DEPLOYMENT	13

INTRODUCTION

This add-in automates data extraction tasks from AspenTech IP.21 and Osisoft PI (Aveva) historians. This way, you can use JMP to diagnose manufacturing problems and monitor several tags daily and weekly.

This document describes administrative tasks required to manage the operation of the add-in. This document is intended for a technical audience, specifically the system administrator in charge of the add-in. This guide only explains administrative tasks for the add-in running. To learn how to get started using the add-in, or how to install the add-in, refer to the appropriate document: *MES Data retrieval - User Guide.pdf*

This document assumes that you have successfully installed the add-in.

SYSTEM CONFIGURATION REQUIREMENTS

The following configuration and modules are required to launch and run the add-in.

- JMP V17.1.0
- Aspen Manufacturing Suite (Aspen InfoPlus.21®) and its 64-bit SQLplus ODBC driver
- OSIssoft PI Client Tools (PI Process Book) and its PI OLEDB driver
- Hostname and access to the IP21/PI server of interest

- Be connected to your enterprise network or VPN

Please contact IT support in case you are missing the packages or configuration above mentioned.

CODE ARCHITECTURE

The source code of the add-in is divided into 4 folders and 1 main JSL script.

Nom	Modifié le	Type	Taille
.git	15/11/2023 17:27	Dossier de fichiers	
config	10/11/2023 16:45	Dossier de fichiers	
deployment	15/11/2023 11:14	Dossier de fichiers	
doc	16/11/2023 16:34	Dossier de fichiers	
include	14/11/2023 15:09	Dossier de fichiers	
.gitignore	10/11/2023 16:45	Fichier source Git I...	1 Ko
LAUNCH_APPLICATION.jsl	15/11/2023 17:24	JMP Script	36 Ko
README.md	13/11/2023 13:52	Fichier source Mar...	3 Ko

Figure 1 - Content of add-in

1.1 MAIN SCRIPT TO RUN THE ADD-IN

LAUNCH_APPLICATION.jsl is the script to launch the MES data retrieval add-in. This is the script that will source all JSL dependencies and other external resources, as well as launch the main interface.

1.2 FOLDER CONFIG

This folder contains configuration variables that are hard-coded in the add-in.

Ce PC > Documents > JMP-MES-connector > config	
Nom	Modifié le
config.jsl	15/11/2023 17:20
guardband_selection_recall.txt	14/11/2023 14:25
sizewindow.txt	31/10/2023 10:35

Figure 2 - Content of config folder in the add-in

- **Sizewindow.txt:** This text file is automatically generated and modified when the add-in is launched. It takes the user's screen size and saves it in this file. This is used to adjust the size of elements in the add-in according to screen size.
- **Guardband_selection_recall.txt:** In the same way, this file is generated and modified each time guardband is launched by the user. It saves the last settings saved in guardband for use via the Recall button of the guardband interface.
- **Config.jsl:** This jsl file is the one that defines hard-coded variables or default paths in the add-in such as the version N°, the path for the servers list, the limit of points extracted for IP21/PI...

```

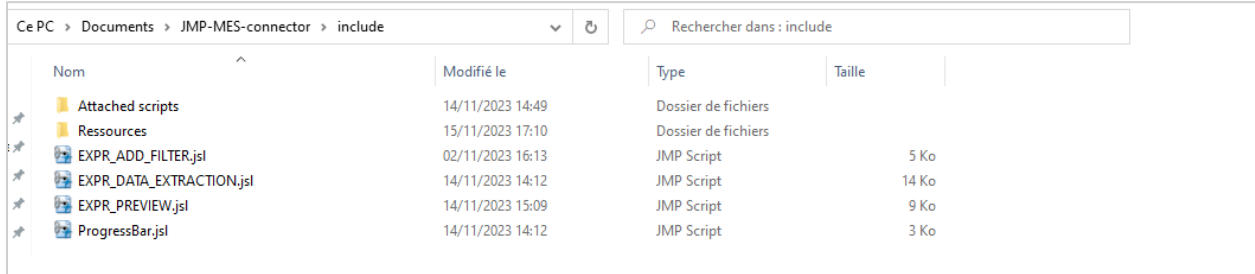
20
21 // Documentation link
22 strPathHelpDoc = strPath || "doc/MES Data retrieval - User Guide.docx" ;
23
24 // Guardband path
25 strPathGuardband = "$DOCUMENTS" || "\Guardband\";
26 str.path_guardband_recall = strPath || "config/guardband_selection_recall.txt";
27
28 // Workflow path
29 strWorkflowPath = "";
30
31 // Default Parameter
32 strTableVisibility = "invisible"; // Visibility of temporary tables
33 NRow_limit_IP21 = 1e6; // Max nb of rows per tag for IP21 data extraction
34 NRow_limit_PI = 1e6; // Max nb of rows per tag for PI data extraction
35 NRow_limit_preview = 1e6; // Max nb of rows per tag for preview
36 max_n_attempts = 10; // Maximal number of attempts
37
38 // Possible tag types
39 IP21Types = {"Analog", "Discrete", "Text"};
40 PITypes = {"R", "I", "D", "S"};
41 AllTypes = List();
42 Insert Into( AllTypes, IP21Types );
43 Insert Into( AllTypes, PITypes );
44 IP21NumTags = {"Analog", "AI"};
45 IP21TextTags = {"Discrete", "Text", "DI"};
46 PINumTags = {"R", "I"};
47 PITextTags = {"D", "S"};
48 PreviewTags = {"Discrete", "Text", "DI", "D", "S", "I"};
49
50 // Comparators
51 CompNum = {">", "<", ">=", "<="};
52
53 // Window Size
54 PathWindowSize = strPath || "config/sizewindow.txt";
55

```

Figure 3 - Content of Config.jsl

1.3 FOLDER INCLUDE

This folder contains all the JSL dependencies sourced by the main script, and required to run the add-in.

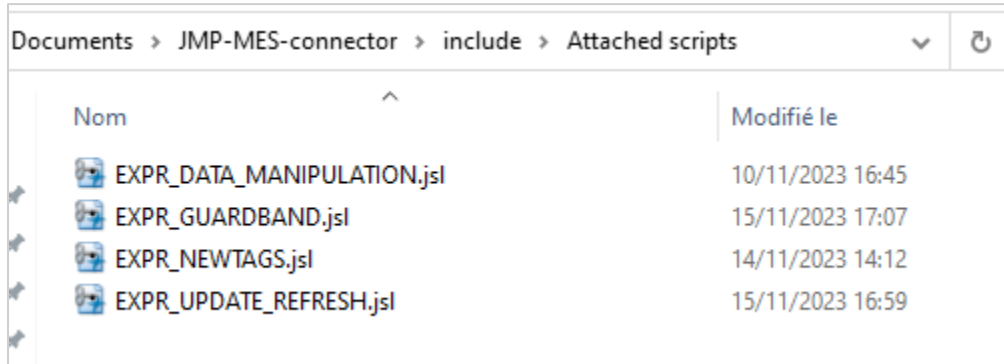


Nom	Modifié le	Type	Taille
Attached scripts	14/11/2023 14:49	Dossier de fichiers	
Ressources	15/11/2023 17:10	Dossier de fichiers	
EXPR_ADD_FILTER.jsl	02/11/2023 16:13	JMP Script	5 Ko
EXPR_DATA_EXTRACTION.jsl	14/11/2023 14:12	JMP Script	14 Ko
EXPR_PREVIEW.jsl	14/11/2023 15:09	JMP Script	9 Ko
ProgressBar.jsl	14/11/2023 14:12	JMP Script	3 Ko

- **EXPR_ADD_FILTER** : The script handles all the mechanics of the algorithm when the user presses the “Add filter” button in the main window.
- **EXPR_DATA_EXTRACTION**: The script handles all the mechanics of the algorithm when the user presses the "Run data extraction" button in the main window.
- **EXPR_PREVIEW**: The script handles all the mechanics of the algorithm when the user presses the "Preview" button in the main window in “Filter” tab.
- **ProgressBar**: The script generates the loading circle that is displayed when a data extraction is launched, allowing the user to see the status of the extraction.

1.3.1 ATTACHED SCRIPTS

This folder contains the 4 scripts attached to the extracted data table: UPDATE/REFRESH, ADD TAGS, ADDITIONAL TOOLS and GUARDBAND.

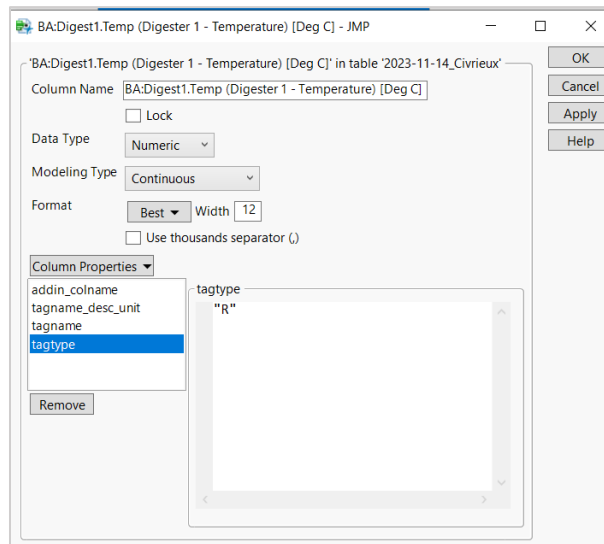


Nom	Modifié le
EXPR_DATA_MANIPULATION.jsl	10/11/2023 16:45
EXPR_GUARDBAND.jsl	15/11/2023 17:07
EXPR_NEWTAGS.jsl	14/11/2023 14:12
EXPR_UPDATE_REFRESH.jsl	15/11/2023 16:59

Figure 4 - Content of folder Include/Attached scripts

In these additional scripts, the algorithm will look for the existing tags in the table. The tags in the table are identified by the column properties:

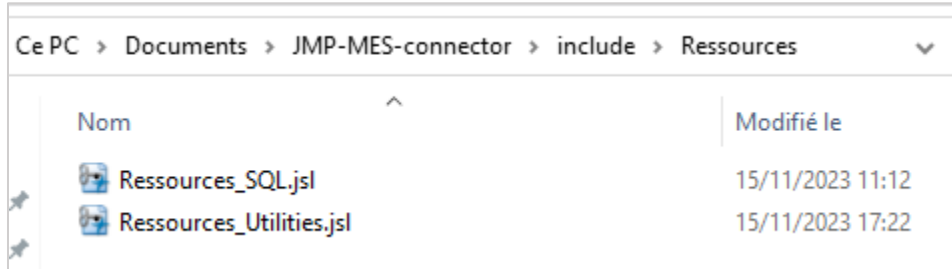
- **Addin_colname:** Name of the output column from the first data extraction.
- **Tagname_desc_unit:** Tag name the unit and the description
- **Tagname:** Tag name without the unit nor the description
- **Tagtype:** Type of the tag



It will also look for timestamp columns TS and TS_TC identified by the column property addin_colname "TS" and "TS.UTC".

1.3.2 RESSOURCES

This folder contains the resources, on which all other JSL dependencies and scripts depend.



Nom	Modifié le
Ressources_SQL.jsl	15/11/2023 11:12
Ressources_Uilities.jsl	15/11/2023 17:22

Figure 5 - Content of folder Include/Ressources

1.3.2.1 Ressources SQL

Ressources_SQL script contains all SQL query definitions and all functions that use these SQL queries. Queries are hard-coded, and the values to be replaced in the query are identified by the "VAR_" prefix. The SQL queries variables can be identified by the "__SQL" prefix, whereas the functions using these SQL queries can be identified by the "f_SQL_" prefix.

```

86  /* *****
87  * II. SQL QUERIES
88  * ***** */
89
90
91  //IP21 -----
92
93  // SQL Query to Find Tags in IP21
94  __SQL_FINDTAGS_IP21 =
95  "SELECT name as tagnames WIDTH 80,
96         name->ip_description as descriptions,
97         name->ip_eng_units as units,
98         name->ip_tag_type as type
99         FROM all_records
100        WHERE tagnames like '%VAR_TAGNAME%'
101        AND descriptions like '%VAR_DESCRIPTION%';";
102
103  // SQL Query to Extract Tags Data from IP21
104  __SQL_EXTRACTTAGS_IP21 =
105  "SELECT
106         T1.NAME WIDTH 80,
107         T1.TS,
108         TRIM ('Z' FROM ISO8601(T1.TS,0)) as TS_UTC,
109         CASE name->IP_TAG_TYPE WHEN 'Analog' THEN CAST(T1.VAR_VALUE as REAL) ELSE T1.VAR_VALUE End as VALUE,
110         STATUS,
111         T1.name->ip_description as DESCR,
112         T1.name->ip_eng_units as EU,
113         T1.name->ip_tag_type as TAGTYPE
114         FROM VAR_TABLE T1
115         VAR_FILTER
116         WHERE ((REQUEST = VAR_REQUEST)
117               AND (PERIOD = VAR_PERIOD)
118               AND T1.VALUE NOT IN ('???????'))
119               AND (NAME IN ('VAR_TAGNAME'))
120               AND ((T1.TS >= 'VAR_START_DATE')
121               AND (T1.TS <= 'VAR_END_DATE')));";
122
123
124  // SQL Query to Get Tag Type in IP21
125  __SQL_TAGTYPE_IP21 = "SELECT name->ip_tag_type as tagtype
126                      FROM all_records
127                      WHERE name = 'VAR_TAGNAME';";

```


1.3.2.1.1 SQL queries

For IP21 and PI servers, we define the following SQL queries:

- **__SQL_FIND_TAGS_[IP21/PI]** : Search tags query , used when the user clicks on the button “Find tag” in the main interface and in the “Add tags” additional scripts.
- **__SQL_EXTRACTTAGS_[IP21/PI]** : Tag extraction query, used when the user run any data extraction. It can be with the “Run data extraction” in the main interface, or in the additional scripts “Update/Refresh” and “Add tags”.
- **__SQL_TAGTYPE_[IP21/PI]** : Query to get the tag type, used when the user add a filter and the algorithm check the tag type. Also used in Preview functionality.
- **__SQL_DISTINCT_VALUES_PREVIEW_[IP21/PI]**: Query to get the distinct values for a given tag over a period AND over some filters. Used in Preview functionality.

1.3.2.1.2 SQL functions

For IP21 and PI servers, we define the following SQL functions:

- **f_SQL_FINDTAGS**: This function retrieves a list of available tags based on filtering criteria, including tag name, tag description. It uses the query **__SQL_FIND_TAGS_[IP21/PI]**.
- **f_SQL_GET_LOCALTIME**: This function retrieves the local server time for the specified server.
- **f_SQL_EXTRACT_TAGS**: This function is used to extract data from selected tags within a specified server and time range and with optional filters. It uses the query **__SQL_EXTRACTTAGS_[IP21/PI]**.
- **f_SQL_GET_TAGTYPE**: This function retrieves the tag type for a given tag name (without unit/description). It uses the query **__SQL_TAGTYPE_[IP21/PI]**.
- **f_SQL_PREVIEW**: This function is used in preview to retrieve distinct values for a tag given a specific time period and optional filters. It uses the query **__SQL_DISTINCT_VALUES_PREVIEW_[IP21/PI]**.

- **f_SQL_CREATE_ONE_FILTER:** This function is used to create the SQL General Structure (GS) for one filter, including the filter's comparison, value, and other parameters.

```

730 f_SQL_CREATE_ONE_FILTER = Function( {StartFormat, EndFormat, TagName, Tagtype, Comp, Value, Condition, ServerType},
731
732     // PI SQL syntax -----
733
734     // General Structure of 1 filter (GS)
735     __SQL_GENERAL_SYNTAX_PI =
736     "SELECT VAR_PREFIX1.TIME FROM [PIPOINT]..[PIPOINT] VAR_PREFIX2, [PIARCHIVE]..[VAR_TABLE] VAR_PREFIX1 WHERE VAR_PREFIX2.TAG = 'VAR_TAGNAME' A
737
738     // Filters for Text and Numeric values
739     __SQL_VARFILTER4TEXT_PI =
740     " ( CAST(VAR_PREFIX1.VALUE AS STRING) VAR_COMP VAR_VALUE OR DIGSTRING(CAST(VAR_PREFIX1.VALUE AS INT32)) VAR_COMP VAR_VALUE ) ";
741     __SQL_VARFILTER4NUM_PI = " VAR_PREFIX1.VALUE VAR_COMP VAR_VALUE ";
742
743     // IP21 SQL syntax -----
744
745     // General Structure of 1 filter (GS)
746     __SQL_GENERAL_SYNTAX_IP21 =
747     "SELECT VAR_PREFIX1.TS, CASE name->IP_TAG_TYPE WHEN 'Analog' THEN CAST(VAR_TABLE_VALUE as REAL) ELSE VAR_TABLE_VALUE End as VAR_PREFIX1VALUE
748
749     // Filters for Text and Numeric values
750     __SQL_VARFILTER4TEXT_IP21 = " VAR_PREFIX1VALUE VAR_COMP VAR_VALUE ";
751     __SQL_VARFILTER4NUM_IP21 = " VAR_PREFIX1.VALUE NOT IN ('?????') AND VAR_PREFIX1VALUE VAR_COMP VAR_VALUE ";
752

```

- **f_SQL_CREATE_ALL_FILTERS:** This function is used to concatenate all individual filters (built with f_SQL_CREATE_ONE_FILTER), along with the specified condition (OR or AND).

```

546 f_SQL_CREATE_ALL_FILTERS = Function( {StartFormat, EndFormat, Filters, Condition, ServerType},
547
548     // IP1 SQL syntax -----
549
550     // Extern structure with all GS
551     __SQL_EXT_STRUCT_UNIONIP21 = " INNER JOIN ( VAR_ALL_GS ) AS FILTERS ON T1.TS = FILTERS.TS ";
552     __SQL_EXT_STRUCT_JOINIP21 = " VAR_ALL_GS ";
553
554     // Intern structure with 1 GS
555     __SQL_INT_STRUCT_UNIONIP21 = " VAR_GS ";
556     __SQL_INT_STRUCT_JOINIP21 = " INNER JOIN ( VAR_GS ) AS VAR_PREFIX ON VAR_PREFIX.TS = T1.TS ";
557     __SQL_LINK_UNIONIP21 = " UNION ";
558     __SQL_LINK_JOINIP21 = " ";
559
560     // Structure in case of nested filters (GS1 and GS2)
561     __SQL_NESTED_UNIONIP21 =
562     " SELECT VAR_PREFIX1.TS,VAR_CGS1VALUE FROM ( ( VAR_GS1 ) AS VAR_PREFIX1 INNER JOIN ( VAR_GS2 ) AS VAR_PREFIX2 ON VAR_PREFIX1.TS = VAR_PREFIX2.TS ) ";
563     __SQL_NESTED_JOINIP21 = " VAR_GS1 UNION VAR_GS2 ";
564
565     // PI SQL syntax -----
566
567     // Extern structure with all GS
568     __SQL_EXT_STRUCT_UNIONPI = " INNER JOIN ( VAR_ALL_GS ) AS OR1 ON OR1.TIME = C1.TIME";
569     __SQL_EXT_STRUCT_JOINPI = " VAR_ALL_GS ";
570
571     // Intern structure with 1 GS
572     __SQL_INT_STRUCT_UNIONPI = " VAR_GS ";
573     __SQL_INT_STRUCT_JOINPI = " INNER JOIN ( VAR_GS ) VAR_PREFIX ON VAR_PREFIX.TIME = C1.TIME ";
574     __SQL_LINK_UNIONPI = " UNION ";
575     __SQL_LINK_JOINPI = " ";
576
577     // Structure in case of nested filters (GS1 and GS2)
578     __SQL_NESTED_UNIONPI =
579     " SELECT VAR_PREFIX1.TIME FROM ( VAR_GS1 ) AS VAR_PREFIX1 INNER JOIN ( VAR_GS2 ) VAR_PREFIX2 ON VAR_PREFIX1.TIME = VAR_PREFIX2.TIME ";
580     __SQL_NESTED_JOINPI = " VAR_GS1 UNION VAR_GS2 ";
581

```

1.3.2.2 Resources utilities

This script contains all the documented functions, identified by the prefix "f_", used and common to all main and additional scripts. In this script, functions can be categorized into different sections:

- I. **Functions for error messages:** These functions allow for the printing of log messages or pop-up window with warning/error messages.
- II. **Functions for text processing:** These functions are involved in every text processing such as for instance removing duplicates in a list, removing special characters, removing text/unit/description from tag name...
- III. **Functions for tag extraction and tag analysis:** These functions are involved in tag extraction and analysis, such as the tag loop extraction, the summary function
- IV. **Other functions**
- V. **Common buttons for attached scripts:**
 - **f_button.find_tags:** This function is associated with the "Find tag" button and is utilized across multiple scripts ("Main UI", "Add tags") to facilitate the search of new tags to add.
 - **f_button.update_filter:** This function is associated with the "Update Filter" button and is utilized across multiple scripts ("Update/Refresh", "Add tags") to facilitate the refinement and preview of data filtration.

1.4 FOLDER DOC

This folder contains all files external to the add-in (User guide, Administrator guide, ...).

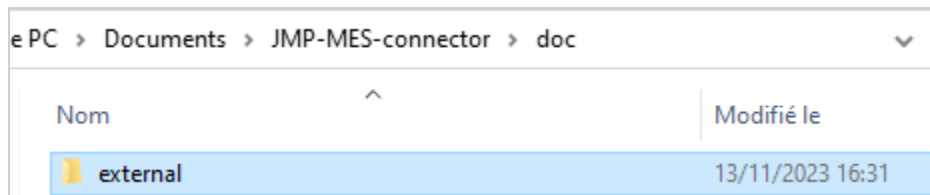


Figure 6 - Content of doc folder

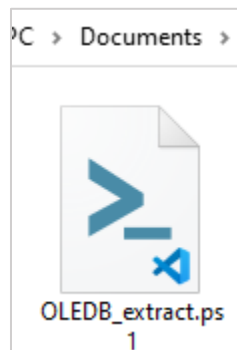


Figure 7 - Content of doc/external folder

1.5 FOLDER DEPLOYMENT

This folder contains the DEV and PROD add-ins that are sent to users after deployment.

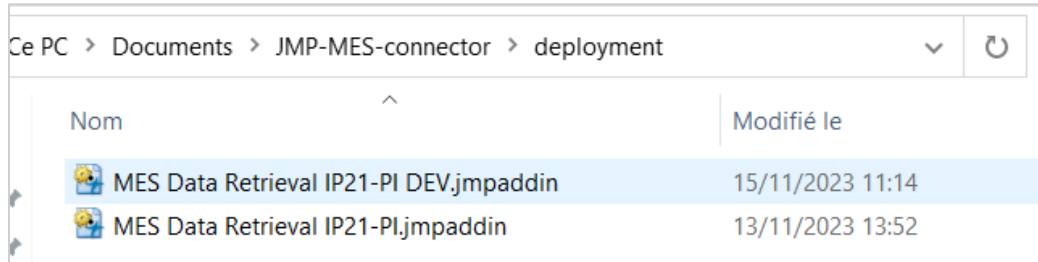


Figure 8 - Content of folder deployment

Proceed as follow to build and deploy the add-in:

1. Change the add-in path in LAUNCH_APPLICATION.jsl depending on whether you're in DEV or PROD.
 - DEV: ADDIN_HOME(MES.Data.Retrieval.DEV)
 - PROD: ADDIN_HOME(MES.Data.Retrieval)

```

43 // Add-in path (DEV)
44 strPath = Get Path Variable( "ADDIN_HOME(MES.Data.Retrieval.DEV)" );
45 //strPath = "C:/Users/CUVI5474/Documents/JMP-MES-connector/";
46
47 // Add-in path (PROD)
48 //strPath = Get Path Variable( "ADDIN_HOME(MES.Data.Retrieval)" );
49

```

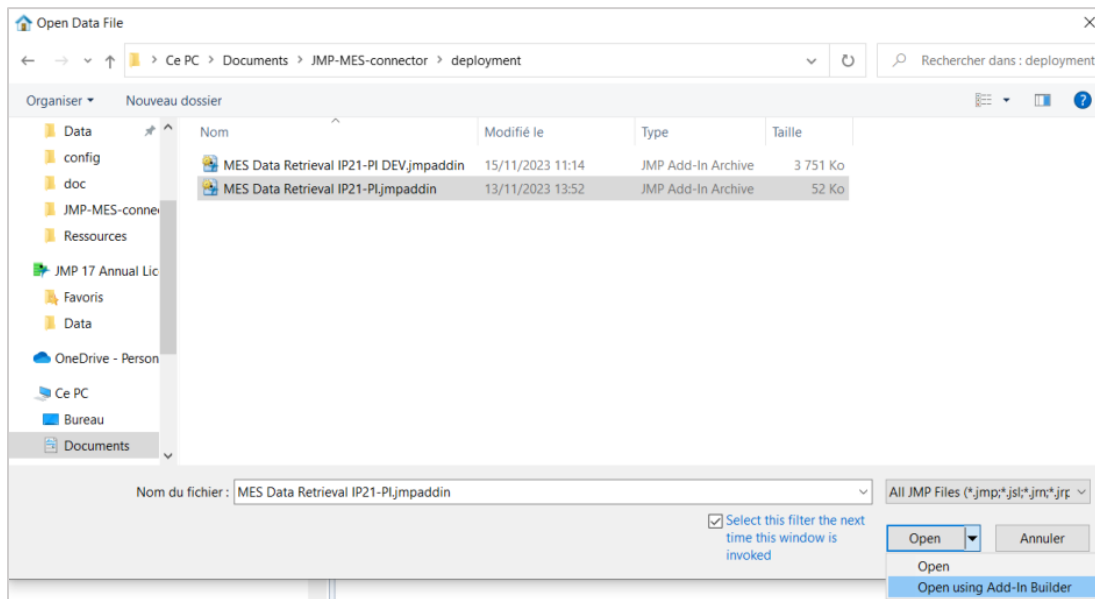
2. Increment the version N° in the config.jsl file. For DEV, the version N° is in general “(DEV)” plus the date of release in yyymmdd format.

```

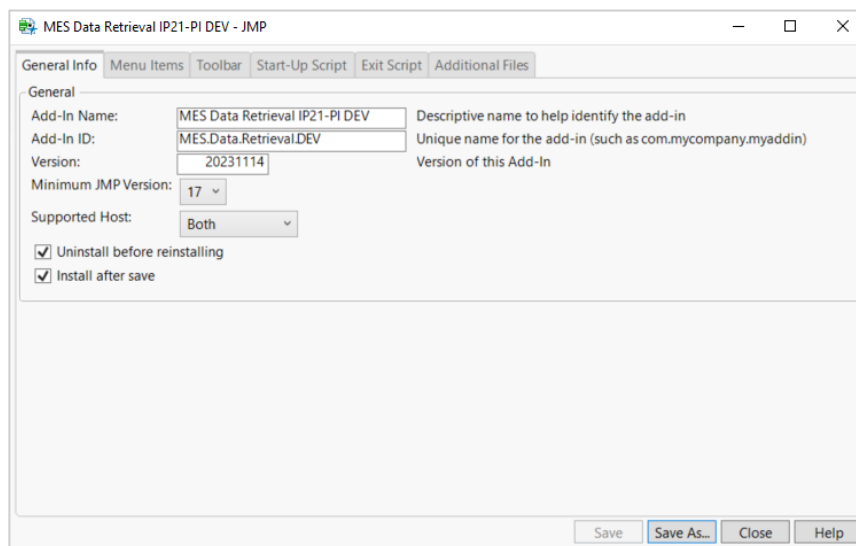
15 // Version
16 Version = "(DEV) 20231114";
17

```

3. Open using Add-in Builder the file MES Data Retrieval IP21-PI DEV.jmpaddin (DEV) or MES Data Retrieval IP21-PI.jmpaddin (PROD)



4. Increment the version of the add-in. In general for DEV, the version N° as the date of release in yyyyymmdd format, and for PROD it is for example 1.0, 1.1 ... Put also the minimum JMP versions.

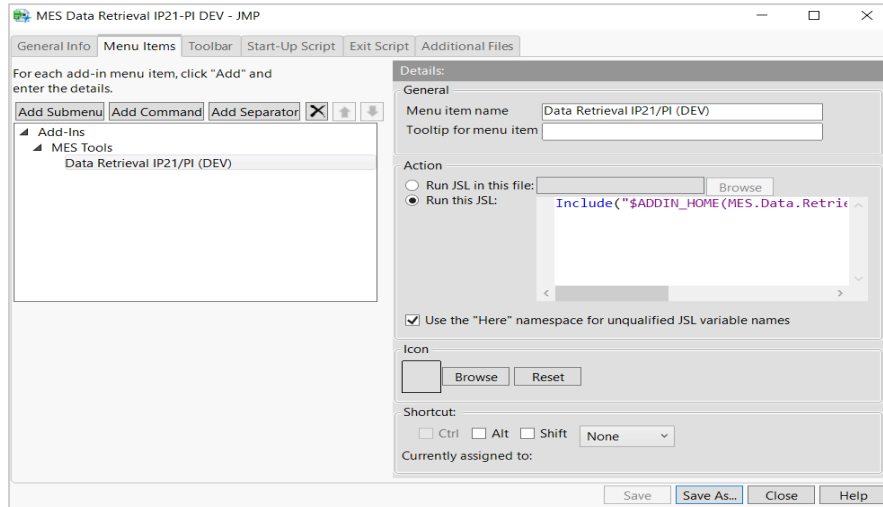


5. You can customize the way the add-in is displayed in JMP Menu in *Menu Items*, as well as its name. Check in *Action* the command *Run this JSL*, and add the following command:

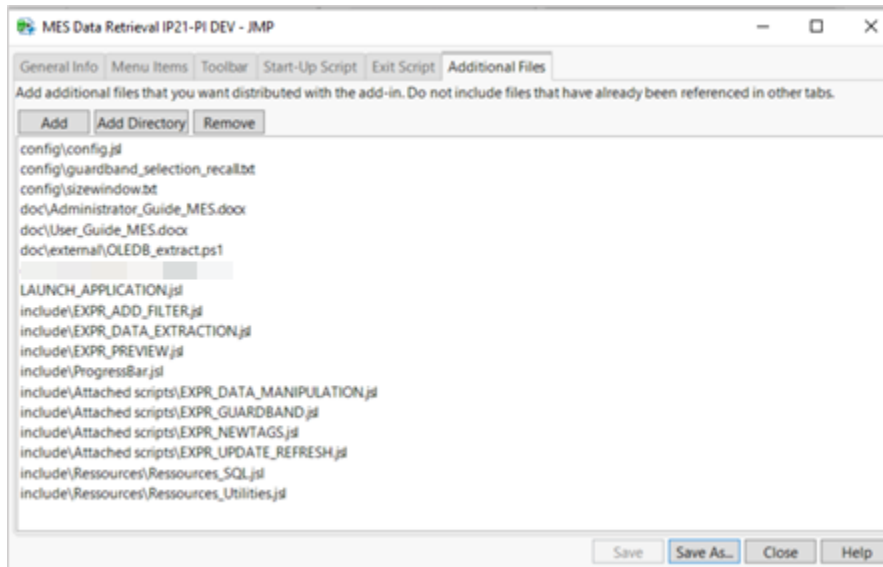
```
Include("$ADDIN_HOME(MES.Data.Retrieval)LAUNCH_APPLICATION.js1")
```

or

```
Include("$ADDIN_HOME(MES.Data.Retrieval.DEV)LAUNCH_APPLICATION.js1")
```



6. Go to *Additional Files* tab and remove all the files. We do this step because we want to remove the old files attached to the add-in and update them with the new ones (the ones with your modifications).



7. Then click on *Add Directory* and add the folder **include**, **doc** and **lib** were you have done your modifications (new version).
8. Click on *Add* and add the file **LAUNCH_APPLICATION.jsl**
9. Click on *Save*. It will generate or update the add-in *MES Data Retrieval IP21-PI DEV.jmpaddin* / *MES Data Retrieval IP21-PI.jmpaddin*.