D

Custom Script Integration

These topics describe how use a custom script and a customizable node to incorporate data from external sources into the Proteome Discoverer results file.

Contents

Scripting Node Overview

The Scripting Node is a customizable node that allows you to run an executable file as the final node in a workflow and to incorporate the result data into the application's result file.

The data exchange between the Proteome Discoverer application and the executable relies on a few mechanisms that do not need .NET code because the executable does not use the application API to read and write data. Instead, the user declares the columns for the tables from the result file to be exported into one or more tab-separated text files along with a JSON file describing the structure and location of those files. The user-defined script or executable should subsequently read these files, perform the user-defined type of analysis, and then optionally write out tab-separated text files containing information with another JSON file describing the structure of new tables and/or columns to be imported into the Proteome Discoverer results.

To create a script for the Scripting Node, programmers can use any programming language that supports running code from the command line, for example, python, R, C#, C++, and so on.

- Use as a Post Processing Node (Default)

For an example, see Figure 410.

- Register a standalone Processing or Consensus node that is preconfigured to an executable and can be given to another Proteome Discoverer user.

For information about how to register a custom workflow node, see Installing or Updating a Scripting Workflow Node.

The Scripting Node cannot node access MS/MS spectra or study information. In general, the Scripting Node can only act upon viewable columns from result file.

To integrate a search engine, a statistic algorithm, or a workflow that requires deeper access to the application code, you must program in C#. Contact pd.support@thermofisher.com for more information.

General Mechanism of the Scripting Node

The Scripting Node does the following:

1. Exports data (tables and columns) listed in the node parameter "Requested Tables and Columns" as one or more tab-separated text files.
2. Writes a node_args.json file, which the script uses to receive information from the application. The file contains essential information about the exported data, including location of the text files and the row IDs and columns for the exported tables.
3. Starts the executable plus an optional script as configured in the corresponding node parameter "Path to Executable" and "Command Line Arguments".
4. After the executable finishes, if the script produces a node_response.json file and new result tables, the node imports the results and integrates them into the result file.

Using the Scripting Node in a Workflow

This topic describes using a Scripting Node with an existing script. The script or executable performs the following:

- Read the node_args.json file
- Read the exported text files
- Depending on what the script does, write a node_response.json file when results are presented in the application

For information about modifying scripts to use with the Scripting Node, see "Integrating Your Script into a Post-Processing Scripting Node" on page 907.

Primarily, the Scripting Node is used as a post-processing node in the consensus workflow. Depending on the Scripting Node's function, you can add it to a processing workflow or a consensus workflow. (The application's default workflows do not include the Scripting Node.)

❖ To use a predefined script in Proteome Discoverer

1. If the latest release of the language that the script was written in is not already installed on the computer running the Proteome Discoverer application, install it.

   Many of this languages are available for download, for example:

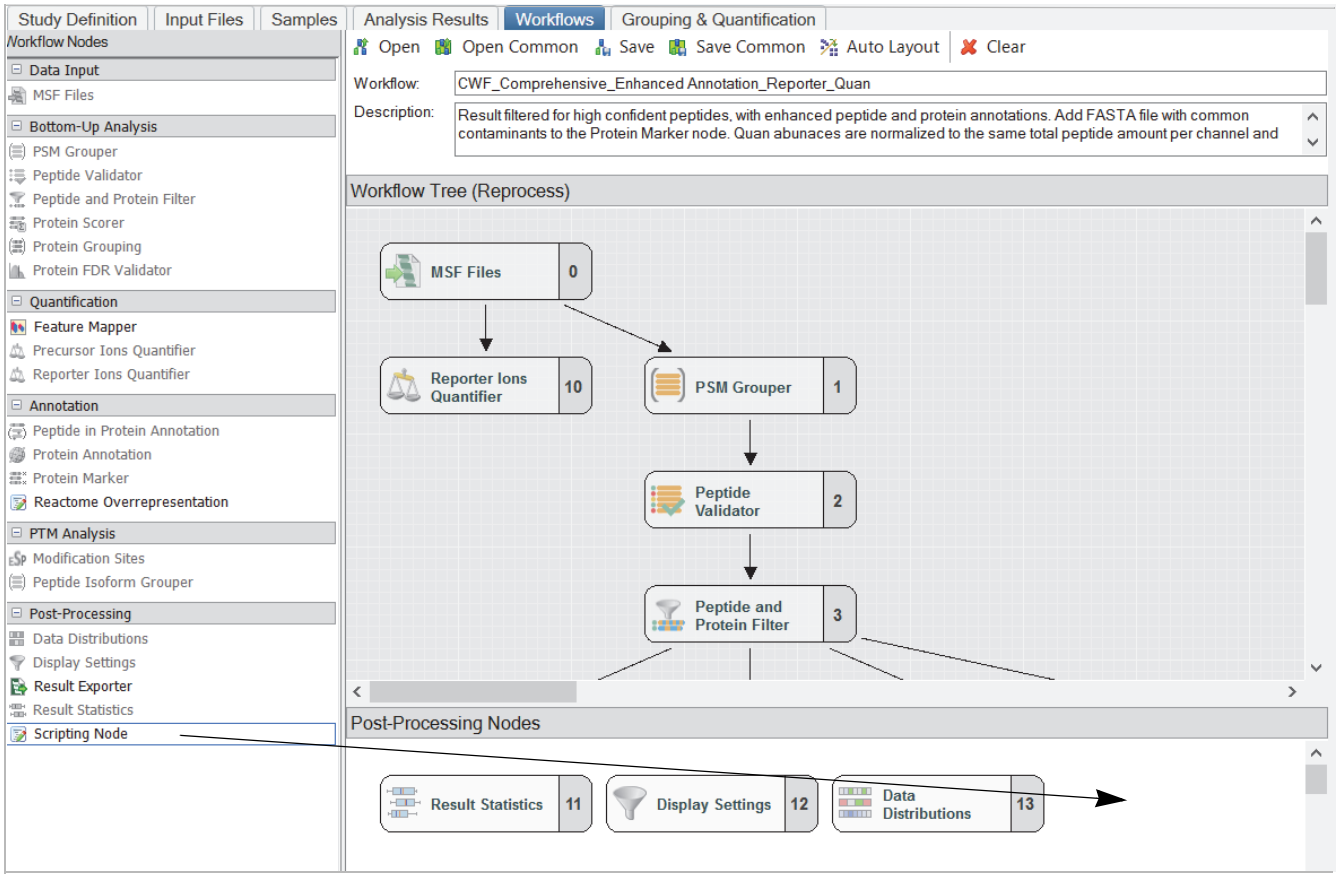
   - For R, navigate to https://www.r-project.org
   - For Python, navigate to https://www.python.org

2. Do one of the following in the Proteome Discoverer application:

   - Choose **Window > Workflow Editor**.
   - Within a study, click the **Workflows** tab.

3. In the Post-Processing section of the Workflow Nodes pane, left-click the **Scripting Node**, and drag it into the Post-Processing Nodes space of the Workflow Tree pane (Figure 410).

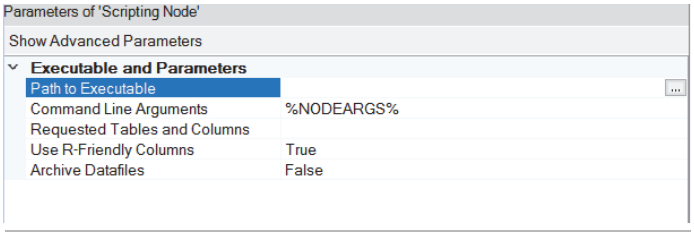
Figure 410. Adding the Scripting Node to a Post-Processing Workflow



4. Click the **Parameters of 'Scripting Node'** tab.

   The Scripting Node parameters appear (Figure 411).

Figure 411. Default Scripting Node parameters



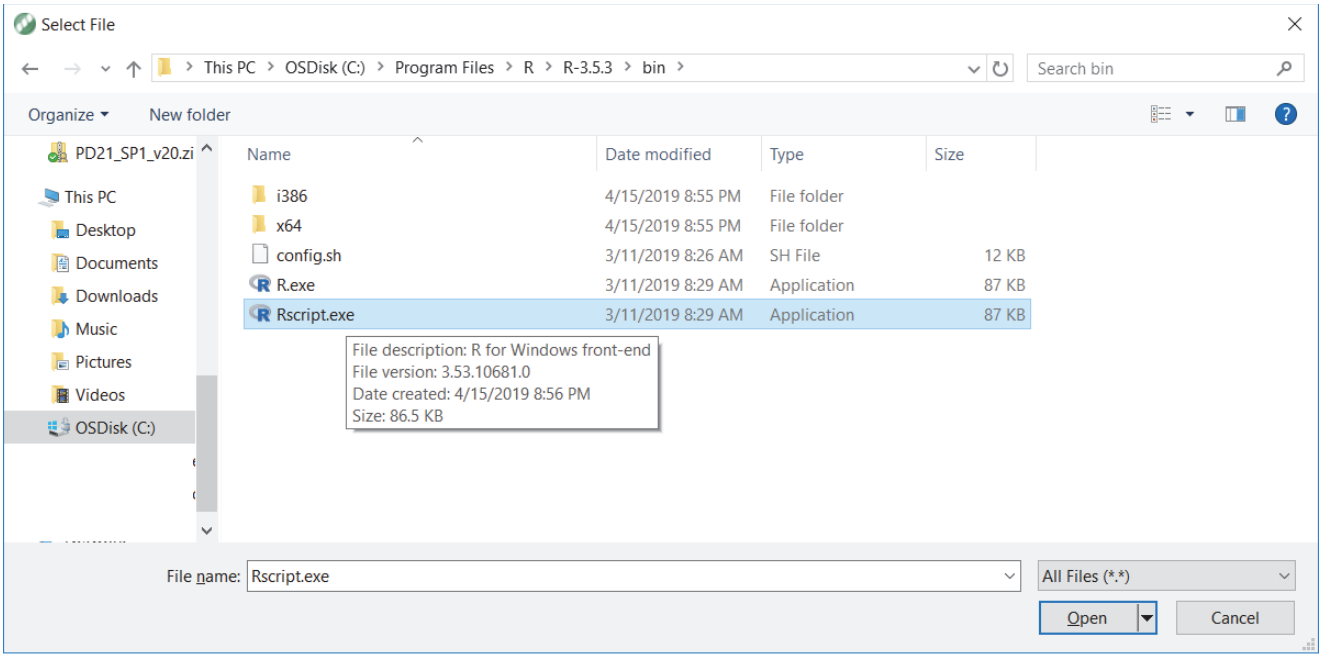
5. For the Path to Executable, enter the path to run the script from the command line.

   For example:

   - For RScript.exe, the default location is the `c:\Program Files\R\R-`*`version`*`\bin`.
   - For Python.exe, the default location is `c:\Python`*`version`*.

   Or click the **Browse** button, and select the executable

Figure 412. Select File



6. For the Command Line Arguments parameter, type the full path of the script executed followed by "%NODEARGS%".

   The Scripting Node replaces the %NODEARGS% text with the full path to the node_args.json file.

   You can send additional arguments to the script. Add them after the "%NODEARGS% argument in space-separated format. The Scripting Node will pass the additional arguments to the external executable, and it will be up to the script to recognize them.

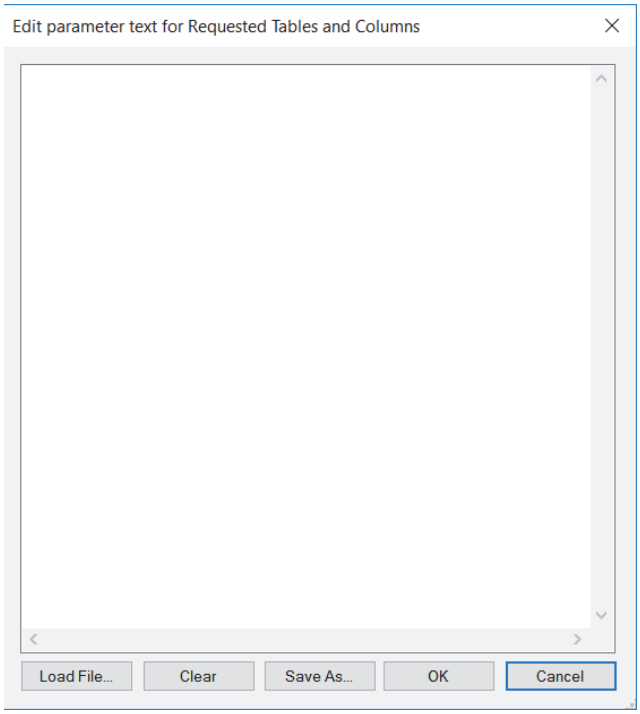
For information on the contents and location of this file, read "Integrating Your Script into a Post-Processing Scripting Node" on page 907.

7. For the Requested Tables and Columns parameter, click the **Browse** button.

   The Requested Tables and Columns dialog opens.

   The Requested Tables and Columns parameter sends the results from the Proteome Discoverer analysis to the script. Refer to any instructions provided with the script regarding the specific tables and the associated columns to select.

   Figure 413. Requested Table and Columns dialog box



8. Do one of the following:

   a. Click Load File to open Windows Explorer select a

   b. Enter the tables and the columns into the dialog box or by loading a file that contains them.

   Use the following format:

   - Follow each table by a ":" and a comma-separated list of column names.

     (Spaces after commas are supported.)

   - To use more than one table name, use the ";" separator between the last column name from the previous table and the next table name.

   - To export all of the columns from a table, use the table name without the ":" symbol or column headers.

```
Table 1:Column 1, Column 2, Column 3; Table 2:Column 1, Column 2,
Column 3; Table 3
```

The following example contains four columns from the Proteins table, 1 column from the Peptide Groups table, and all columns from the PSMs table for input into the script.

```
Proteins:Accession,Description,Adjusted Ratio P-Value,Abundance
Ratio; Peptide Groups:Sequence,Abundance Ratio P-Value;PSMs
```

Columns such as Abundance Ratio P-Value might contain any number of sub-columns depending on how many ratios were selected for a given analysis. The Scripting Node sends all sub-columns to the script. (Selecting a subset of sub-columns is not supported.)

Note The order of tables and columns is likely to be important for the script.

Note Any typographic errors in the names of the tables or columns results in failing to send that column to the script.

See Table Name and Column Name Listing, for a listing. Also, see the media's Scripting Node folder for an Excel file that contains this information.

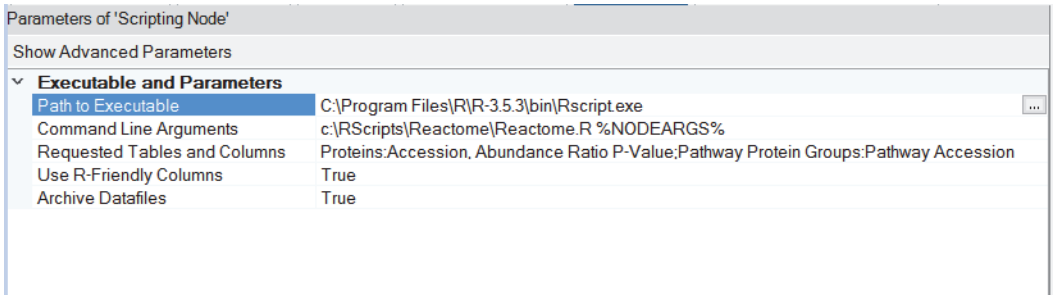
9. For the Use R-Friendly Columns parameter, choose **True**.

   The parameter specifies whether to convert symbols such as # (as in # of Unique Peptides), % (as in % Sequence Coverage), and separators such as parenthesis or brackets. This setting allows the downstream scripting languages that use symbols for programming tasks to correctly parse the text.

10. For the Archive Datafiles parameter, choose **True**.

    This parameter is useful for debugging scripts and for exporting results from the application. The option creates a zip file in the study folder. The zip file contains the text files for each table sent to the script as well as the JSON files used to send information between the Scripting Node and the script (Figure 414).

Figure 414. Example Scripting Node parameters



Integrating Your Script into a Post-Processing Scripting Node

Use the following approach when integrating a new script into the Scripting Node:

1. Create a new empty script.

   The script contents depend on what you want it to do. For practice, the media's Scripting Node folder contains scripts that you can use as templates.

2. Use an example dataset with the type of result for which the script will be used.

   a. Add the Scripting Node to the workflow, and make sure that the Archive Datafiles parameter is True.

      This setting will export all of the files required by the script.

3. Modify the custom script to accept the node_args.json file.

   This information is detailed in Reading the node_args.json File into the Script.

4. Load the text files containing the Proteome Discoverer result information into the script.

5. If writing data back into the Proteome Discoverer application do the following:

   a. Create the output text files.

   b. Create the connection tables (if necessary).

   c. Create the node_response.json file.

      This information is detailed in Writing Results as New Columns to an Existing Table.

6. Test the script within the application.

   Testing is an iterative process. The primary challenge is to correctly format the output text files and the node_response.json files so that the Scripting Node can recognize the results.

   If the node_response.json file is not recognized (and results are not presented in the application), compare it to the node_args.json file to make sure the structure is the same.

Reading the node_args.json File into the Script

To pass data to the script, you must use the %NODEARGS% text in the Command Line Arguments parameter of the Scripting Node. The Scripting Node converts the %NODEARGS% text to the full path in a temporary folder in the Proteome Discoverer scratch folder, which contains a node_args.json file.

This node_args.json file contains information about:

- The names of the text files that contain the exported tables and columns chosen in the Selected Tables and Columns parameter mentioned above.
- Each column from each table exported from the application.

Figure 415. Example node_args.json file


```
{
  "CurrentWorkflowID": 73,
  "ExpectedResponsePath": "C:/ProgramData/Thermo/Proteome Discoverer
2.4/Scratch/Job73/Script(16)/node_response.json",
  "Tables": [
    {
      "TableName": "Proteins",
      "DataFile": "C:/ProgramData/Thermo/Proteome Discoverer
2.4/Scratch/Job73/Script(16)/TargetProtein.txt",
      "DataFormat": "CSV",
      "Options": {},
      "ColumnDescriptions": [
        {
          "ColumnName": "Proteins Unique Sequence ID",
          "ID": "ID",
          "DataType": "Long",
          "Options": {}
        },
        {
          "ColumnName": "Accession",
          "ID": "",
          "DataType": "String",
          "Options": {}
        },
        {
          "ColumnName": "Abundance Ratio P-Value Insulin  Control",
          "ID": "",
          "DataType": "Float",
          "Options": {
            "DataGroupName": "AbundanceRatioPValue"
          }
        },
        {
          "ColumnName": "Abundance Ratio P-Value IGF-1  Control",
          "ID": "",
          "DataType": "Float",
          "Options": {
            "DataGroupName": "AbundanceRatioPValue"
          }
        }
      ]
    },
    {
      "TableName": "Pathway Protein Groups",
      "DataFile": "C:/ProgramData/Thermo/Proteome Discoverer
2.4/Scratch/Job73/Script(16)/PathwayProteinGroup.txt",
      "DataFormat": "CSV",
      "Options": {},
      "ColumnDescriptions": [
        {
          "ColumnName": "Pathway Protein Groups Group ID",
          "ID": "ID",
          "DataType": "Int",
          "Options": {}
        },
        {
          "ColumnName": "Pathway Accession",
          "ID": "",
          "DataType": "String",
          "Options": {}
        }
      ]
    },
    {
      "TableName": "TargetProtein-PathwayProteinGroup",
      "DataFile": "C:/ProgramData/Thermo/Proteome Discoverer
2.4/Scratch/Job73/Script(16)/TargetProtein-PathwayProteinGroup.txt",
      "DataFormat": "CSVConnectionTable",
      "Options": {
        "FirstTable": "Proteins",
        "SecondTable": "Pathway Protein Groups"
      },
      "ColumnDescriptions": [
        {
          "ColumnName": "Proteins Unique Sequence ID",
          "ID": "ID",
          "DataType": "Long",
          "Options": {}
        },
        {
          "ColumnName": "Pathway Protein Groups Group ID",
          "ID": "ID",
          "DataType": "Int",
          "Options": {}
        }
      ]
    }
  ]
`
```

The node_args.json File Parameter Descriptions

This topic describes the values and data types in the node_args.json file.

- `CurrentWorkflowID` specifies the ID of the current workflow. This value is needed if you want to store new tables with a workflow ID column. Use in nodes that must be registered.
- `ExpectedResponsePath`, a key-value pair, contains the filename of the node_response.json file that must be produced by the script to declare new information that should be imported back into the Proteome Discoverer results. For the correct format of the node_response.json file, see Figure 419.
- `Tables`, an array, contains descriptions of the tables that Proteome Discoverer provides as .txt files to the executable. Each entry consists of the key-value pair:
  - `TableName` showing the table display name
  - `DataFormat` of the data (denoted here as CSV, but actually the table format is a tab-separated .txt file.)
  - `DataFile` showing full path of the .txt file that contains the exported columns from the given application table.
  - `ColumnDescriptions`, an array of objects, that specify the columns exported in the file. Each column description contains:
    - `ColumnName`: The name of the exported column.
    - `ID`: A flag that indicates whether the column is an ID.
      - `ID` if it is a usual ID column,
      - `WorkflowID` when the column is a workflow ID
      - `Other` when it is another ID column type.
    - `DataType`: specifies the value type of the column. Possible data types are:
      - `String`: text
      - `Int`: integer
      - `Long`: a long integer
      - `Float`: floating point
      - `Boolean`: `true` or `false`

  The value domains of all types include the empty data string, which is a valid data value for any of these types. The column descriptions also contain all ID columns contained in the table regardless of whether they were requested. The ID Columns also include the table name in their name to be consistent with exporting connections.

Guidelines for R Developers

Consider the following:

- Read the ID column into the R script as a string, especially for the ID values from the Proteins table. (As shown in earlier, the ID columns exported from any table in Proteome Discoverer are exported with quotes around them.) If the columns are in a numeric value, it can lead to inaccurately importing them to the script. Inaccurately importing columns into the script has downstream consequences when to importing results back into the application.
- Use the rjsonio library to import the JSON files into the script. The two other JSON libraries available at the time of software release, jsonlite and rjson, either modify the structure of the JSON object or add extraneous characters. These libraries produce node_response.json files that cannot be recognized by the application without custom modification,

  When using the rjsonio library, make sure that the format of the node_response.json file matches that of the node_args.json file. When using this library, the following code snippet reads the node_args.json file and writes out the JSON structure to node_response.json:

```
library(RJSONIO)

fileConn <- file(inputFile)# inputFile is the full path of the
node_args.json file

nodeArgs <- readLines(fileConn)

nodeArgs <-paste(nodeArgs, collapse = "\n")

nodeArgs <- gsub("\u00A0", "", nodeArgs)

close(fileConn)

PD_json_in <- fromJSON(nodeArgs, simplify=F)



responseJSON <- toJSON(PD_json_out, indent=3)

jsonfileconn <- file(jsonOutFile)# jsonOutFile is the full path of the
node_response.json file.  This needs to be in the same folder as
node_args.json

writeLines(responseJSON, jsonfileconn)

close (jsonfileconn)
```

For a suitable example template script, which follows these and other suggested guidelines, refer to the ProteinStartsWith.R script in the application media.

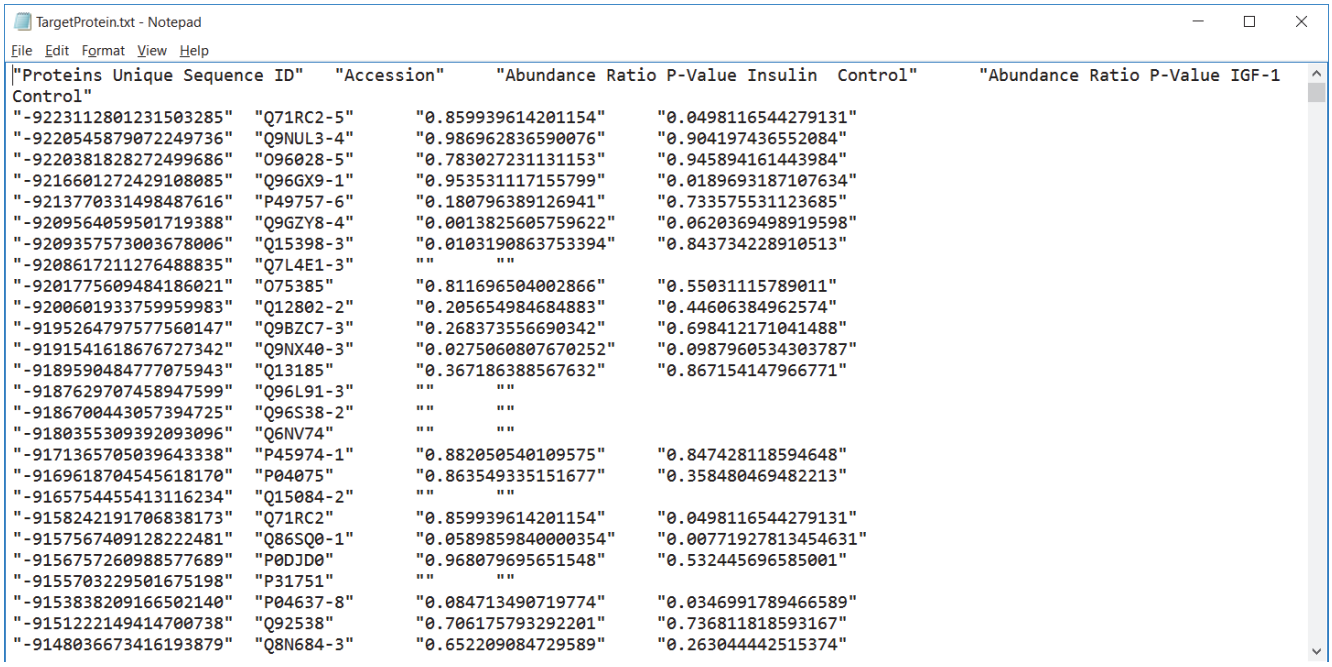
Reading the Exported Tables and Columns into the Script

Each table specified in the Selected Tables and Columns parameter in the scripting node produces a separate text file containing an ID column plus the list of columns specified in the node. The text file's name is similar to the name of the table shown in the application. The text file is tab-separated with each entry encapsulated with quotes. Figure 416 and Figure 417 show two tables produced with a Scripting Node exporting results from the Proteins and Pathway Protein Groups tables.

For any Distribution Map column, like the Abundance Ratio P-Value as shown in Figure 416, all of the sub-columns are exported. As a result, the number of columns that are exported to the text file for distribution map columns depends on the number of ratios or samples in any given dataset. The title of each column is the column name + subcolumn name.

The column header for Abundance Ratio P-value does not have the "/" character due to setting the "Use R-Friendly Headers" parameter to True in the Scripting Node.

Figure 416. Example of exported text file containing the ID column plus the selected columns


TargetProtein.txt - Notepad

File Edit Format View Help

```
"Proteins Unique Sequence ID"   "Accession"     "Abundance Ratio P-Value Insulin  Control"      "Abundance Ratio P-Value IGF-1
Control"
"-9223112801231503285"  "Q71RC2-5"      "0.859939614201154"     "0.0498116544279131"
"-9220545879072249736"  "Q9NUL3-4"      "0.986962836590076"     "0.904197436552084"
"-9220381828272499686"  "O96028-5"      "0.783027231131153"     "0.945894161443984"
"-9216601272429108085"  "Q96GX9-1"      "0.953531117155799"     "0.0189693187107634"
"-9213770331498487616"  "P49757-6"      "0.180796389126941"     "0.733575531123685"
"-9209564059501719388"  "Q9GZY8-4"      "0.0013825605759622"    "0.0620369498919598"
"-9209357573003678006"  "Q15398-3"      "0.0103190863753394"    "0.843734228910513"
"-9208617211276488835"  "Q7L4E1-3"      ""      ""
"-9201775609484186021"  "O75385"        "0.811696504002866"     "0.55031115789011"
"-9200601933759959983"  "Q12802-2"      "0.205654984684883"     "0.44606384962574"
"-9195264797577560147"  "Q9BZC7-3"      "0.268373556690342"     "0.698412171041488"
"-9191541618676727342"  "Q9NX40-3"      "0.0275060807670252"    "0.0987960534303787"
"-9189590484777075943"  "Q13185"        "0.367186388567632"     "0.867154147966771"
"-9187629707458947599"  "Q96L91-3"      ""      ""
"-9186700443057394725"  "Q96S38-2"      ""      ""
"-9180355309392093096"  "Q6NV74"        ""      ""
"-9171365705039643338"  "P45974-1"      "0.882050540109575"     "0.847428118594648"
"-9169618704545618170"  "P04075"        "0.863549335151677"     "0.358480469482213"
"-9165754455413116234"  "Q15084-2"      ""      ""
"-9158242191706838173"  "Q71RC2"        "0.859939614201154"     "0.0498116544279131"
"-9157567409128222481"  "Q86SQ0-1"      "0.0589859840000354"    "0.00771927813454631"
"-9156757260988577689"  "P0DJD0"        "0.968079695651548"     "0.532445696585001"
"-9155703229501675198"  "P31751"        ""      ""
"-9153838209166502140"  "P04637-8"      "0.084713490719774"     "0.0346991789466589"
"-9151222149414700738"  "Q92538"        "0.706175793292201"     "0.736811818593167"
"-9148036673416193879"  "Q8N684-3"      "0.652209084729589"     "0.263044442515374"
```

When the Scripting Node exports multiple tables, it exports additional files that map the associations between the tables. For example, when the Scripting Node exports the Proteins and Pathway Protein groups tables (Figure 417), it exports a second connection table that connects the proteins with their associated pathways by their ID values.

Figure 417. Example of Pathway Protein Groups export including the column selected in the Scripting Node parameter plus an ID

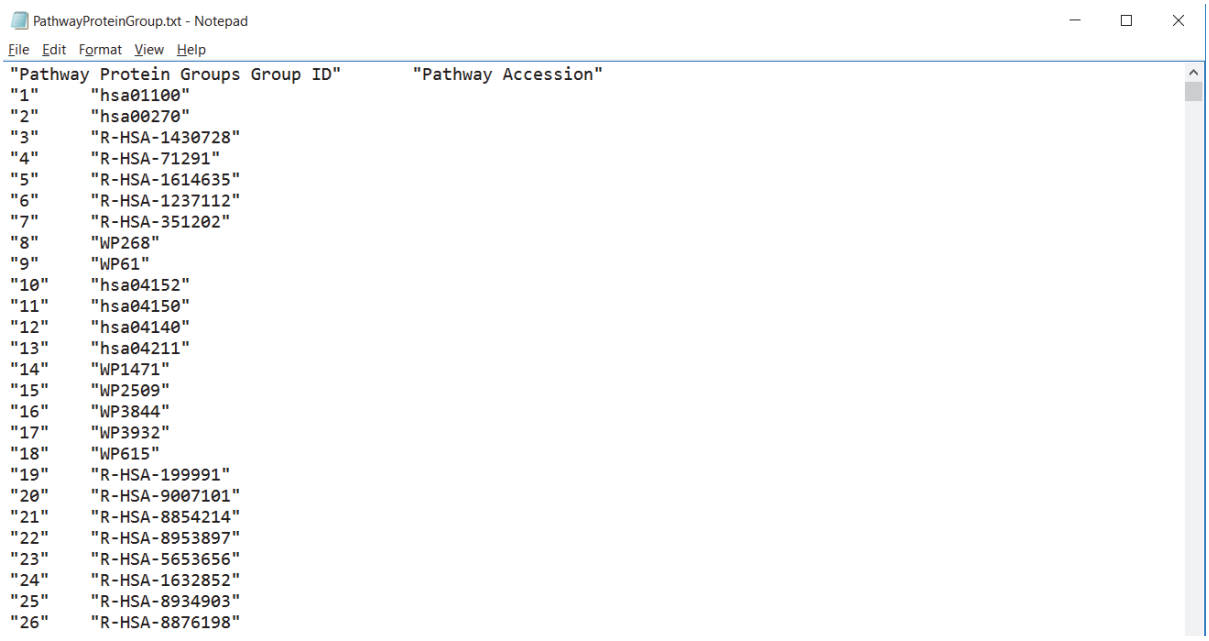
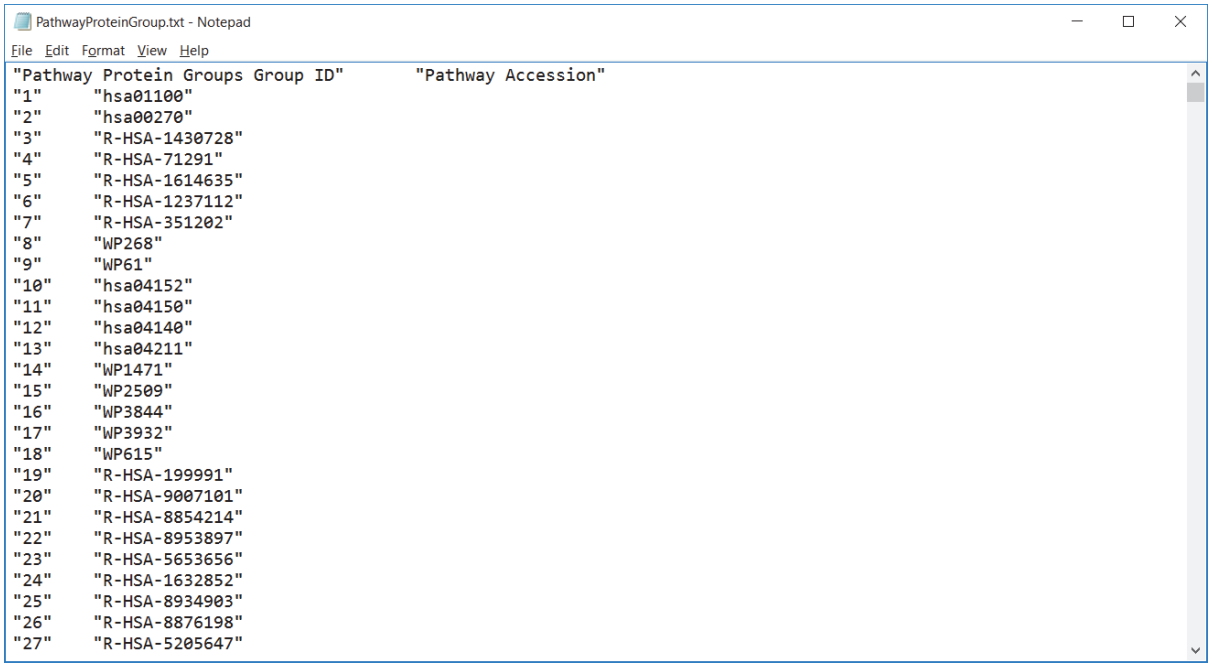


Table 418 shows an example connection table that maps the associations between the Proteins and Pathway Protein Groups. The name of this connection table will include the two tables involved with the association. A connection table will be produced between each pair of tables selected in the Scripting Node parameters. As the number of tables selected for export in the Scripting Node increases, the number of connection tables will also increase accordingly.

Figure 418. Example of Pathway Protein Groups export including the column selected in the Scripting Node parameter plus an ID



The node parameter, Use R-friendly Headers, specifies whether the column names are the same as if they were generated by an ordinary text export using that option.

Guidelines for R Developers

Consider the following guidelines:

- Read the ID column into the R script as a string, especially for the ID values from the Proteins table. (As seen in earlier figures (for example, Figure 418) the ID columns exported from any table in Proteome Discoverer are exported with quotes around them.) The scripting language might unexpectedly modify IDs on import when reading them as a numeric value, which will lead to errors when importing results back into the application.
- Use the following options when using `read.table` to import the .csv files:
  - `header = TRUE`
  - `check.names = FALSE`
  - `colClasses = character` (This keeps the ID columns intact as described previously.)

Writing Results as New Columns to an Existing Table

If you want to write results back for display in the application, you must perform the tasks described in this section.

The main tasks are:

1. Create a node_response.json file that tells the application which new columns or tables to be created.
2. Create a new text file for each table in Proteome Discoverer to be modified or created.
3. If multiple tables have associations, create the connection table for those associations.
4. Write the node_response.json file to the same temporary folder as the node_args.json and exported text files for the Scripting Node to use.

   The structure of the response file is the same as the node_args.json file, but the fields CurrentWorkflowID and ExpectedResponseDescription are ignored.

   The script calculates the results of the new columns, exports these new columns as a tab-separated text file, and annotates the node_response.json file to tell the application to expect the new columns. Then the Scripting Node imports the columns back into the application result file for presentation.

This example node_response.json file produces two new columns in the Pathway Protein Groups table (Figure 419).

Figure 419. Example node_response.json File


```
{
 "CurrentWorkflowID": 73,
 "ExpectedResponsePath":"C:/ProgramData/Thermo/Proteome Discoverer
2.4/Scratch/Job73/Script(16)/node_response.json",
 "Tables":[
  {
   "TableName":"Pathway Protein Groups",
   "DataFile":"C:/ProgramData/Thermo/Proteome Discoverer
2.4/Scratch/Job73/Script(16)/PathwayProteinGroup.out.txt",
   "DataFormat":"CSV",
   "Options":{ },
   "ColumnDescriptions":[
    {
     "ColumnName":"Pathway Protein Groups Group ID",
     "ID":"ID",
     "DataType":"Int",
     "Options":{ }
    },
    {
     "ColumnName":"Pathway Accession",
     "ID":"",
     "DataType":"String",
     "Options":{ }
    },
    {
     "ColumnName":"-10LogPValue Insulin  Control",
     "ID":"",
     "DataType":"Float",
     "Options":{ }
    },
    {
     "ColumnName":"-10LogPValue IGF-1  Control",
     "ID":"",
     "DataType":"Float",
     "Options":{ }
    }
   ]
  }
 ]
}
```


Two new columns named "`-10LogPValue Insulin Control`" and "`-10LogPValue IGF-1 Control`" are created in the Pathway Protein Groups table. The ID columns were kept from the original exported .txt file for the PathwayProteinGroup.txt file.

The corresponding data file PathwayProteinGroup.out.txt includes the new columns with the values calculated by the script. This example output file from the script has the two additional columns to import back into the application through the Scripting Node. The empty columns in the application result have `NA` as the entry (Figure 420).

Figure 420. Example output file from script with the two additional columns

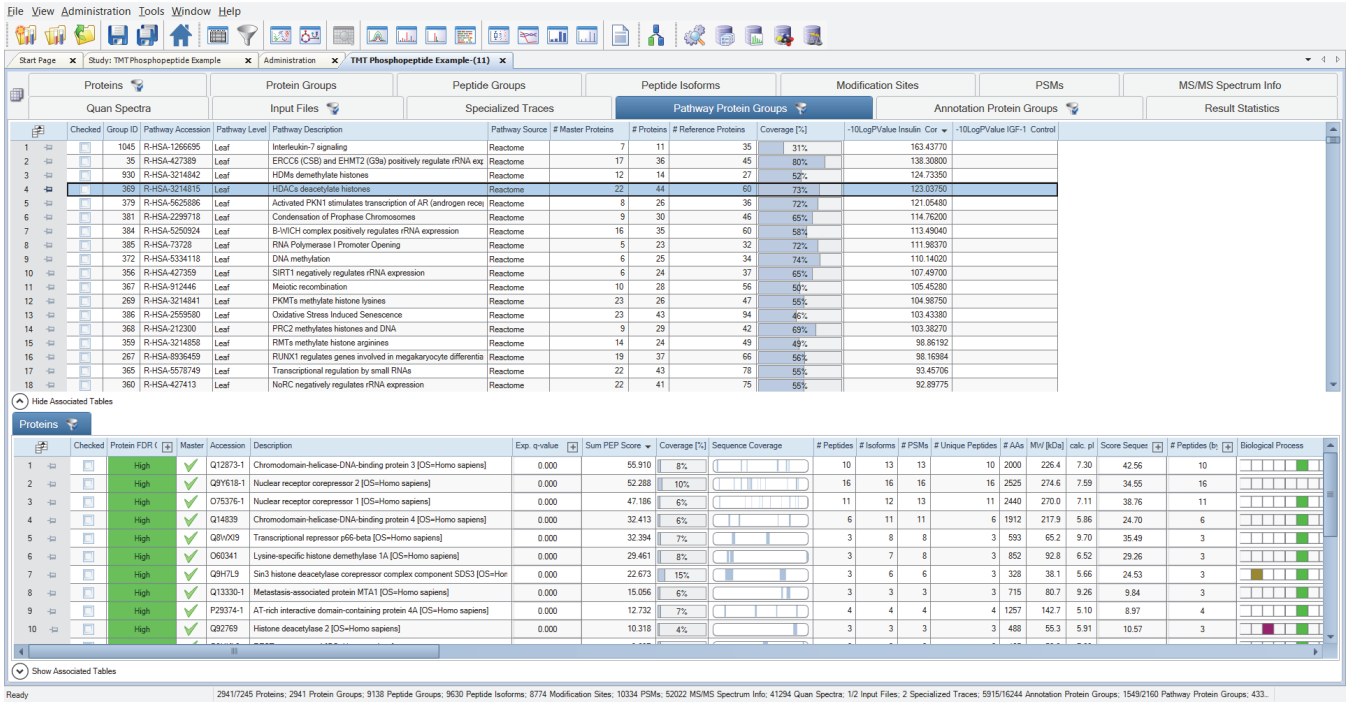
PathwayProteinGroup.out.txt - Notepad

File Edit Format View Help

```
"Pathway Protein Groups Group ID"       "Pathway Accession"     "-10LogPValue Insulin  Control" "-10LogPValue IGF-1
Control"
1       "hsa01100"      NA      NA
2       "hsa00270"      NA      NA
3       "R-HSA-1430728" NA      NA
4       "R-HSA-71291"   NA      NA
5       "R-HSA-1614635" NA      NA
6       "R-HSA-1237112" NA      NA
7       "R-HSA-351202"  NA      NA
8       "WP268" NA      NA
9       "WP61"  NA      NA
10      "hsa04152"      NA      NA
11      "hsa04150"      NA      NA
12      "hsa04140"      NA      NA
13      "hsa04211"      NA      NA
14      "WP1471"        NA      NA
15      "WP2509"        NA      NA
16      "WP3844"        NA      NA
17      "WP3932"        NA      NA
18      "WP615" NA      NA
19      "R-HSA-199991"  NA      NA
20      "R-HSA-9007101" NA      NA
21      "R-HSA-8854214" NA      NA
22      "R-HSA-8953897" NA      NA
23      "R-HSA-5653656" NA      NA
24      "R-HSA-1632852" NA      NA
25      "R-HSA-8934903" NA      NA
26      "R-HSA-8876198" NA      NA
```

The scripting node reads the output text files specified by the node_response.json file and displays the new columns in the tables in the application. Figure 421 shows the two new columns added to the Pathway Protein Groups table as specified in the node_response.json example file shown in Figure 419.

Figure 421. Pathway Protein Groups including additional columns



The node_response.json File Parameter Descriptions

In the node_response.json file, there are a number of options that can be selected for the new columns for specialized display in the result. The array `Options` holds these optional parameters as key-value pairs that are recognized when creating the new columns. There are different key values available:

- `PositionAfter`—The new column will be placed after the specified column.
- `PositionBefore`—The new column will be placed before the specified column.
- `RelativePosition`—An integer value specifying the relative position of the column.
- `PlotType` - specifies the plot type where the new column can be plotted in a chart
  - Valid values are `Numeric`, `Categorical`, and `Ordinal`. If there are multiple plot types, separate by comma (for example, `Numeric, Categorical`).
  - If no plot type is defined:
    - Boolean columns are automatically classified as `Categorical`
    - Numeric columns are classified as `Numeric, Ordinal`.
    - All others are specified as `None`.
- `FormatString` - specified how numerical values are formatted and displayed. By default, float values have a format string of F5, which means they are displayed to 5 significant figures.
- `SpecialCellRenderer`—The data within a column can be shown with an alternative presentation rather than as a raw data value. Here are the possible options:
  - `exclamationMark`—90550C2D-EB8D-443E-9F23-9E735FB23F9A - displays a string column. It shows an error icon if the value is not empty. The tool tip for the image is the full text.
  - `boolCheckMark`—330D9522-50CC-41B7-9D45-5E5D8F708103 - displays a Boolean value. It shows a green check mark if the value is true.
  - `boolCrossMark`—4B8956F1-3A54-423D-8654-60676825230F - displays a Boolean value. It shows a check mark (X) if the value is true.
  - `boolYesNo`—9085D14E-388D-4EF0-91C1-B4B2BDF33AD3 - displays a Boolean value with the words "Yes" and "No".
  - `fileName`—EB29D794-4F2E-4785-8B80-A24D8C0FB3E4 - for string column. Displays a filename (a string) by only showing the filename in the table field and showing the full path in the tooltip.
  - `fileSize`—FBBFBB0B-A2D0-468F-95BF-25BCD0A42FE9 - displays an integer number as a filesize.
  - `percentBar`—64516B23-565C-4036-87DA-29189E9E62A3 - displays a number between 0 and 100 as a column between 0% and 100%.

– `monospaced`—59E4C754-AD43-42FD-B4D9-C8CF0F622BD3 - displays any number or text using a monospaced font rather than a proportional font.

– `concatenatedMultiline`—410B767F-0C21-4E97-BB3D-041186955807 - displays a text that consists of multiple lines as one line that features the multiple lines concatenated using a ; (semicolon).

Guidelines for R Developers

Make sure the ID column for the target table is written to the output text file as a string for the Proteins table.

The example in Figure 418 works for the Pathway Protein Groups because the IDs are small integers. For tables with IDs of `DataType: Long`, this only works using a `String`, as shown by the quotes around the ID value for each row in the output file.

Creating New Tables and Connecting Them to Existing Tables

For more complex scripts, it may be necessary to create new tables in the Proteome Discoverer results.

In addition to the steps to start the script described in Writing Results as New Columns to an Existing Table, do the following:

1. Create a new table and add it to the node_response.json for display in the results and a new tab-separated text file must be created with contents of the new table.
2. If there is a relationship between this new table and one or more existing tables, then a connection table is required to connect the IDs from each table to be usable by the "Show Associated Tables" option in the application.

❖ **To create a new table and connect it to an existing result table within the executable + script**

1. Create the new table to be imported into Proteome Discoverer
   a. Designate a unique ID number for each row in the new table.

      It is easiest to designate each row as an integer starting with an ID of 1.

      The ID for the new table must have the name of the table plus an additional ID term. In the example in Figure 422, the ID column is "ProteinStartsWith StartsWithID" for the table called ProteinStartsWidth. When the table name is not in the ID, the import will fail.

   b. Write the column headers as the first row in the output .txt file, separated by tabs.

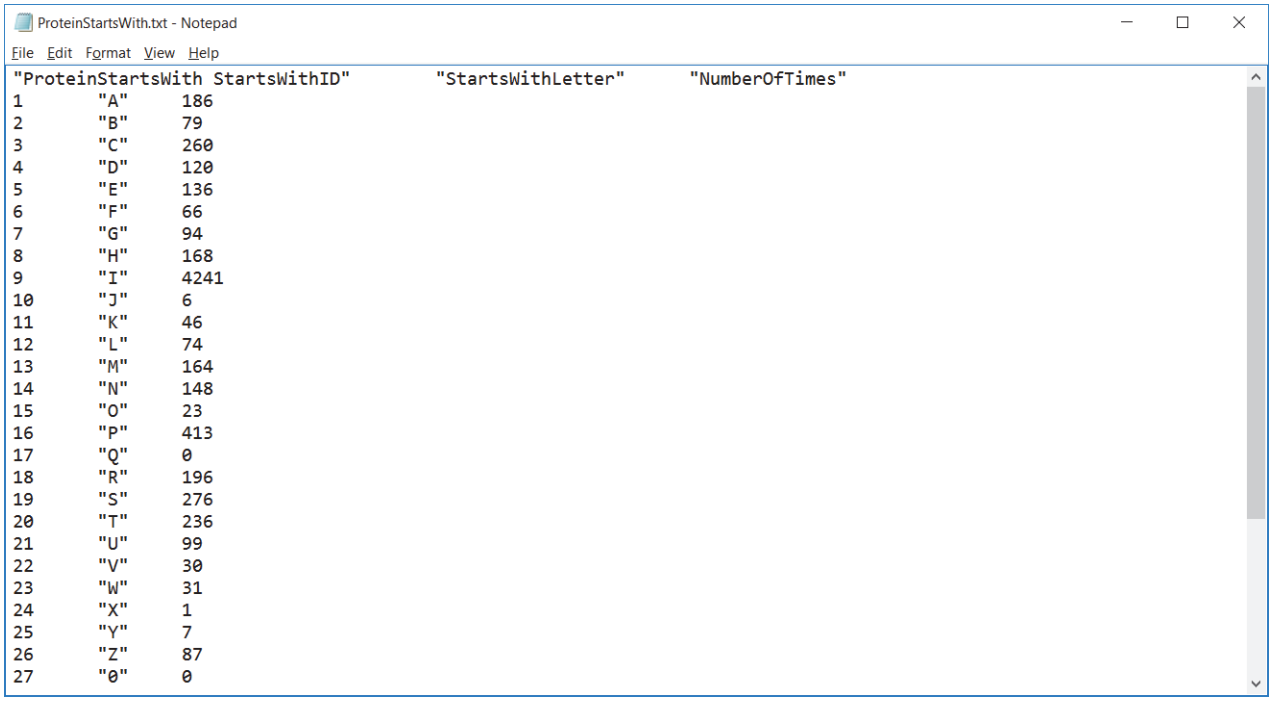
   c. Fill in the rest of the table columns with the results of interest.

      Figure 422 shows the results. The second column, ProteinStartsWith, shows the letters A-Z and numbers 0-9. The third column, StartsWithID, shows the number of times a protein was found with a description that starts with that character.

d. Write this file out as a .txt file to the same temporary folder where node_args.json and node_response.json are located.

These data were generated using the second familiarization exercise in the *Proteome Discoverer Familiarization Guide*. The protein counts depend on the input dataset.

Figure 422. The example table produced by the "ProteinStartsWith.R" script



2. Create the connection tables between the newly generated table and one or more existing tables.

   The Scripting Node uses the connection table when importing the data to allow use of the "Show Associated Tables" function in the application. The file must comply with the following:

   - A new .txt file needs contain the ID numbers that are related between the two tables to be connected. For this example script, the "ProteinStartsWith StartsWithID" must be associated with all of the proteins whose descriptions that start with that character. The example script performs this row-by-row through the "TargetProtein.txt" file exported by the Scripting Node when the Proteins table is exported.
   - The ID column headers need to have the same name as the ID columns from other text files for the two tables to be connected. In this example, the columns are "ProteinStartsWith StartsWith ID" and "Proteins Unique Sequence ID".

- For the .txt file name, use the names of the two tables to be connected, separated with a hyphen ("-"). For example, in Figure 423, ProteinStartsWith-TargetProtein.txt.

Figure 423. Example connection table between the newly created ProteinStartsWith table and the Proteins table

ProteinStartsWith-TargetProtein.txt - Notepad

File Edit Format View Help

"ProteinStartsWith StartsWithID"	"Proteins Unique Sequence ID"
"9"	"-9223112801231503285"
"9"	"-9220545879072249736"
"9"	"-9220381828272499686"
"13"	"-9216601272429108085"
"9"	"-9213770331498487616"
"9"	"-9209564059501719388"
"9"	"-9209357573003678006"
"9"	"-9208617211276488835"
"19"	"-9201775609484186021"
"9"	"-9200601933759959983"
"9"	"-9195264797577560147"
"9"	"-9191541618676727342"
"3"	"-9189590484777075943"
"9"	"-9187629707458947599"
"9"	"-9186700443057394725"
"21"	"-9180355309392093096"
"21"	"-9171365705039643338"
"6"	"-9169618704545618170"
"9"	"-9165754455413116234"
"12"	"-9158242191706838173"
"16"	"-9157567409128222481"
"18"	"-9156757260988577689"
"18"	"-9155703229501675198"
"9"	"-9153838209166502140"
"7"	"-9151222149414700738"
"9"	"-9148036673416193879"
"8"	"-9144980634344963458"

3. Create a node_response.json file that includes the new table and the connection table. The file must do the following:

   - The node_response.json file (Figure 424) informs the Scripting Node which new data to present in the results. When a new table is to be created, the name of the table, the link to the .txt file that was created by the script that contains the data for the new table, and the description of each column name should be included.

     The names in the node_response.json must be exactly the same as those from the column headers in the text file and in quotes. The ID column should be labeled as "ID":"ID".

   - The information for any connection table also needs to be added as an additional table in the node_response.json file. The data file link should be where the connection table (Figure 423) was saved. The data format in this case is a "CSVConnectionTable" and the Options should specify the first table (for example, "ProteinStartsWith") and the second table (for example, "Proteins"). For the specific example, the table is structured as Figure 424.

Figure 424. The node_response.json file

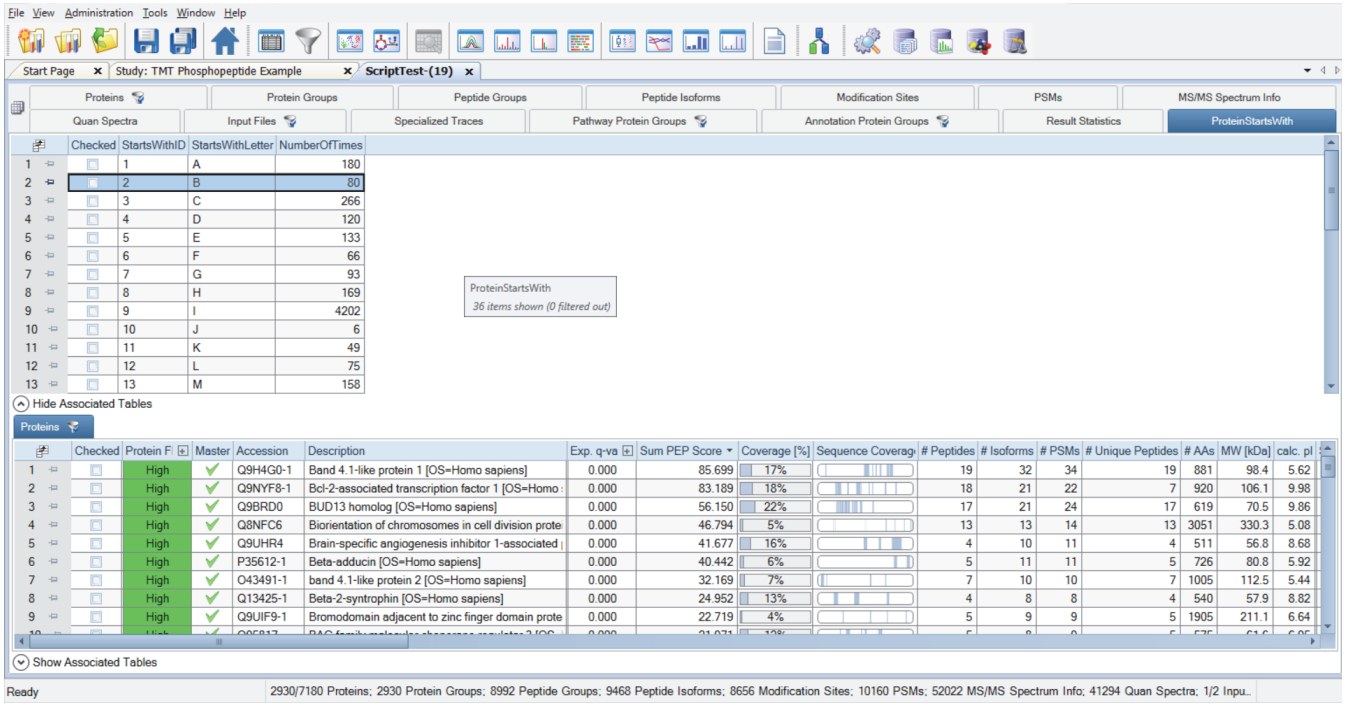

```
{
 "CurrentWorkflowID": 113,
 "ExpectedResponsePath":"",
 "Tables":[
  {
   "TableName":"ProteinStartsWith",
   "DataFile":"C:/ProgramData/Thermo/Proteome Discoverer
2.4/Scratch/Job113/Script(16)/ProteinStartsWith.txt",
   "DataFormat":"CSV",
   "Options":{},
   "ColumnDescriptions":[
    {
     "ColumnName":"ProteinStartsWith StartsWithID",
     "ID":"ID",
     "DataType":"Int",
     "Options":{}
    },
    {
     "ColumnName":"StartsWithLetter",
     "ID":"",
     "DataType":"String",
     "Options":{}
    },
    {
     "ColumnName":"NumberOfTimes",
     "ID":"",
     "DataType":"Int",
     "Options":{}
    }
   ]
  },
  {
   "TableName":"ProteinStartsWith-TargetProteins",
   "DataFile":"C:/ProgramData/Thermo/Proteome Discoverer
2.4/Scratch/Job113/Script(16)/ProteinStartsWith-TargetProtein.txt",
   "DataFormat":"CSVConnectionTable",
   "Options":{
    "FirstTable":"ProteinStartsWith",
    "SecondTable":"Proteins"
   },
   "ColumnDescriptions":[
    {
     "ColumnName":"ProteinStartsWith StartsWithID",
     "ID":"ID",
     "DataType":"Int",
     "Options":{}
    },
    {
     "ColumnName":"Proteins Unique Sequence ID",
     "ID":"ID",
     "DataType":"Long",
     "Options":{}
    }
   ]
  }
 ]
}
```


4. Save the node_response.json file to the same temporary folder as the node_args.json file originally written by the Scripting Node.

   The scripting node will automatically load the node_response.json file and import the specified results.

   For the ProteinStartsWith example, the result in the application looks like Figure 425. The connection table allows the application to display only those proteins that start with the letter selected in the ProteinStartsWith table.

Figure 425. Output of ProteinStartsWith script for the data from Familiarization Exercise 2



	Checked	StartsWithID	StartsWithLetter	NumberOfTimes
1		1	A	180
2		2	B	80
3		3	C	266
4		4	D	120
5		5	E	133
6		6	F	66
7		7	G	93
8		8	H	169
9		9	I	4202
10		10	J	6
11		11	K	49
12		12	L	75
13		13	M	158

	Checked	Protein F	Master	Accession	Description	Exp. q-va	Sum PEP Score	Coverage [%]	Sequence Coverag	# Peptides	# Isoforms	# PSMs	# Unique Peptides	# AAs	MW [kDa]	calc. pI
1		High	✓	Q9H4G0-1	Band 4.1-like protein 1 [OS=Homo sapiens]	0.000	85.699	17%		19	32	34	19	881	98.4	5.62
2		High	✓	Q9NYF8-1	Bcl-2-associated transcription factor 1 [OS=Homo	0.000	83.189	18%		18	21	22	7	920	106.1	9.98
3		High	✓	Q9BRD0	BUD13 homolog [OS=Homo sapiens]	0.000	56.150	22%		17	21	24	17	619	70.5	9.86
4		High	✓	Q8NFC6	Biorientation of chromosomes in cell division prote	0.000	46.794	5%		13	13	14	13	3051	330.3	5.08
5		High	✓	Q9UHR4	Brain-specific angiogenesis inhibitor 1-associated	0.000	41.677	16%		4	10	11	4	511	56.8	8.68
6		High	✓	P35612-1	Beta-adducin [OS=Homo sapiens]	0.000	40.442	6%		5	11	11	5	726	80.8	5.92
7		High	✓	O43491-1	band 4.1-like protein 2 [OS=Homo sapiens]	0.000	32.169	7%		7	10	10	7	1005	112.5	5.44
8		High	✓	Q13425-1	Beta-2-syntrophin [OS=Homo sapiens]	0.000	24.952	13%		4	8	8	4	540	57.9	8.82
9		High	✓	Q9UIF9-1	Bromodomain adjacent to zinc finger domain prote	0.000	22.719	4%		5	9	9	5	1905	211.1	6.64

Guidelines for Developers

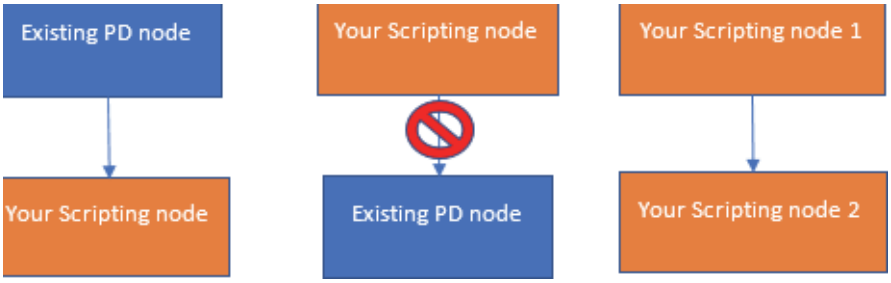
Consider the following:

- It may be more convenient to create a node_response.json file in a text editor, leaving the DataFile parameters blank, and save the file in the same folder as the script. After starting the scripting node, the script can read in the generic node_response.json, add the path of the text files in the temporary folder, and write the updated node_response.json to the temporary folder.
- Use the rjsonio library to load the pre-made node_response.json file, and fill in the details as mentioned previously.

Registering Standalone Scripting Nodes

In addition to the Scripting Node provided, you can register your own node that runs a script. Your node can run in a consensus or processing workflow or as post-processing node. The rules for integrating your own scripting node are different than those for the provided Scripting Node.

Figure 426. Rules for adding nodes to workflows



❖ To register a standalone node

1. To create a registered node, create a new subfolder under the \Tools\Scripts folder in the Proteome Discoverer installation folder.

   The default location is:

   C:\Program Files\Thermo\Proteome Discoverer 2.4\Tools\Scripts*MyScript*

2. Copy the script to this folder.

3. Create a node.json file, and copy it to this folder.

   The node.json file is used to configure the input values for the script and the other nodes this standalone Scripting Node connects with (Figure 427).

Figure 427. Example node.json file


```
{
"Guid": "41e5696e-51de-40bc-9850-d6266665d2c7",
"Category": "Feature Detection & Quantification",
"Name": "Test Node",
"Description": "This is My Test Node",
"Version": 1,
"Publisher": null,
"DevelopedBy": "USer XYZ",
"Homepage": null,
"LegalInformation": null,
"WorkflowType": "Consensus",
"ImageLarge": "IMG_32x32.png",
"ImageSmall": "IMG_16x16.png",
"License": null,

"ScriptProcessorArguments": {

"ExecutableCommandLineArguments": "%NODEARGS%",
"ExecutablePath": "D:/SampleScript/SampleScript.exe",

"RequestedTablesAndColumns": "PSMs: Delta Score, Modifications",

"UseRFriendlyNames": "true"
},
"Parameters": [
{
"DataType": "Float",
"Name": "Threshold [%]",
"Default": "1",
"Minimum": "0",
"Maximum": "100"
},
{
"DataType": "String",
"Name": "Display Name",
}
],
"Connections": [
{
"Incoming": true,
"DataType": "http://thermo.magellan.com/owl/DataTypes/Psms",
"Attributes": ["http://thermo.magellan.com/owl/DataTypes/WithDecoys"]
}
]
}
```


The node.json File Contents

The node.json file contains:

- `Guid`: a unique GUID (32 hexadecimal characters from 0-E), which identifies the node. Each node must have its own GUID. There are several online GUID generators you can use to generate a random GUID. The GUID must have the structure shown in Figure 427.
- `Category`: optional parameter that controls where the node is displayed in the workflow editor in the list of available workflow nodes. If the category is empty, "Post-Processing" is used for post-processing nodes, or "Misc" for others.
- `Name`: the node name displayed in the Workflow Editor.
- `Description`: optional, a description for the node displayed in the Workflow Editor.

- `WorkflowType`: "Consensus" when the node is available in consensus workflows. Otherwise the node is usable in processing workflows.
- `Version`: optional node version number.
- `Publisher`: optional publisher information.
- `DevelopedBy`: optional developers of the node.
- `Homepage`: optional homepage URL.
- `LegalInformation`: optional legal information relevant for this node.
- `ImageLarge`: optional relative path to the large node icon. Image should have size of 32x32 pixels
- `ImageSmall`: optional relative path to the small node icon. Image should have size of 16x16 pixels.
- `ScriptProcessorArguments`:
  - `ExecutableCommandLineArguments`: `%NODEARGS%`
  - `ExecutablePath`: `D:/SampleScript/SampleScript.exe`
  - `UseRFriendlyNames`: `false`
  - `RequestedTablesAndColumns`: `PSMs: Delta Score`, `Modifications`
- Parameters is optional section that defines parameters for the user shown in the workflow tree. Parameters is an array of parameter definitions. Each has following settings:
  - `DataType`: can be `FloatingPointType`, `StringType`, `IntegerType`, `BooleanType`. Defines the value type of the parameter.
  - `Name`: the parameter display name.
  - `Description`: optional parameter description shown in the Workflow Editor.
  - `Category`: optional parameter category shown in the Workflow Editor.
  - `Default`: optional default value for the parameter.
  - `Minimum`: optional for integer and double parameters. This defines a minimum allowed value to be entered.
  - `Maximum`: optional for integer and double parameters. This defines a minimum allowed value to be entered.
- `SelectionValues`: optional list of predefined selection values. This only affects floating-point, integer, and string parameters. Add an empty selection value, for example, if the value is not required to select a value.

- `Connection:` optional section that defines the connection points for the node in the Workflow Editor. If no connection point is defined, the node is a post-processing node. To connect the node with other nodes, you must define one incoming connection point. Once a connection point is defined, the node can be used as a consensus node or a processing node. To allow other nodes to connect to the node, you must define outgoing connections.
- Each connection has the following settings:
  - `Incoming`: true or false. Set this to true if the connection is incoming. Set it to false, if it is outgoing.
  - `DataType`: defines which data should be received, see the description below.
  - `Attributes`: optional attributes for the data type, see the description below
  - `Group`: optional group name, this is used as identifier to group different connections defined for this node.

  The `DataType` and `Attributes` can be extracted from a workflow template. To connect your Scripting Node to an existing application node, create a workflow with this application node, save it, and open the workflow with a text editor. In the workflow xml, search for the Proteome Discoverer node definition.

  For example, to connect your node to the PSM Grouper node. The xml of the PSM Grouper node looks like following:

  The PSM Grouper node has one outgoing connection providing peptide groups. To connect your node to this node, you need to define following incoming connection point:

```
{
"Incoming": true,
"DataType": "http://thermo.magellan.com/owl/DataTypes/PeptideGroups"
}
```

  For example, to connect your node with Peptide Validator node, which defines following connections:

  To connect to Peptide Validator node, and not to the PSM Grouper node, you must add attributes to your connection point:

```
{
"Incoming": true,
"DataType": "http://thermo.magellan.com/owl/DataTypes/PeptideGroups",
"Attributes":
["http://thermo.magellan.com/owl/DataTypes/Validated","http://thermo.
magellan.com/owl/DataTypes/WithDecoys"]
}
```

Note After adding a new subfolder for a new script node or after changing anything in the node configuration, you need to scan the node for missing features. For instructions, see "Installing or Updating a Scripting Workflow Node." You need to close all open studies or standalone workflow editors afterwards or restart the application.

Node Parameters

Table 191 describes the parameters of the Scripting Node.

Table 191. Scripting Node parameters (Sheet 1 of 2)

Parameter	Definition
Path to Executable	Specifies the path to an arbitrary command line executable that is started as the workflow execution reaches the Scripting Node. Users of integrated development environments (IDEs) for R or python should run their scripts using the corresponding command line tool, i.e., RScript.exe or python.exe, respectively. If a compiled executable program is to be used, the full path of that executable should be selected.
Command Line Arguments	The parameter should include all arguments that need to be passed directly to the executable. If using a scripting language, in general the name of the script to be executed will be the first command line argument. The path to the node_args.json file written by the Scripting Node can be inserted into the command line using the placeholder %NODEARGS%. This file will be written into the scratch directory. More information about the structure of the node_args.json file is discussed below. If more parameters past %NODEARGS% are specified, see Reading the node_args.json File into the Script.

Table 191. Scripting Node parameters (Sheet 2 of 2)

Parameter	Definition
Use R-Friendly Names	If set to True, column names are exported in a format that is more readily used by scripting languages such as R. That is, characters such as ",",#,%, [], or / are removed from the column header string altogether or replaced by alphanumeric text.
Archive Datafiles	If set to True, a zip file is created in the study folder with all files used by the Scripting Node. The zip file includes the node_args.json file and the various text files with the exported columns. This option is useful when debugging a script, and the exported results from the application can be used for other purposes. If results are being read back into the application for display, all of the text files containing information are archived as well as the node_response.json file.

Table Name and Column Name Listing

These are the exact names of columns to be exported for use with an external script. These are only correct when the "Use R-friendly headers" option is set to True in the Scripting Node.

Table 192. Paragraphs example

Table	Column Names
Proteins	Abundance Ratio Adj P-Value, Abundance Ratio P-Value, Abundance Ratios, Abundance Ratios log2, Abundances, Abundances Counts, Abundances Grouped, Abundances Grouped Count, Abundances Grouped CV, Abundances Normalized, Abundances Scaled, Accession, Biological Process, calc pI, Cellular Component, Checked, Chromosome, Coverage in Percent, Coverage in Percent by Search Engine, Description, Ensembl Gene ID, Entrez Gene ID, Exp q-value, FASTA Title Lines, Found in Sample Groups, Found in Samples, Gene Symbol, GO Accessions, KEGG Pathway Accessions, KEGG Pathways, Master, Modifications, Molecular Function, MW in kDa, Number of AAs, Number of Crosslinks, Number of Decoy Proteins, Number of Isoforms, Number of Peptides, Number of Peptides by Search Engine, Number of Protein Annotation Groups, Number of Protein Groups, Number of Protein Pathway Groups, Number of Protein Unique Peptides, Number of PSMs, Number of PSMs by Search Engine, Number of Razor Peptides, Number of Unique Peptides, Pfam IDs, Protein FDR Confidence, Protein Group IDs, Reactome Pathway Accessions, Reactome Pathway Accessions All, Reactome Pathways, Score Sequest HT, Sequence, Sum PEP Score, WikiPathway Accessions, WikiPathways
Protein Groups	Checked, Found in Sample Groups, Found in Samples, Group Description, Number of Isoforms, Number of Proteins, Number of PSMs, Number of Unique Peptides, Protein Group ID
Peptide Groups	Abundance Ratio, Abundance Ratio Adj P-Value, Abundance Ratio log2, Abundance Ratio P-Value, Abundances, Abundances Counts, Abundances Grouped, Abundances Grouped Count, Abundances Grouped CV, Abundances Normalized, Abundances Scaled, Annotated Sequence, Charge by Search Engine, Checked, Concatenated Rank by Search Engine, Confidence, Confidence by Search Engine, Delta Cn by Search Engine, Delta M in ppm by Search Engine, Delta mz in Da by Search Engine, Delta Score by Search Engine, Found in Sample Groups, Found in Samples, Master Protein Accessions, Master Protein Descriptions, Modifications, Modifications all possible sites, Modifications in Master Proteins, Modifications in Master Proteins all Sites, mz in Da by Search Engine, Number of Isoforms, Number of Missed Cleavages, Number of Protein Groups, Number of Proteins, Number of PSMs, Percolator PEP by Search Engine, Percolator q-Value by Search Engine, Percolator SVMScore by Search Engine, Positions in Master Proteins, Protein Accessions, PSM Ambiguity, Quan Info, Qvality PEP, Qvality q-value, Rank by Search Engine, RT in min by Search Engine, Search Engine Rank by Search Engine, Sequence, Sequence Length, SVM_Score, Theo MHplus in Da, Top Apex RT in min, XCorr by Search Engine,

Table 192. Paragraphs example

Table	Column Names
PSMs	Activation Type, Annotated Sequence, Charge, Checked, Concatenated Rank, Confidence, Delta Cn, Delta M in ppm, Delta mz in Da, Delta Score, File ID, First Scan, Identifying Node, Identifying Node No, Identifying Node Type, Intensity, Ion Inject Time in ms, Ions Matched, Isolation Interference in Percent, Last Scan, Master Protein Accessions, Master Protein Descriptions, Master Scans, Matched Ions, MHplus in Da, Modifications, MS Order, mz in Da, Number of Missed Cleavages, Number of Protein Groups, Number of Proteins, Original Precursor Charge, Peptides Matched, Protein Accessions, Protein Descriptions, PSM Ambiguity, PSMs Peptide ID, PSMs Workflow ID, Rank, RT in min, Search Engine Rank, Search ID, Sequence, Spectrum File, Theo MHplus in Da, Total Ions, XCorr
MS/MS Spectrum Info	Activation Type, Best PSM Ambiguity, Checked, Creator Node No, File ID, First Scan, Ion Inject Time in ms, Isolation Interference in Percent, Last Scan, Mass Analyzer, MS Order, MSMS Spectrum Info Spectrum ID, Number of Master Scans, Number of PSMs, Number of Scans, Original Mass, Original Precursor Charge, Precursor Charge, Precursor Intensity, Precursor MHplus in Da, Precursor mz in Da, RT in min, Scans, Search ID, Spectrum File
Input Files	Creation Date, File ID, File Name, Input Files, Input Files Workflow Level, Instrument Hardware, Instrument Name, Ref File ID, RT Range in min, Software Revision
CSMs	All Scans, Charge, Charges A, Charges B, Checked, Confidence, Crosslink localisation probability, Crosslink Strategy, Crosslink Type, Crosslinker, Crosslinker Position A, Crosslinker Position B, CSMs, Delta XlinkX Score, Delta M in ppm, File ID, First Scan, Gene Name A, Gene Name B, Gene Names, Identified By, Identifying Node No, Is Decoy, Leading Protein Position A, Leading Protein Position B, MHplus in Da, Modifications A, Modifications B, mz in Da, Mzs A in Da, Mzs B in Da, Number of Identified MS2 Scans, Number of Identified MS3 Scans, Protein Descriptions A, Protein Descriptions B, Reporter Ion Score, RT in min, Sequence, Sequence A, Sequence B, Spectrum File, Spectrum file path, XlinkX Score
Crosslink Summary	Crosslink Summary, Number CSM's, Number interlinks, Number intralinks, Number of isotopes re-sequenced MS2, Number of isotopes re-sequenced MS3, Number of isotopes sequenced MS2, Number of isotopes sequenced MS3, Number Reporters, Raw file
Crosslinks	Accession A, Accession B, Ambiguity, Checked, Confidence, Crosslink localization probability, Crosslink Sequence, Crosslink Type, Crosslinker, Crosslinks, Crosslinks Peptide Group ID, Max XlinkX Score, Modifications A, Modifications B, Number of CSMs, Number of Proteins, Position A, Position B, Protein Descriptions A, Protein Descriptions B, PSM Ambiguity, Sequence A, Sequence B

Table 192. Paragraphs example

Table	Column Names
Crosslink Reporter Peaks	Charge, Charge xA, Charge xB, Crosslink event, Crosslink Reporter Peaks, Crosslink Strategy, Crosslinker, Diagnostic Peaks Isotope Error, Diagnostic Peaks p-Value, Fragmentation efficiency, Fragmentation Scan Numbers, Fragmentation Spectrum IDs, Intensity, Mass in Da, Modification definitions A, Modification definitions B, Modifications A, Modifications B, mz in Th, mz xA in Th, mz xB in Th, Pre Scan Number, Pre Spectrum ID, Relative presence, Retention Time in min, Scan Number, Spectrum File Path, Spectrum ID, Unmodified Mass xA in Da, Unmodified Mass xB in Da,
Crosslink MS3 Scans	Crosslink Event, Crosslink localisation probability, Crosslink MS3 Scans, Crosslink Reagent, Crosslink Strategy, Definition of Peptide, Delta M in ppm, First Scan, Fraction, FractionIonsMatchedAseries, FractionIonsMatchedBseries, FractionIonsMatchedCseries, FractionIonsMatchedXseries, FractionIonsMatchedYseries, FractionIonsMatchedZseries, Fragment mz error, Fragment mz error IQR, Fragment mz error IQR ppm, Fragment mz error ppm, Fragmentation efficiency, Gene Names, Identification Delta Score, Identification Score, Identification Usage, IntensityIonsMatchedAseries, IntensityIonsMatchedBseries, IntensityIonsMatchedCseries, IntensityIonsMatchedXseries, IntensityIonsMatchedYseries, IntensityIonsMatchedZseries, Is Decoy, Is Significant, Isolation Interference, Leading Protein, Leading Protein Crosslink Position, Master Scan, Matched Ion Count, Missed Cleavages, Modifications, Number of Matches, Number of Modifiable Residues, Number of Modified Residues, Number of Unique Variable Modifications, Peptide ID, Peptide Position, Precursor Charge, Precursor Isotope Offset, Precursor MHplus in Da, Precursor mz in Th, Protein Crosslink Positions, Protein Descriptions, Reporter ID, Reporter Score, RT in min, Sequence, Sequence Coverage, Spectrum File Path, Theoretical Mass in Da, Total Ion Count, Validator Q-value

Table 192. Paragraphs example

Table	Column Names
Crosslink MS2 Scans	Annotated Sequences, Charge State Peptide A, Charge State Peptide B, Crosslink Event, Crosslink localisation probability A, Crosslink localisation probability B, Crosslink MS2 Scans, Crosslink Positions A, Crosslink Positions B, Crosslink Reagent, Crosslink Strategy, Crosslink Type, Definition of Peptide A, Definition of Peptide B, Delta M A in ppm, Delta M B in ppm, Delta M in ppm, First Scan, Fraction, FractionIonsMatchedAseries, FractionIonsMatchedBseries, FractionIonsMatchedCseries, FractionIonsMatchedXseries, FractionIonsMatchedYseries, FractionIonsMatchedZseries, Fragment mz error, Fragment mz error IQR, Fragment mz error IQR ppm, Fragment mz error ppm, Fragmentation efficiency, Gene Names A, Gene Names B, Identification Delta Score, Identification Score, Identification Usage, IntensityIonsMatchedAseries, IntensityIonsMatchedBseries, IntensityIonsMatchedCseries, IntensityIonsMatchedXseries, IntensityIonsMatchedYseries, IntensityIonsMatchedZseries, Is Decoy, Is Significant, Isolation Interference, Leading Crosslink Position A, Leading Crosslink Position B, Leading Protein A, Leading Protein B, Master Scan, Matched Ion Count, Missed Cleavages A, Missed Cleavages B, Modifications A, Modifications B, mz's Peptide A in Da, mz's Peptide B in Da, Number of Matches A, Number of Matches B, Number of Modifiable Residues A, Number of Modifiable Residues B, Number of Modified Residues A, Number of Modified Residues B, Number of Unique Variable Modifications A, Number of Unique Variable Modifications B, Peptide ID A, Peptide ID B, Peptide Position A, Peptide Position B, Precursor Charge, Precursor Isotope Offset, Precursor MHplus in Da, Precursor mz in Th, Protein Descriptions A, Protein Descriptions B, Reporter ID, Reporter Score, RT in min, Sequence A, Sequence B, Sequence Coverage A, Sequence Coverage B, Spectrum File Path, Theoretical Mass in Da, Total Ion Count, Unmodified mass Peptide A MHplus in Da, Unmodified Mass Peptide B MHplus in Da, Validator Q-value
Crosslink Full Scan Statistics	Crosslink Full Scan Statistics, Cycle time, Number of MS2 scans, Number of MS3 scans, Raw file, Retention time, Scan number
Specialized Traces	Checked, Description, File ID, Scan Filter, Specialized Traces ID, Spectrum File, TraceDetector Type
Result Statistics	Arithmetic Mean, Count, CV, IQR, Maximum, Median, Minimum, Name, Result Statistics ID, SD, Sum
Pathway Protein Groups	Checked, Number of Master Proteins, Number of Proteins, Pathway Accession, Pathway Description, Pathway Level, Pathway Protein Groups Group ID, Pathway Source
Consensus Features	Abundances Count (Distribution), Abundances Normalized (Distribution), Abundances Origin (Distribution), Abundances per File (Distribution), Avg Apex RT in min, Avg mz in Da, Charge, Checked, Consensus Features, Delta ApexRT in min, Delta M in ppm, Left RT in min, Max Apex SN, Number of Isotopes, Number of Peptides, Number of PSMs, Quan Info, Quan Info Details, Right RT in min
Annotation Protein Groups	Annotation Accession, Annotation Description, Annotation Level, Annotation Protein Groups Group ID, Annotation Source, Checked, Number of Master Proteins, Number of Proteins

Table 192. Paragraphs example

Table	Column Names
PrSMs	Activation Type, Annotated Sequence, C Score, Charge, Checked, Confidence, Corrected Delta Mass Da, Corrected Delta Mass ppm, Delta Mass in Da, Delta Mass in ppm, External Top Down Displays, File ID, Fragment Count, Fragmentation Scans, Identifying Node, Identifying Node No, Identifying Node Type, Intensity, Ion Inject Time in ms, Ions Matched in Percent, Ions Matched in Percent, -Log E-Value, -Log P-Score, Mass in Da, Matched Ions, Modifications, MS Order, mz in Da, Number of Fragmentation Scans, Number of Precursor Scans, Number of Proteins, Original Precursor Charge, Percent Residue Cleavages, Precursor Scans, Protein Accessions, Protein Descriptions, PrSMs PrSM ID, PrSMs Workflow ID, Q-Value, Rank, RT in min, Search Engine Rank, Search ID, Sequence, Spectrum File, Theo Mass in Da, Total Ions, XCorr
Modification Sites	Checked, Confidence, Master, Modification Name, Modification Sites Modification Site ID, Modification Sites Workflow ID, Motif, Number of Modified PSMs, Number of Unmodified PSMs, Peptide Sequence, Position, Position in Peptide, Protein Accession, Protein Description, Site Probability, Target Amino Acid
Quan Spectra	Abundance, Abundances Normalized, Activation Type, Average Reporter SN, Best PSM Ambiguity, Checked, Creator Node No, File ID, First Scan, Ion Inject Time in ms, Isolation Interference in Percent, Last Scan, Mass Analyzer, Master Scans, MS Order, Number of Master Scans, Number of PSMs, Number of Scans, Original Mass, Original Precursor Charge, Precursor Charge, Precursor Intensity, Precursor MHplus in Da, Precursor mz in Da, Quan Info, Quan Info Details, Quan Spectra Spectrum ID, Quan Spectra Workflow ID, RT in min, Scans, Search ID, Spectrum File
Peptide Isoforms	Abundance, Abundance Ratio, Abundance Ratio Adj P-Value, Abundance Ratio log2, Abundance Ratio P-Value, Abundances Count, Abundances Grouped, Abundances Grouped Count, Abundances Grouped CV in Percent, Abundances Normalized, Abundances Scaled, Charge by Search Engine, Checked, Concatenated Rank by Search Engine, Confidence by Search Engine, Delta Cn by Search Engine, Delta M in ppm by Search Engine, Delta mz in Da by Search Engine, Delta Score by Search Engine, Master Protein Accessions, Master Protein Descriptions, Modification Pattern, Modifications, mz in Da by Search Engine, Number of Isoforms, Number of Missed Cleavages, Number of Protein Groups, Number of Proteins, Number of PSMs, Peptide Isoforms Peptide Group ID, Percolator PEP by Search Engine, Percolator q-Value by Search Engine, Percolator SVMScore by Search Engine, Protein Accessions, PSM Ambiguity, Quan Info, Rank by Search Engine, RT in min by Search Engine, Search Engine Rank by Search Engine, Sequence, Sequence Length, Theo MHplus in Da, XCorr by Search Engine

Table 192. Paragraphs example

Table	Column Names
Peptide Isoforms	Abundance, Abundance Ratio, Abundance Ratio Adj P-Value, Abundance Ratio log2, Abundance Ratio P-Value, Abundances Count, Abundances Grouped, Abundances Grouped Count, Abundances Grouped CV in Percent, Abundances Normalized, Abundances Scaled, Charge by Search Engine, Checked, Concatenated Rank by Search Engine, Confidence by Search Engine, Delta Cn by Search Engine, Delta M in ppm by Search Engine, Delta mz in Da by Search Engine, Delta Score by Search Engine, Master Protein Accessions, Master Protein Descriptions, Modification Pattern, Modifications, mz in Da by Search Engine, Number of Isoforms, Number of Missed Cleavages, Number of Protein Groups, Number of Proteins, Number of PSMs, Peptide Isoforms Peptide Group ID, Percolator PEP by Search Engine, Percolator q-Value by Search Engine, Percolator SVMScore by Search Engine, Protein Accessions, PSM Ambiguity, Quan Info, Rank by Search Engine, RT in min by Search Engine, Search Engine Rank by Search Engine, Sequence, Sequence Length, Theo MHplus in Da, XCorr by Search Engine

Table 192. Paragraphs example

Table	Column Names
Isoforms	Accession, Biological process, Calc pi, Cellular component, Checked, Description, Ensembl gene id, Entrez gene id, Fasta title lines, Gene symbol, Go accessions, Isoforms unique sequence id, Kegg pathway accessions, Kegg pathways, Molecular function, Mw in kda, Number of aas, Number of characterized proteoforms, Number of protein annotation groups, Number of protein pathway groups, Number of proteoforms, Number of prsms, Pfam ids, Proteoform characterization confidence, Q-value, Sequence
Proteoforms	Average PrSM Detected Neutral Mass, Best PrSM C-Score, Charge by Search Engine ProSightPD Absolute Mass Search, Charge by Search Engine ProSightPD BioMarker Search, Charge by Search Engine ProSightPD Gene Restricted Absolute Mass Search, Charge by Search Engine ProSightPD Gene Restricted BioMarker Search, Checked, Confidence, Confidence by Search Engine ProSightPD Absolute Mass Search, Confidence by Search Engine ProSightPD BioMarker Search, Confidence by Search Engine ProSightPD Gene Restricted Absolute Mass Search, Confidence by Search Engine ProSightPD Gene Restricted BioMarker Search, Delta Mass in Da by Search Engine ProSightPD Absolute Mass Search, Delta Mass in Da by Search Engine ProSightPD BioMarker Search, Delta Mass in Da by Search Engine ProSightPD Gene Restricted Absolute Mass Search, Delta Mass in Da by Search Engine ProSightPD Gene Restricted BioMarker Search, Delta Mass in ppm by Search Engine ProSightPD Absolute Mass Search, Delta Mass in ppm by Search Engine ProSightPD BioMarker Search, Delta Mass in ppm by Search Engine ProSightPD Gene Restricted Absolute Mass Search, Delta Mass in ppm by Search Engine ProSightPD Gene Restricted BioMarker Search, External Top Down Displays, -Log E-Value by Search Engine ProSightPD Absolute Mass Search, -Log E-Value by Search Engine ProSightPD BioMarker Search, -Log E-Value by Search Engine ProSightPD Gene Restricted Absolute Mass Search, -Log E-Value by Search Engine ProSightPD Gene Restricted BioMarker Search, -Log P-Score by Search Engine ProSightPD Gene Restricted Absolute Mass Search, -Log P-Score by Search Engine ProSightPD Gene Restricted BioMarker Search, mz in Da by Search Engine ProSightPD Absolute Mass Search, mz in Da by Search Engine ProSightPD BioMarker Search, mz in Da by Search Engine ProSightPD Gene Restricted Absolute Mass Search, mz in Da by Search Engine ProSightPD Gene Restricted BioMarker Search, Number of Proteins, Number of PrSMs, Percent Residue Cleavages, Protein Accessions, Proteoform Characterization Confidence, Proteoforms Proteoform Group ID, Q-value, Rank by Search Engine ProSightPD Absolute Mass Search, Rank by Search Engine ProSightPD BioMarker Search, Rank by Search Engine ProSightPD Gene Restricted Absolute Mass Search, Rank by Search Engine ProSightPD Gene Restricted BioMarker Search, RT in min by Search Engine ProSightPD Absolute Mass Search, RT in min by Search Engine ProSightPD BioMarker Search, RT in min by Search Engine ProSightPD Gene Restricted Absolute Mass Search, RT in min by Search Engine ProSightPD Gene Restricted BioMarker Search, Search Engine Rank by Search Engine ProSightPD Absolute Mass Search, Search Engine Rank by Search Engine ProSightPD BioMarker Search, Search Engine Rank by Search Engine ProSightPD Gene Restricted Absolute Mass Search, Search Engine Rank by Search Engine ProSightPD Gene Restricted BioMarker Search, Sequence, Sequence Length, Theo Mass in Da