



**Software Tool House Inc**

---

**Software Tool House Inc.**

**Meta-Update**

**Job Console**

**User's Guide**

---

© 2025 Software Tool House Inc.  
Release 6.25  
Updated: 2025-Jan-10



# Preface

## Audience

This document is intended for Remedy ARS and/or ServiceNow Administrators and developers.

It is expected that the reader will have knowledge of the Remedy ARS system and be familiar with workflow development. It would behoove the reader to be familiar with his ARS server's platform and scripting tools.

## Limitation of Liability

This program is provided "as-is". We are in no way liable for any losses arising from your use of this program, the sample scripts, or the documentation. It is your responsibility to evaluate this program. It is your responsibility to backup and protect your data. It is your responsibility to evaluate your use of this program for any particular purpose.

This manual does not represent a commitment to maintain any syntax or operation, nor is it warranted to be complete or accurate.

## Copyrights

This program and this manual are copyrighted © 1996-2025 by Software Tool House Inc. Meta-Layer, Meta-Update, Meta-Query, Meta-Delete, Meta-Schema and Meta-Archive are trademarks of Software Tool House Inc.

ARS, Remedy are registered trademarks of BMC Corporation.

ServiceNow is a registered trademark of ServiceNow, Inc.

Solaris is a registered trademark of Sun Microsystems Inc.

Windows is a registered trademark of Microsoft Corporation.

PCRE (Perl Compatible Regular Expression) library is copyrighted © 1997 – 2025 by University of Cambridge and is distributed under the BSD license.

The curl library is copyrighted © 1996 – 2025 by daniel@haxx.se and is distributed under a MIT/X derivative license.

## Updates

This program and this manual may change from time to time. The latest version is available at our web site: [www.softwaretoolhouse.com](http://www.softwaretoolhouse.com).

## Comments

Your comments are welcome! Please see: [www.softwaretoolhouse.com/support](http://www.softwaretoolhouse.com/support) and click **Comments**, or email us at [support@softwaretoolhouse.com](mailto:support@softwaretoolhouse.com). We look forward to hearing from you!



## Document Library

The following documents are included with Meta-Update.

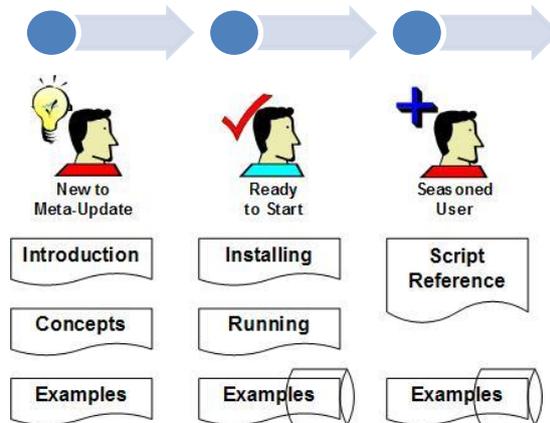
File	Contents
Meta-Update Installation Guide	Meta-Update and the Job Console installation guide.
Meta-Update Users Guide	This is a detailed reference on Meta-Update scripting. It is used by script developers. It covers developing and debugging scripts.
Meta-Update Samples Guide	This is a detailed reference on many of the Meta-Update sample scripts. The samples do useful things and this document can be used for learning Meta-Update scripting. Templates for the samples are installed with the Job Console application.
Meta-Update Job Console Users Guide	This is a detailed reference on developing templates and firing jobs using the Job Console.
<i>This document.</i>	
Meta-Update Release Notes	This highlights changes made in this release of Meta-Update.
Trace Daemon Users Guide	The "Trc" version of the binaries communicate with a process called the trace daemon. This is the User Guide for this.
Meta-Update Release Notes	This highlights changes made in this release of Meta-Update.

## Organisation

This document is divided into a sequence of sections.

The diagram to the right outlines which User's Guide sections we recommend as you use Meta-Update more and more.

Here's an overview of the User's Guide sections:



### Introduction

This is a short description of what Meta-Update is and the functions and usage of the Job Console. Read it if you are new to Meta-Update, or want to learn what Meta-Update does at a high level. You do not need to be an ARS Administrator or developer to read this section.

### Concepts

It introduces the Meta-Update scripting language and gives a general understanding of what to do to implement a script. It is required reading until familiar with Meta-Update scripting.

### Installing

The installation notes for UNIX and Windows.

### Running

Running Meta-Update and interpreting the output. Developing and debugging scripts.

### Script Reference

This explains how to code Meta-Update scripts in detail. You will need to refer to this section when you don't remember how a specific setting or option is specified.

### Licensing

This explains how the Meta-Update server based licensing works.

### Samples

These samples, which are part of the software distribution and available on the web, are another way to learn how to build scripts. These samples perform different functions and have detailed explanations of the Meta-Update scripts used. Bits of these scripts may be copied and tailored to your scripts.

## Document Conventions

Typefaces and conventions and icons are used in this document to add specific meaning as follows:

Icon & Type Conventions	Meaning
	Windows specific. Does not apply to Linux.
	Linux specific. Does not apply to Windows.
	Applies to BMC Remedy ARS server sessions. Cannot be used for, or Does not refer to, ServiceNow sessions..
	Applies to ServiceNow sessions. Cannot be used for , or does not refer to BMC Remedy ARS server sessions.
	Caution. Failure to follow recommended actions may cause data loss.
<p>Courier Bold</p>	<p>Courier Bold indicates a command you can enter. For example:</p> <pre>  <code>set SthApiRetry=90-92 0 60 93 0 30</code>  <code>export SthApiRetry=90-92 0 60 93 0 30</code> </pre>

## Table of Contents

Preface .....	2
Document Library .....	4
Organisation .....	5
Document Conventions.....	6
Table of Contents.....	7
Introduction.....	9
Concepts .....	11
Overview .....	12
Users .....	15
The Job Console Form.....	16
Overview .....	17
The Job Form .....	23
Overview .....	24
The Job Template Form .....	29
Overview .....	30
Index .....	50



# Introduction





## Concepts

# Concepts

## Overview

Meta-Update Job Control is a BMC Remedy Application.

The purpose of Job Control is to fire and control Meta-Update scripts and other Jobs and to communicate the results back to the user that fired the job.

Meta-Update script files must be readable by a Queue Server. A Queue Server can be started on a Remedy server or any other machine with IP access to the Remedy Server.

Meta-Update scripts can be developed and tested on any workstation or server.

Once tested and promoted, they are copied to the Queue Servers. This can be under a Source Version control system.

Please see the [Meta-Update User's Guide](#) for a complete reference to Meta-Update scripting.

All Meta-Update Job Console users must be Remedy Administrators and must be set to allow Meta-Update use and rights.

**Job Templates** are defined for available scripts.

A job template specifies a script, a script path, and required or optional arguments.

Different templates may be created for a single script with different argument values and defaults.

A **Job** is created from a **Template**. Arguments are checked and once ready, a job can be **fired**.

A fired job enters a **Job Queue** and waits its turn.

A **Job Queue Process** may be run on a Remedy server, another server, or any workstation with connectivity to the Remedy server.

Each **Job Queue Server** uses a configurable number of slots to run jobs simultaneously. That way, even with long running jobs, other jobs will continue.

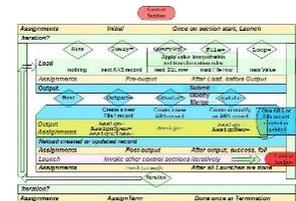


Figure 4 Flow of a Meta-Update section

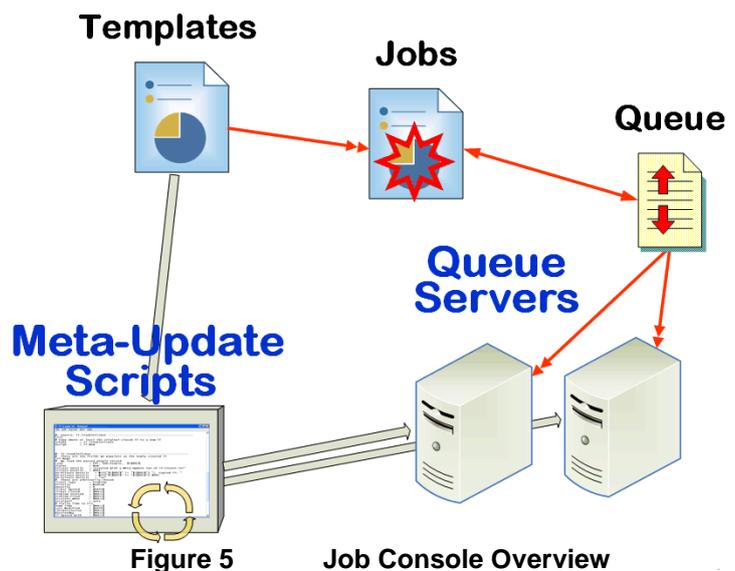
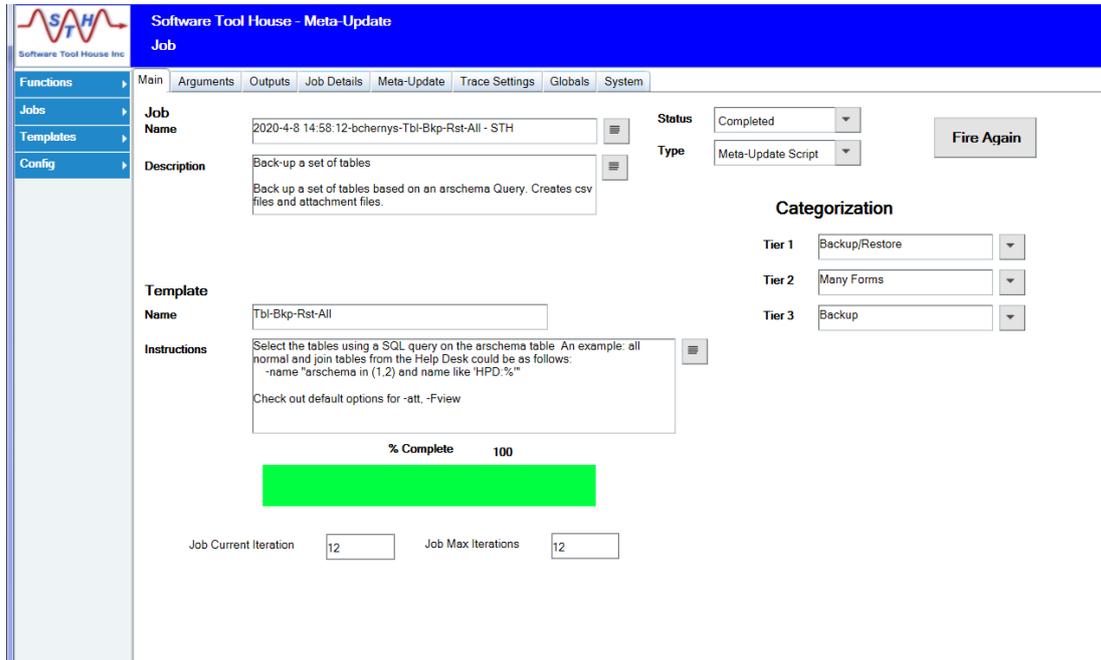


Figure 5

Job Console Overview

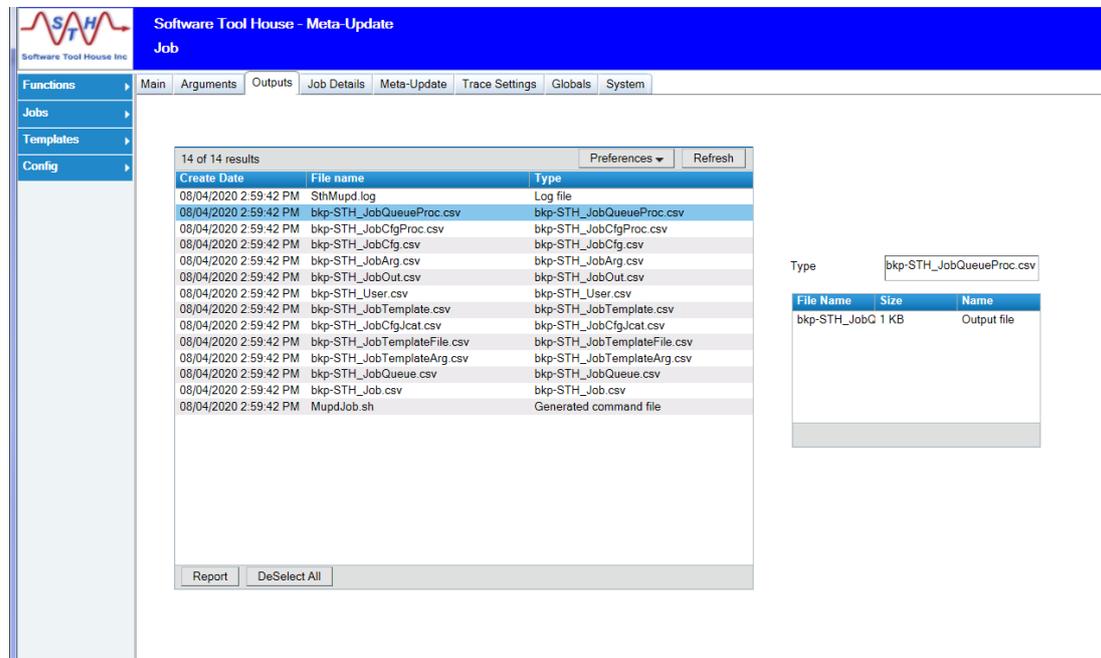
The primary interface for users is the **Job**. If the script is so configured, progress bars indicate a Job's status.



The screenshot shows the 'Job' configuration page in the Software Tool House - Meta-Update application. The job is titled '2020-4-8 14:58:12-bchernys-Tbl-Bkp-Rst-All - STH' and has a status of 'Completed'. The description indicates it's a backup script. The progress bar is fully green, showing 100% completion. The 'Fire Again' button is visible, along with categorization options for Tier 1 (Backup/Restore), Tier 2 (Many Forms), and Tier 3 (Backup).

**Figure 6 A Completed Job Showing Progress Bar**

When a Job completes, **Output Files** are loaded as attachments and the slot is free to run another job. These files may also remain on the server at the control of the job Template.

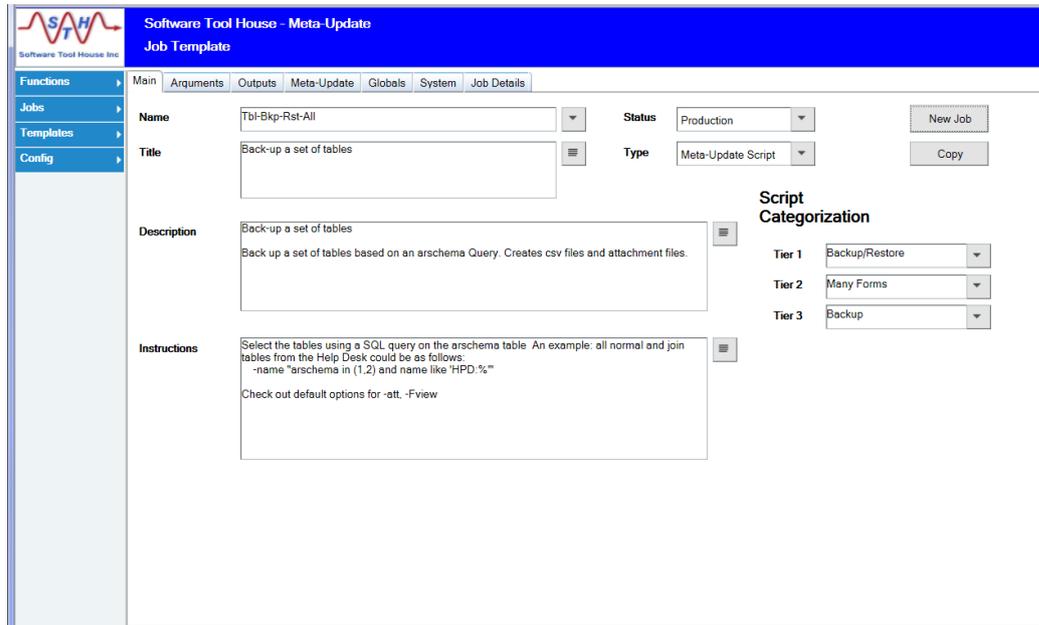


The screenshot shows the 'Outputs' tab of the job configuration page. It displays a table with 14 results, including file names, sizes, and types. A 'Report' button and a 'DeSelect All' button are at the bottom of the table. A preview of the selected file 'bkp-STH\_JobQueueProc.csv' is shown on the right.

Create Date	File name	Type
08/04/2020 2:59:42 PM	SthMupd.log	Log file
08/04/2020 2:59:42 PM	bkp-STH_JobQueueProc.csv	bkp-STH_JobQueueProc.csv
08/04/2020 2:59:42 PM	bkp-STH_JobCfjProc.csv	bkp-STH_JobCfjProc.csv
08/04/2020 2:59:42 PM	bkp-STH_JobCfj.csv	bkp-STH_JobCfj.csv
08/04/2020 2:59:42 PM	bkp-STH_JobArg.csv	bkp-STH_JobArg.csv
08/04/2020 2:59:42 PM	bkp-STH_JobOut.csv	bkp-STH_JobOut.csv
08/04/2020 2:59:42 PM	bkp-STH_User.csv	bkp-STH_User.csv
08/04/2020 2:59:42 PM	bkp-STH_JobTemplate.csv	bkp-STH_JobTemplate.csv
08/04/2020 2:59:42 PM	bkp-STH_JobCfjJcat.csv	bkp-STH_JobCfjJcat.csv
08/04/2020 2:59:42 PM	bkp-STH_JobTemplateFile.csv	bkp-STH_JobTemplateFile.csv
08/04/2020 2:59:42 PM	bkp-STH_JobTemplateArg.csv	bkp-STH_JobTemplateArg.csv
08/04/2020 2:59:42 PM	bkp-STH_JobQueue.csv	bkp-STH_JobQueue.csv
08/04/2020 2:59:42 PM	bkp-STH_Job.csv	bkp-STH_Job.csv
08/04/2020 2:59:42 PM	MupdJob.sh	Generated command file

**Figure 7 Job Output Files**

The primary interface for administrators and script developers is the **Job Template**.



**Figure 8 Job Template**

A single **Job Template** specifies a Meta-Update script, its location on a **Queue Server**, arguments, standard script features used, default trace levels, and help information.

Each **Job** is fired by a **Job Template**.

**Job Templates** may also define jobs running Meta-Schema and Meta-Query as well as other batch or binaries..

Some jobs will communicate progress to Job Control. Scripts need to enable this.

Job Control users must be Remedy administrators. These are defined in a form and only those users so defined are allowed to use Job Control.

There several roles available of users of the Meta-Update Job Console:

- |                   |   |
|-------------------|---|
| Script Manager    | Can place templates into production usage. Can modify script permissions and categorizations. |
| Script User Admin | Specifies users and groups rules for the Job Console  |
| Script Developer  | Can modify scripts on the server and build templates.   |
| Script User       | Can run scripts   |

## Users

Meta-Update Queue Servers Login Names must be Remedy Administrators. The Meta-Update Job Console application additionally enforces Role based rights through workflow. Remedy.

Some common BMC Remedy functions require Administrator rights. For example Query SQL.

It is possible for non-administrators to execute some of the sample scripts, specifically those that don't do SQL queries.

For this reason, the Meta-Update Server Queue runs under a fixed license and Administrator Remedy User Name.

It is often desirable to have a specific user used for Meta-Update functions. Multiple Queue servers can be started on multiple machines using this common login.

Meta-Update Jobs, however, run under the User that submitted them.

All Job Console users must be defined in a form. They must be Remedy Administrators as mentioned above.

None the less they are assigned Roles:

➤	User Administrator	Can edit users, edit templates, fire jobs
➤	Catalog Administrator	Can edit various configuration catalogues
➤	Template Developer	Can edit templates and fire jobs
➤	Job Submitted	Can fire jobs by selecting templates

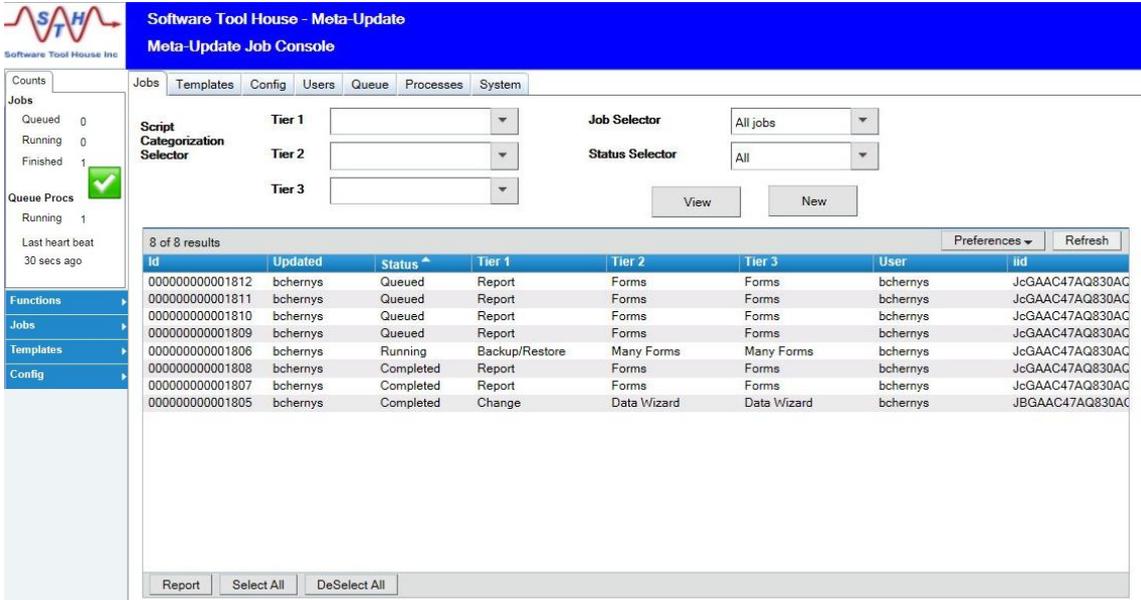
User Id that installs the Job Console application the first time is automatically set to be a Meta-Update Job Console user with all roles checked.



# The Job Console Form

## Form Overview

Meta-Update Job Console form accesses all application functions in different tabs.



Software Tool House - Meta-Update  
Meta-Update Job Console

Jobs Templates Config Users Queue Processes System

Script Categorization Selector: Tier 1, Tier 2, Tier 3

Job Selector: All jobs

Status Selector: All

View New

8 of 8 results

Id	Updated	Status	Tier 1	Tier 2	Tier 3	User	iid
00000000001812	bchernys	Queued	Report	Forms	Forms	bchernys	JcGAAC47AQ830AC
00000000001811	bchernys	Queued	Report	Forms	Forms	bchernys	JcGAAC47AQ830AC
00000000001810	bchernys	Queued	Report	Forms	Forms	bchernys	JcGAAC47AQ830AC
00000000001809	bchernys	Queued	Report	Forms	Forms	bchernys	JcGAAC47AQ830AC
00000000001806	bchernys	Running	Backup/Restore	Many Forms	Many Forms	bchernys	JcGAAC47AQ830AC
00000000001808	bchernys	Completed	Report	Forms	Forms	bchernys	JcGAAC47AQ830AC
00000000001807	bchernys	Completed	Report	Forms	Forms	bchernys	JcGAAC47AQ830AC
00000000001805	bchernys	Completed	Change	Data Wizard	Data Wizard	bchernys	JcGAAC47AQ830AC

Report Select All DeSelect All

**Figure 9 Jobs tab: Queued, Running, Completed jobs.**

Tabs are shown when the user is defined and has the appropriate role.

Tabs present:

- Jobs in the system in any state: Queued, Running, Completed
- Templates available in the system
- Users users and roles defines that can use the Job Console
- Config
- Queue Jobs in the Queue
- Processes Running Queue processes

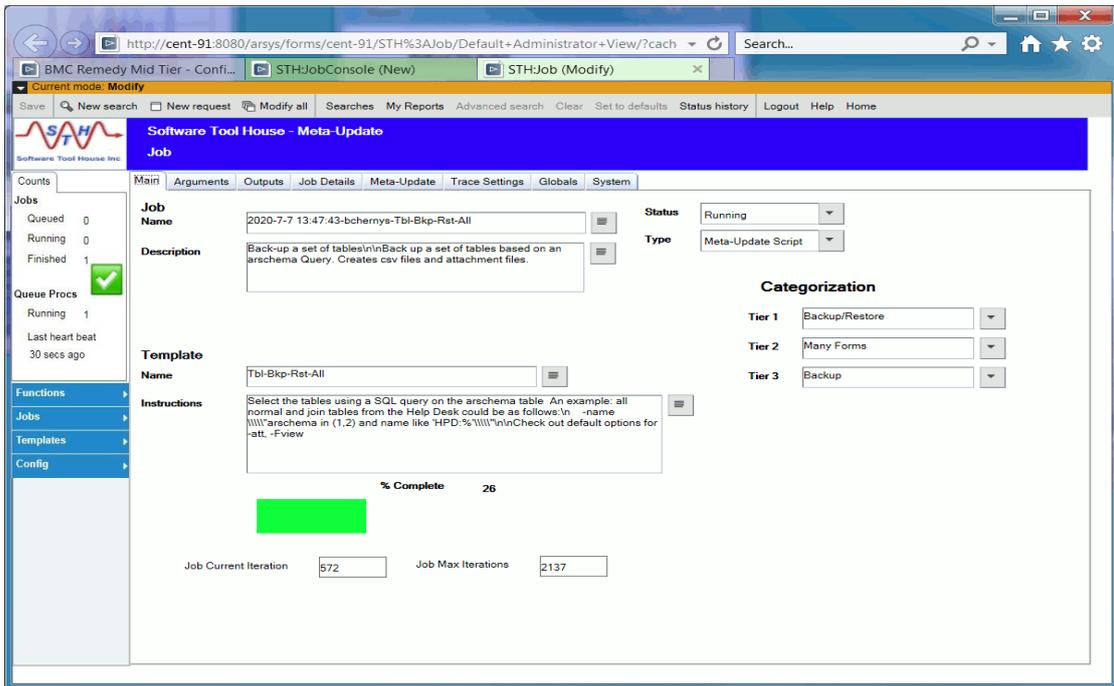
## Jobs Tab

Jobs to appear in the table may be selected through the fields available.

By default jobs are sorted first by status then by id in descending order, effectively, in reverse time order with the newest at the top and the oldest at the bottom. The order of the Status is such that Queued (waiting) jobs are first and completed jobs are last.

Selecting a job and pressing the View button (or Double clicking) will bring up that Job's record which will include all input and output files.

A running job may display its progress which will change.



The owner of the job may choose to delete the job's files on the record or the directory from either the job or the console. Note that only completed jobs may be deleted.

Selecting New in the console will let you choose a template and create a new job from it.

Alternatively, the Templates tab will allow you to click a template and create a new job saving a click, selecting a template in a dialogue and clicking Ok.

## Templates Tab

Templates to appear in the table may be selected through the fields available.

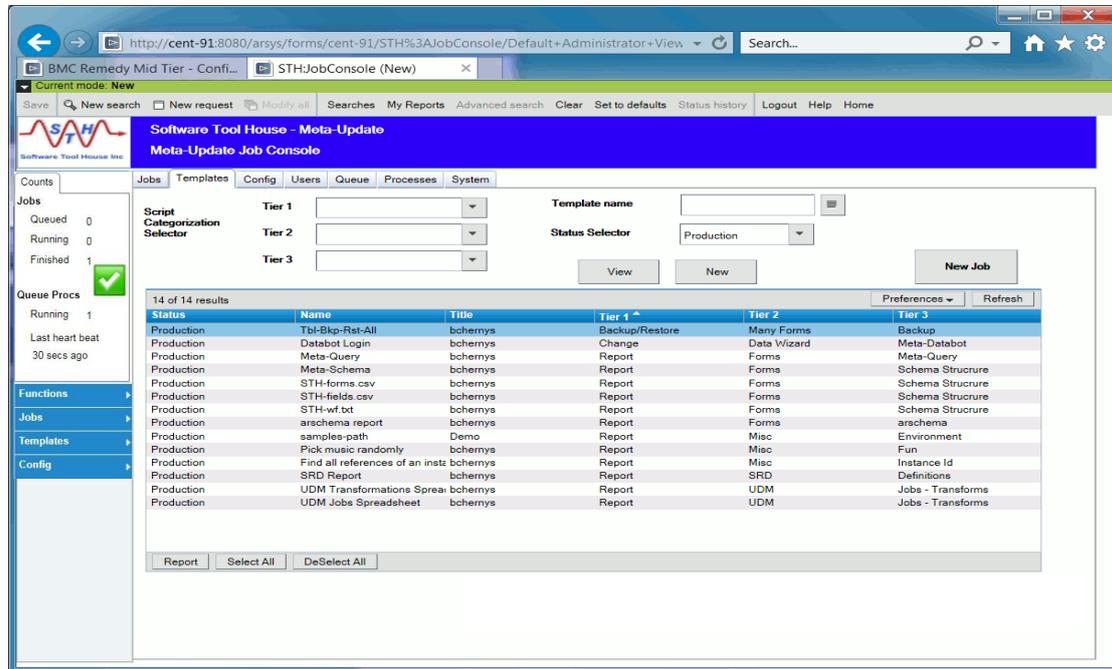


Figure 10 Templates Tab

By default Templates in Production are displayed.

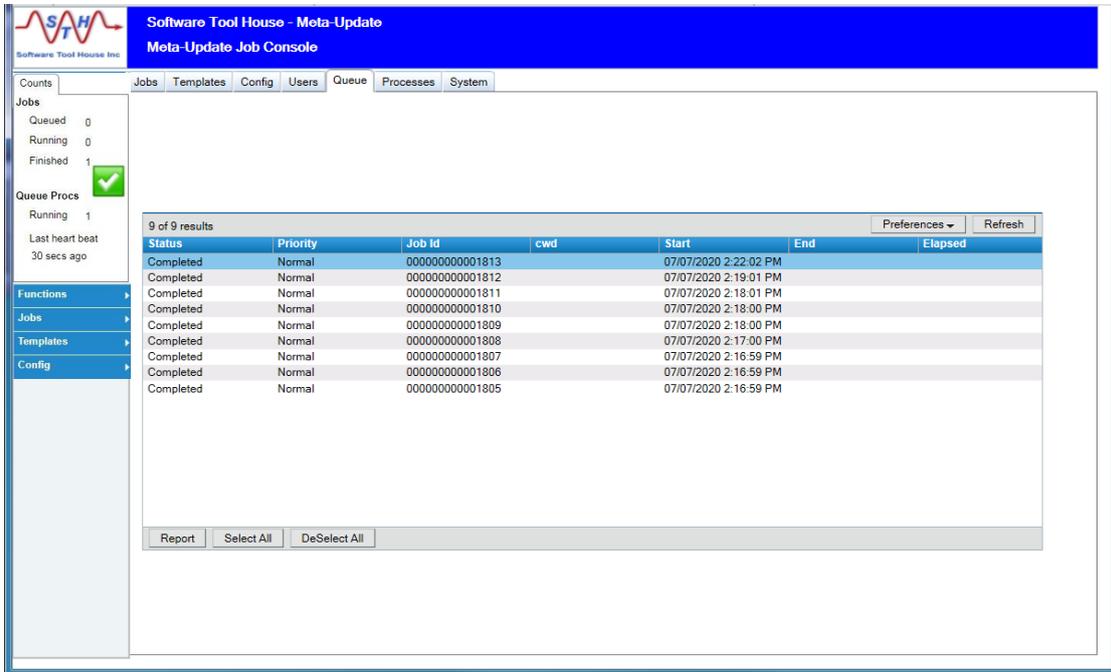
Double clicking a Template will create a new Job for users with the role Job:Submitter. For Template Developers, the Template will be opened.

Templates can be used to fire scripts with different arguments and groups of different job templates.

A Template is deemed Complete when all required arguments are filled in. Only Complete Templates may be grouped as a single job.

## Queue Tab

There is one Queue record for each Job record.



Software Tool House - Meta-Update  
Meta-Update Job Console

Counts: Queued 0, Running 0, Finished 1

Queue Procs: Running 1, Last heart beat 30 secs ago

9 of 9 results

Status	Priority	Job Id	cwd	Start	End	Elapsed
Completed	Normal	00000000001813		07/07/2020 2:22:02 PM		
Completed	Normal	00000000001812		07/07/2020 2:19:01 PM		
Completed	Normal	00000000001811		07/07/2020 2:18:01 PM		
Completed	Normal	00000000001810		07/07/2020 2:18:00 PM		
Completed	Normal	00000000001809		07/07/2020 2:18:00 PM		
Completed	Normal	00000000001808		07/07/2020 2:17:00 PM		
Completed	Normal	00000000001807		07/07/2020 2:16:59 PM		
Completed	Normal	00000000001806		07/07/2020 2:16:59 PM		
Completed	Normal	00000000001805		07/07/2020 2:16:59 PM		

Report Select All DeSelect All

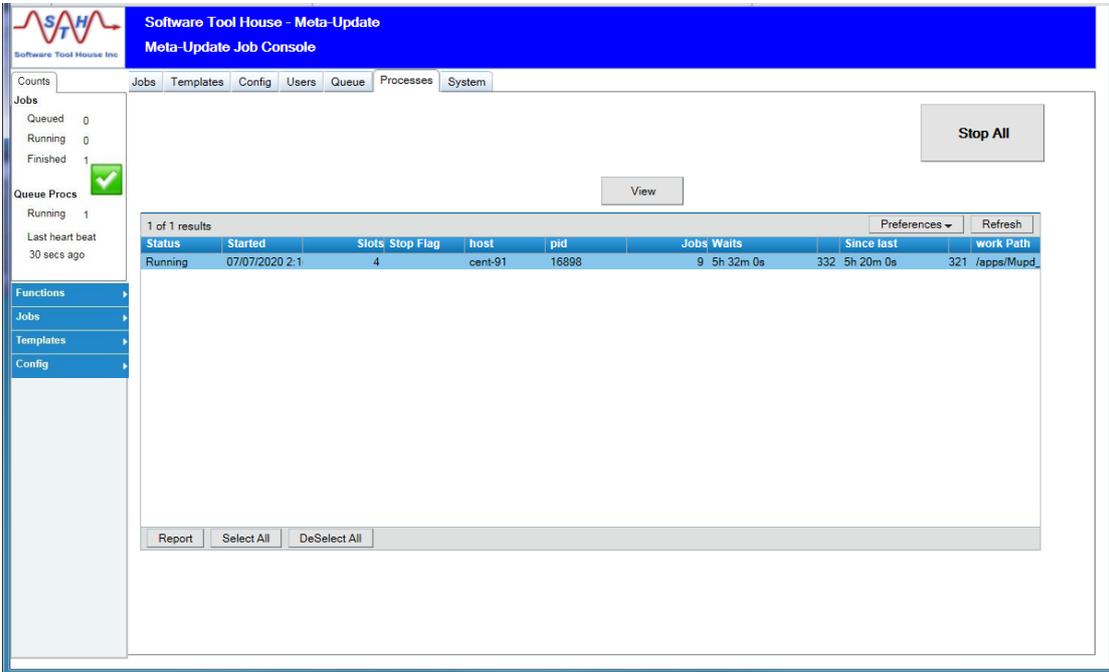
Figure 11 Queue Tab

After a Job completes, the results are loaded into the Job record, and then the Queue record is marked as completed.

There are also Queue records for Server work area deletions. When the queue server processes these, the Job's work area on the server is marked as deleted and the Queue record for both the Job and the Deletion are deleted.

## Processes Tab

There is one Running process record for Queue process started on a host.



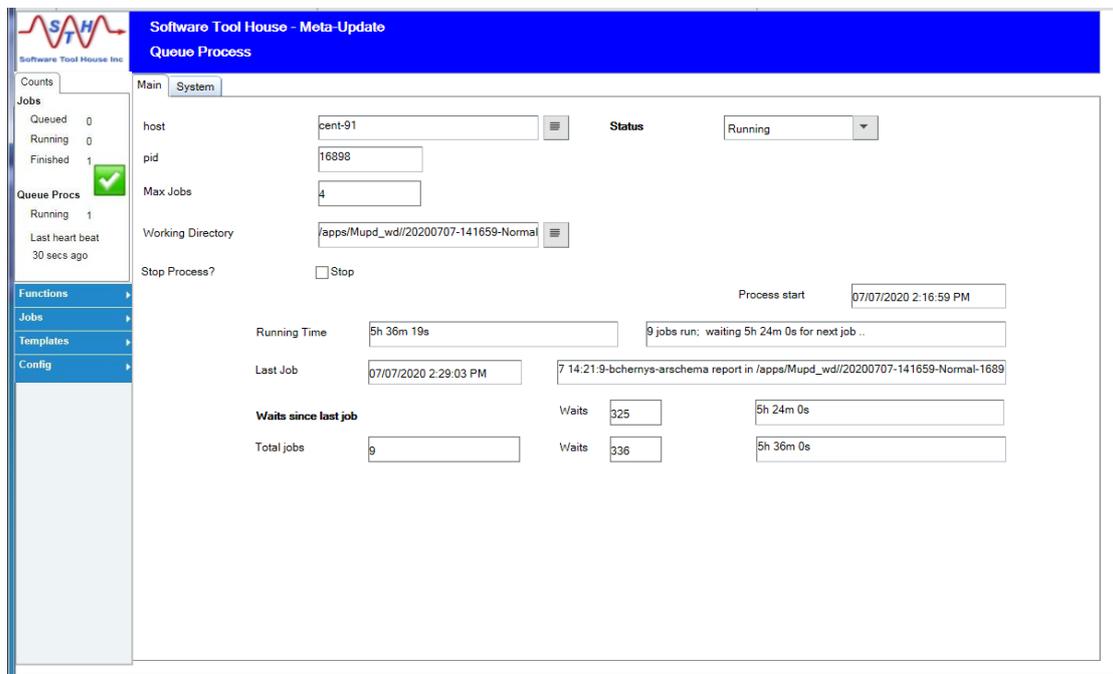
The screenshot shows the 'Processes' tab in the 'Meta-Update Job Console'. On the left, there are 'Counts' for Jobs (Queued: 0, Running: 0, Finished: 1) and Queue Procs (Running: 1). A 'View' button is present above a table with 1 result. The table has columns: Status, Started, Slots, Stop Flag, host, pid, Jobs, Waits, Since last, and work Path. The single record is: Running, 07/07/2020 2:1, 4, cent-91, 16898, 9 5h 32m 0s, 332 5h 20m 0s, 321 /apps/Mupd. There are also 'Report', 'Select All', and 'DeSelect All' buttons at the bottom.

Status	Started	Slots	Stop Flag	host	pid	Jobs	Waits	Since last	work Path
Running	07/07/2020 2:1	4		cent-91	16898	9	5h 32m 0s	332 5h 20m 0s	321 /apps/Mupd

Figure 12 Queue processes

When a Queue process is started it creates its own record. As it fires jobs and waits for more jobs or for jobs to complete it continuously updates its own record.

Double clicking will open details of the record:



The screenshot shows the 'Queue Process' details view. It includes fields for host (cent-91), pid (16898), Max Jobs (4), and Working Directory (/apps/Mupd\_wd/20200707-141659-Normal). The status is 'Running'. The process start time is 07/07/2020 2:16:59 PM. The running time is 5h 36m 19s, and it shows 9 jobs run, waiting 5h 24m 0s for the next job. The last job is 07/07/2020 2:29:03 PM, with a wait time of 7 14:21:9-bchernys-arschema report in /apps/Mupd\_wd/20200707-141659-Normal-1689. There are also 'Waits since last job' and 'Total jobs' sections with wait times of 325 (5h 24m 0s) and 336 (5h 36m 0s) respectively.

Figure 13 A Running Queue Process



The Stop all button will cause all processes to Stop. No more jobs will be run.

If a process does is not stopped normally but fails to update its own Process record, an indicator is red is highlighted on the detail form.

## The Job Form

## Form Overview

Meta-Update Job form is opened from the Jobs or Templates tab of the Job Console.

A template defines the available Jobs. On the template tab, select a template and press the New Job button.

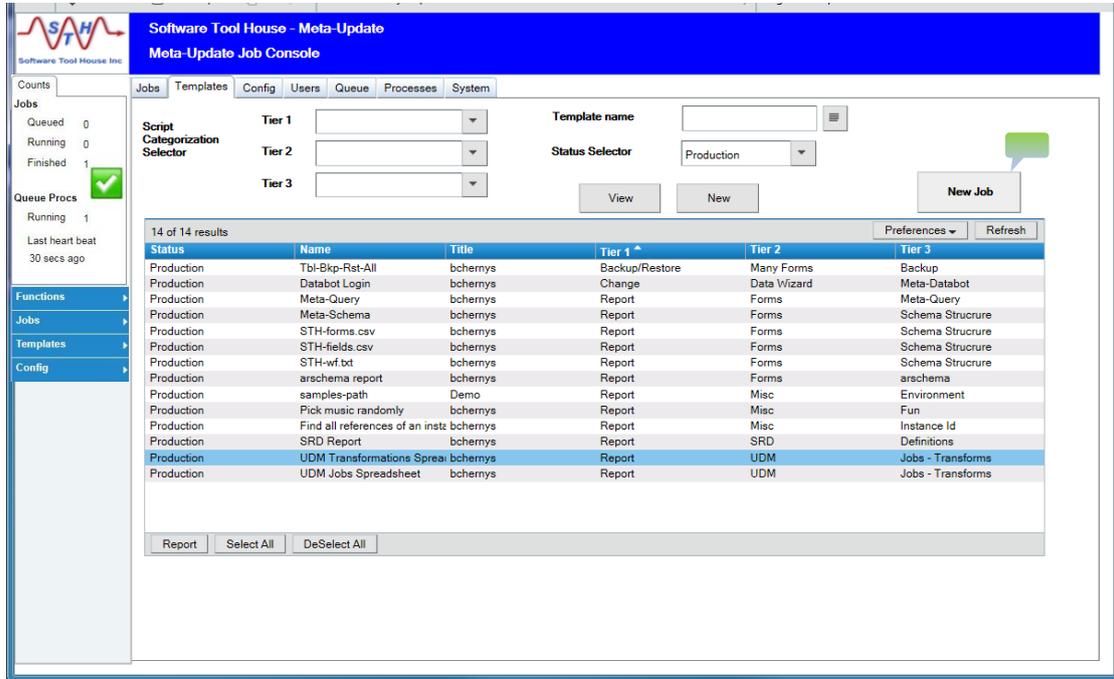


Figure 15 Template Selection to create a new Job

A new job will be created.

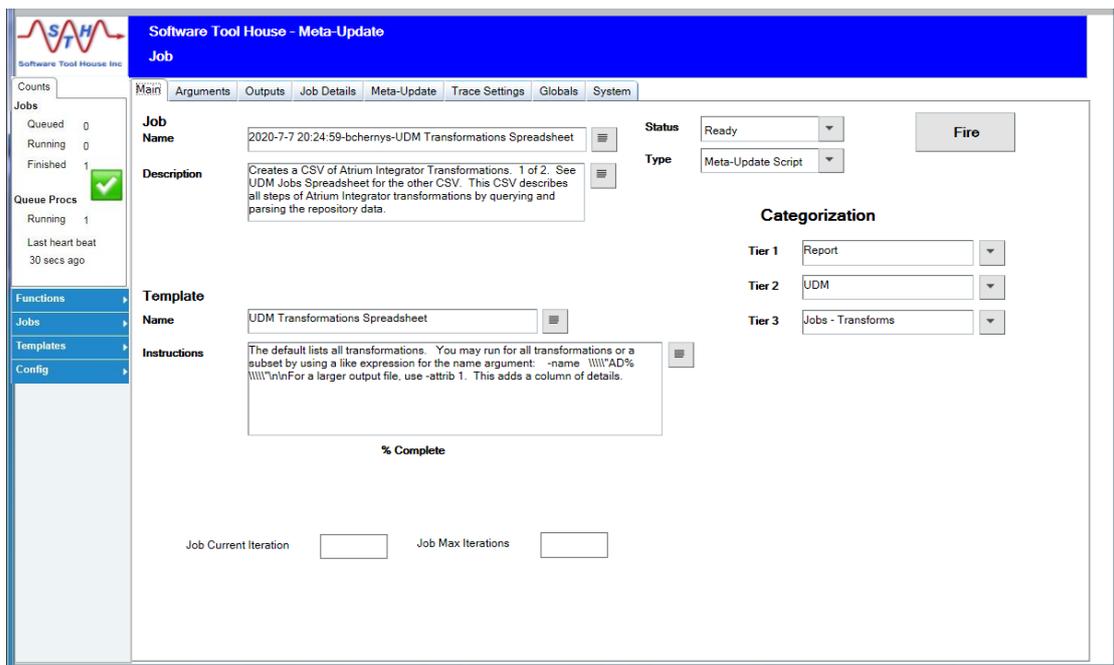


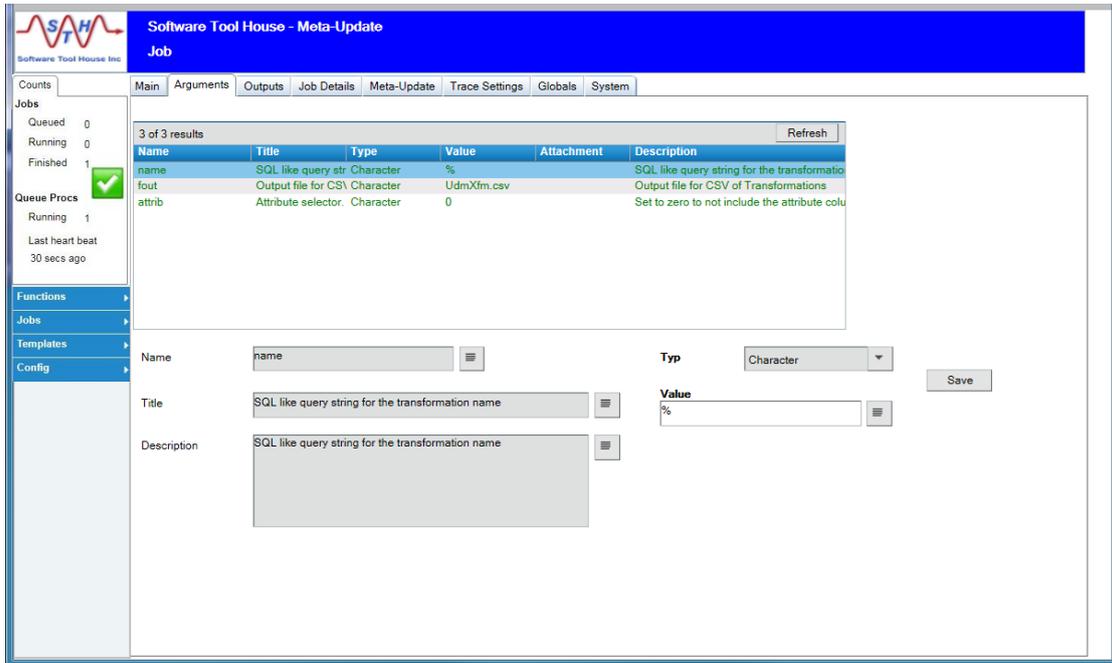
Figure 16 New Job Ready to Fire



## Setting a Jobs' Arguments

In the above image, the job is ready to fire. All required arguments are set to the defaults defined in the Template.

Other jobs will need some arguments filled in. You will be taken directly to the arguments tab.



The screenshot shows the 'Arguments' tab in the 'Software Tool House - Meta-Update Job' console. The interface includes a sidebar with navigation options like 'Jobs', 'Queue Procs', 'Functions', 'Jobs', 'Templates', and 'Config'. The main area displays a table with 3 results:

Name	Title	Type	Value	Attachment	Description
name	SQL like query str	Character	%		SQL like query string for the transformation
fout	Output file for CSV	Character	UdmXfm.csv		Output file for CSV of Transformations
attrib	Attribute selector	Character	0		Set to zero to not include the attribute colu

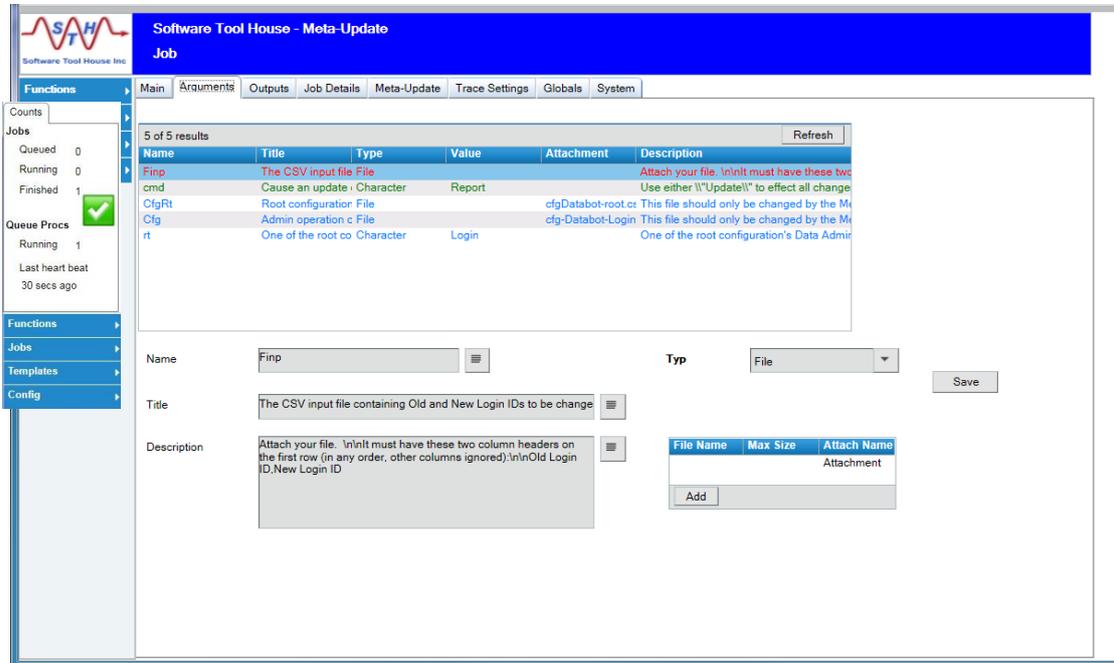
Below the table is a form for editing an argument. The 'Name' field is 'name', 'Type' is 'Character', and 'Value' is '%'. The 'Title' and 'Description' fields both contain 'SQL like query string for the transformation name'. A 'Save' button is located to the right of the form.

Figure 17 Arguments – all with default values

Arguments that are not Locked by the Template can be changed in this tab.

Select the argument row in the table, change the value, and press Save.

The next example is of a job with a required argument (in red) and other arguments with default values. The required argument is a file. Attach a file as described to the left and press Save.



The screenshot shows the 'Software Tool House - Meta-Update Job' console. On the left, there are navigation tabs for 'Functions', 'Jobs', 'Templates', and 'Config'. The main area displays a table of arguments with the following data:

Name	Title	Type	Value	Attachment	Description
Finp	The CSV input file	File			Attach your file. \n\nIt must have these two
cmd	Cause an update	Character	Report		Use either \"Update\" to effect all change
CfgRt	Root configurator	File		cfgDatabot-root.ct	This file should only be changed by the M
Cfg	Admin operation c	File		cfg-Databot-Login	This file should only be changed by the M
rt	One of the root co	Character	Login		One of the root configuration's Data Adm

Below the table, there is a form for editing the 'Finp' argument. The 'Name' field is 'Finp', and the 'Type' is 'File'. The 'Title' is 'The CSV input file containing Old and New Login IDs to be change'. The 'Description' is 'Attach your file. \n\nIt must have these two column headers on the first row (in any order, other columns ignored):\n\nOld Login ID,New Login ID'. There is an 'Add' button for attachments and a 'Save' button.

Figure 18 A required "File" argument in red.

Note that even if all arguments have defaults, you may have to change some. The example above has a default value for `-cmd Report`. It is the Meta-Databot application which allows you to run the script in two command modes. `Report` and `Update`.

Once you do the report and review the changes that would occur, you can use the "Fire Again" button on the job to generate a new job, change the `cmd` argument to `Update` and fire the job.

The Databot template's instructions describe this argument and includes documentation attachments.

Once you Fire a job, it enters the Job Queue. If the Job Queue Server is waiting, it won't start until the wait is completed. This will be under 60s.

Once a Job starts running, if the template and the script support it, a long running job can indicate its progress to the Job record. Refreshing the record will show the progress bar in action.

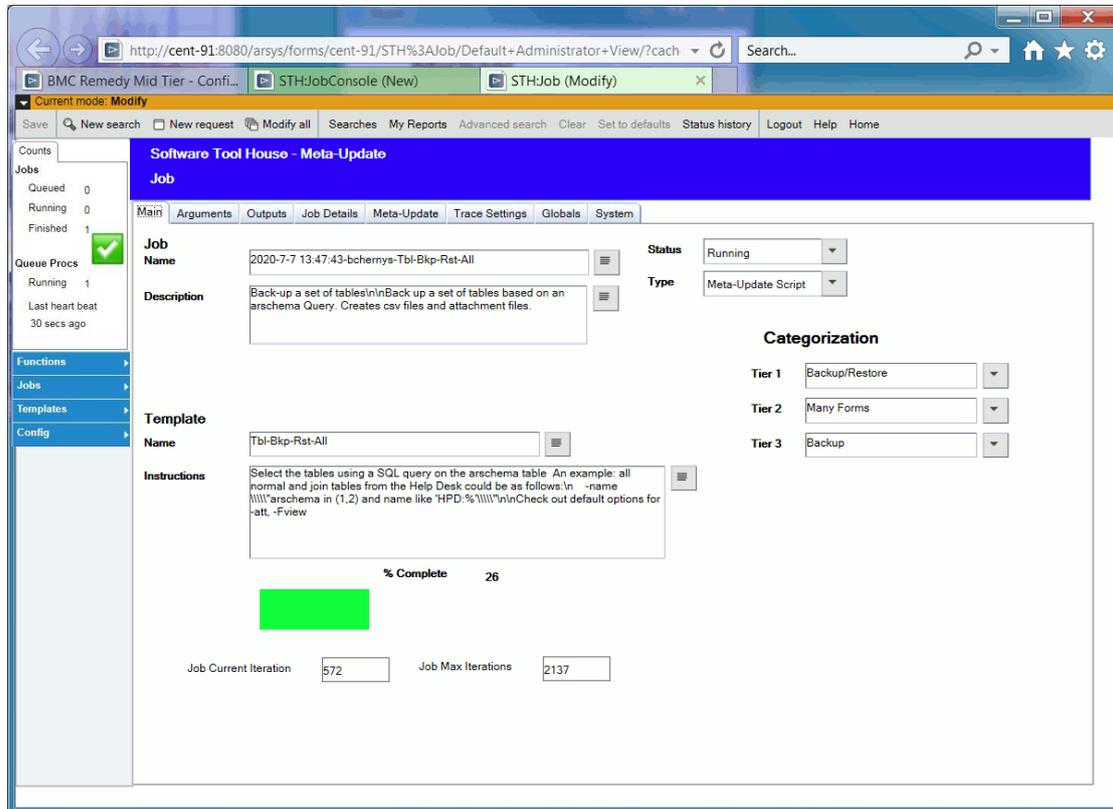
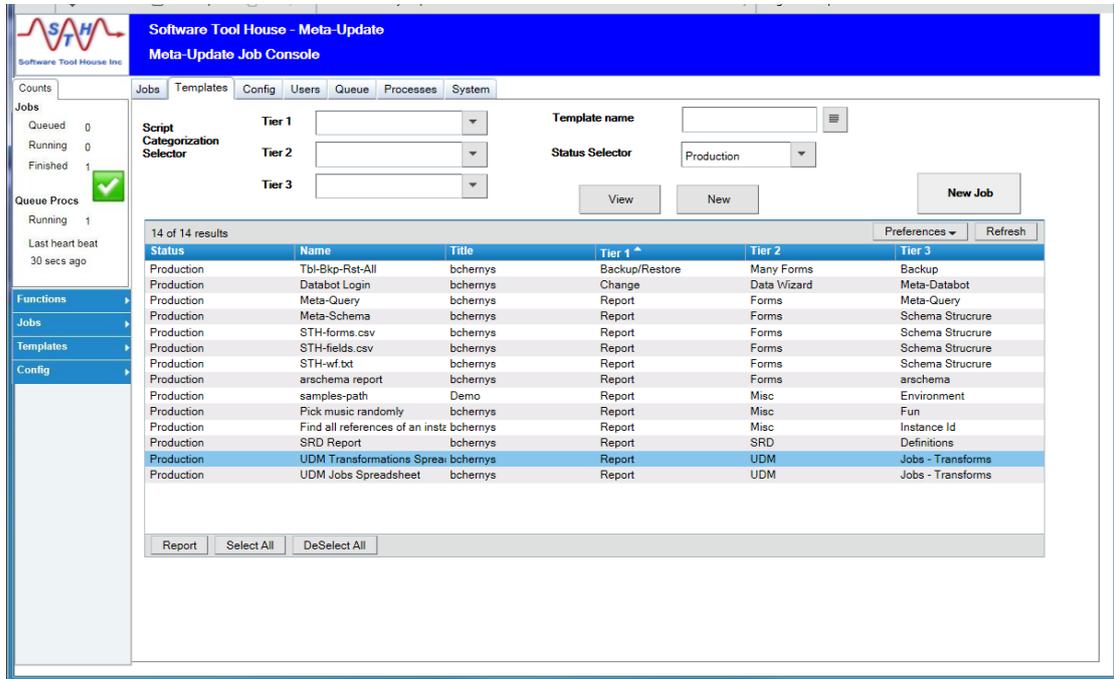


Figure 19 A long running job with a progress bar

# The Job Template Form

## Form Overview

Meta-Update Job Template form is opened from the Templates tab of the Job Console.



Software Tool House - Meta-Update  
Meta-Update Job Console

Counts: [ ] Jobs: Queued 0, Running 0, Finished 1, Queue Procs: Running 1, Last heart beat 30 secs ago

Script Categorization Selector: Tier 1 [ ], Tier 2 [ ], Tier 3 [ ]

Template name: [ ] Status Selector: Production

Buttons: View, New, New Job

Status	Name	Title	Tier 1 ^	Tier 2	Tier 3
Production	Tbi-Bkp-Rst-All	bchernys	Backup/Restore	Many Forms	Backup
Production	Databot Login	bchernys	Change	Data Wizard	Meta-Databot
Production	Meta-Query	bchernys	Report	Forms	Meta-Query
Production	Meta-Schema	bchernys	Report	Forms	Schema Structure
Production	STH-forms.csv	bchernys	Report	Forms	Schema Structure
Production	STH-fields.csv	bchernys	Report	Forms	Schema Structure
Production	STH-wf.txt	bchernys	Report	Forms	Schema Structure
Production	arschema report	bchernys	Report	Forms	arschema
Production	samples-path	Demo	Report	Misc	Environment
Production	Pick music randomly	bchernys	Report	Misc	Fun
Production	Find all references of an inst:	bchernys	Report	Misc	Instance Id
Production	SRD Report	bchernys	Report	SRD	Definitions
Production	UDM Transformations Sprea	bchernys	Report	UDM	Jobs - Transforms
Production	UDM Jobs Spreadsheet	bchernys	Report	UDM	Jobs - Transforms

Buttons: Report, Select All, DeSelect All

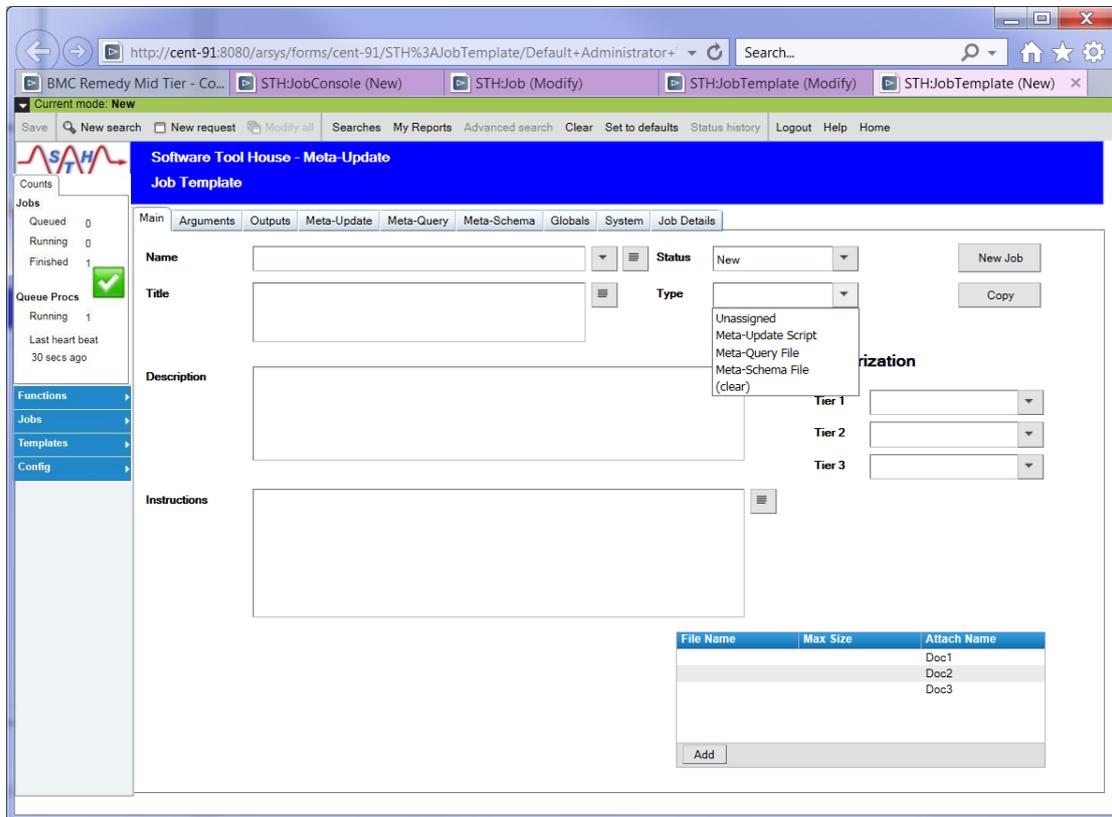
Figure 20 Template tab

On the template tab, click View to view or change a template or New to create a new template.

You are only allowed to create or modify templates with the **Template Developer** role.

A template is used to create a new job. It explains what the job does and defines all required and optional arguments for the job.

A template can be a Meta-Update script, a Meta-Schema or Meta-Query run. Additionally, a template can be a Multi-Job.



**Figure 22 A new template with the Type drop-down shown**

A Multi-Job template lists one or more other templates. When a Multi-Job template is run, all implied jobs are run as a single job, in sequence, and occupying one slot in the Queue Processor.



## Main tab

Different information is required depending on the Template's **Type** field. As each **Type** is selected, different Tabs come into view.

The main purpose of the Main tab is to offer documentation to the user on what the job does and how to set it up.

There are attachment slots for documents which are available to the job submitter. An example are the Meta-Datobot template samples.

A template is marked Job Complete if it requires no user input for a Job generated from that template. For a Meta-Update templates, all arguments must have defaults. For Meta-Query and Meta-Schema templates, all arguments must have values. Only Job Complete templates can be selected in a Multi-Job.

## Types of Templates

There are three types of templates:

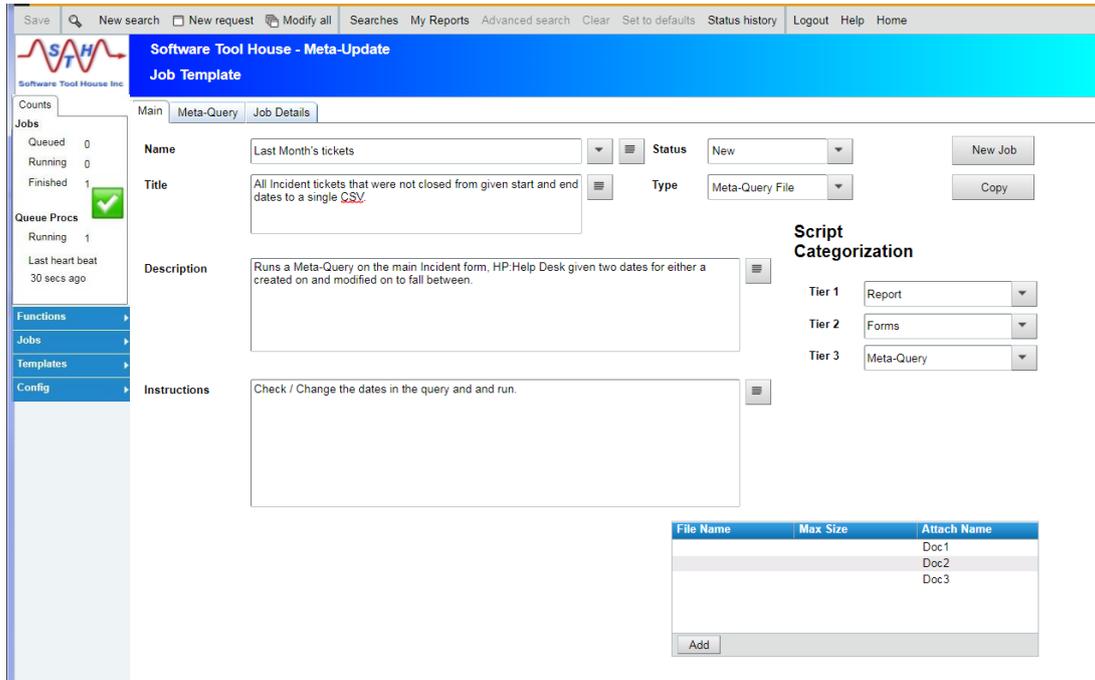
Meta-Update Script	Unlimited functionality based on a Meta-Update script. Defines arguments required by the script.
Meta-Query File	Used to generate a CSV file based on a query against a form or issue a direct SQL statement.
Meta-Schema File	Used to generate a CSV of forms and field definitions or text file of forms and workflow.
Multi-Job	Used to run multiple templates as a single job.

For each of these types, a single Type tab is opened and for all these types, the tab Job Details is also available.

## Meta-Query File

Meta-Query is used to generate a CSV of all records and fields from a form given a query. It can also issue direct SQL statements.

All fields are required on the Main panel except for Categorization (strongly suggested) and attachments.



**Figure 23 A new Meta-Query Template**

There are two Meta-Query **Types** available:

- Form** A Remedy (or ServiceNow) query against a form.
- sql** A Remedy executed direct SQL statement – usually a select.

## Form Type

A form query is a regular Remedy query with the addition that field names can be used between ticks.

Further, because of the nature of the command line, the entire query text must be enclosed in quotes. Any quotes embedded in the query text must be escaped with a back-slash.

Examples:

For CMDB Categorizations and classes:

```
Form:          PCT:Product Catalog
          "'Manufacturer' = \"\"
Or:          "'Manufacturer' = $NULL$
```

All open Incidents created last month:

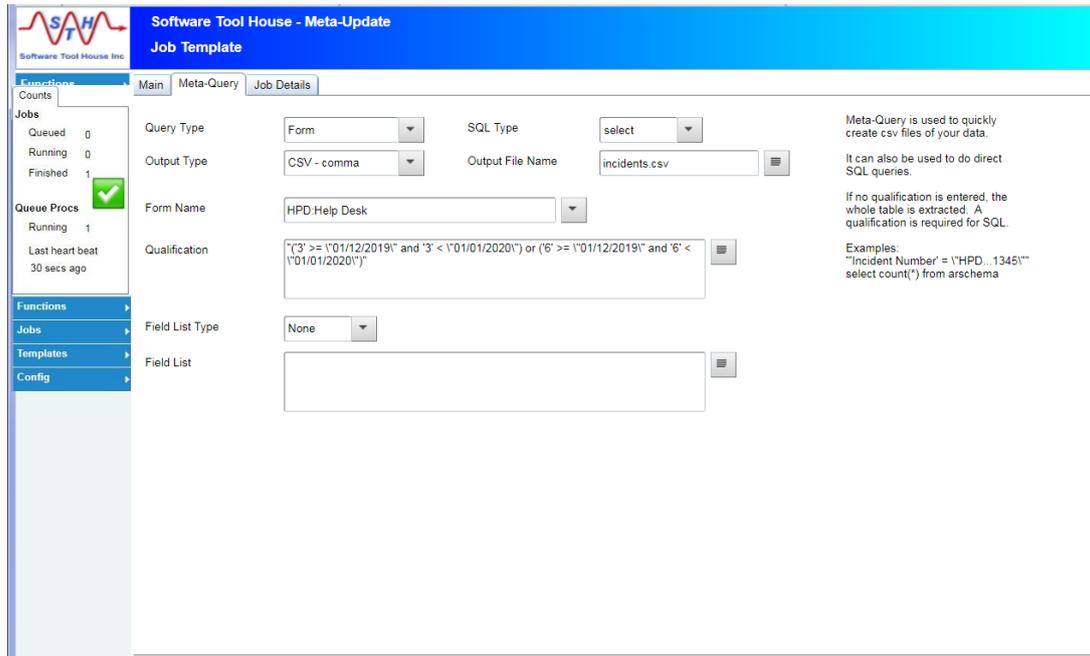
```
Form:          HPD:Help Desk
          "'Status' < \"Resolved\" and
          'Submit Date' >= \"01/01/2020\" and
          'Submit Date' < \"02/01/2020\""
```

If you want to parameterize a query, say automatically running a report for last month's incidents, we recommend the Meta-Update sample, `200-Mqry-dtes-prog.ini`, included in the distribution.

A query will always return all records in the results. A null query is the same as `1=1` and returns the complete table. Meta-Query will repeat the query in chunks if needed.

You can design templates with defaults in all the fields Meta-Query tab, say for a specific extract done regularly. Or you may use the one included in the samples which has none of the Meta-Query fields filled in.

### The Meta-Query tab



The screenshot shows the 'Software Tool House - Meta-Update Job Template' interface. The 'Meta-Query' tab is active, displaying the following configuration fields:

- Query Type:** Form
- SQL Type:** select
- Output Type:** CSV - comma
- Output File Name:** incidents.csv
- Form Name:** HPD.Help Desk
- Qualification:** `"('3' >= '01/12/2019'" and '3' < '01/01/2020') or ('6' >= '01/12/2019'" and '6' < '01/01/2020')`
- Field List Type:** None
- Field List:** (Empty)

On the left side, there is a sidebar with 'Jobs' and 'Queue Procs' sections. The 'Jobs' section shows: Queued 0, Running 0, Finished 1. The 'Queue Procs' section shows: Running 1, Last heart beat 30 secs ago. Below this is a navigation menu with 'Functions', 'Jobs', 'Templates', and 'Config'.

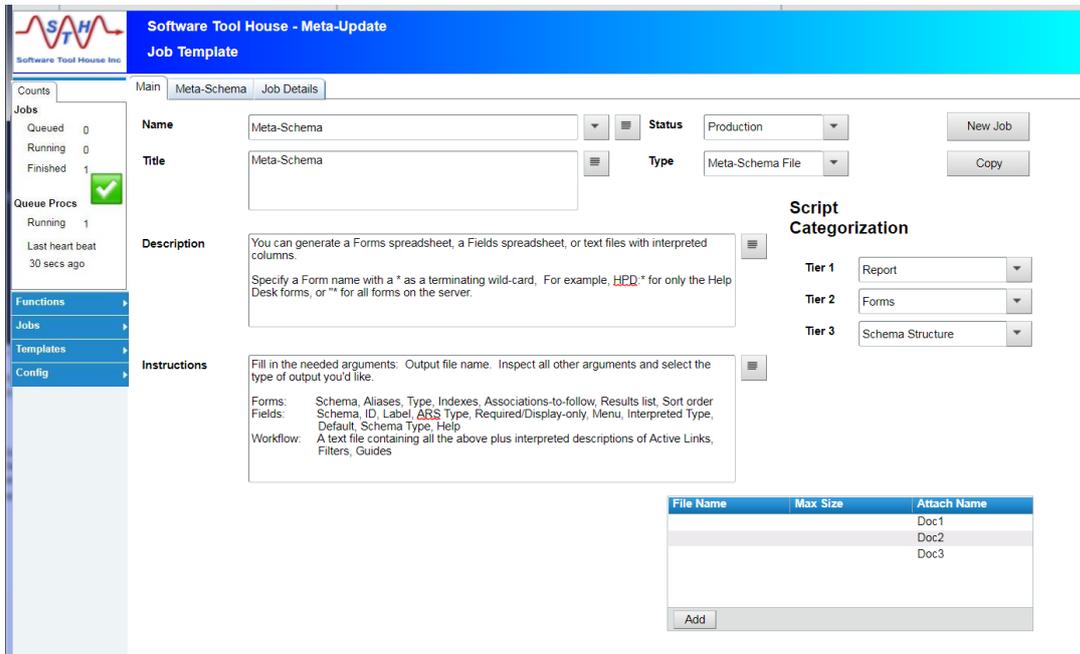
On the right side, there is explanatory text: 'Meta-Query is used to quickly create csv files of your data. It can also be used to do direct SQL queries. If no qualification is entered, the whole table is extracted. A qualification is required for SQL. Examples: "Incident Number" = 'HPD...1345'" select count(\*) from arschema'

**Figure 24 A Meta-Query template with all fields filled in**

## Meta-Schema File

Meta-Schema is used to generate a CSV of all fields and tables in a Remedy server. It can also produce a text file describing workflow.

All fields are required on the Main panel except for Categorization (strongly suggested) and attachments.



**Software Tool House - Meta-Update**  
Job Template

Counts: Queued 0, Running 0, Finished 1

Queue Procs: Running 1, Last heart beat 30 secs ago

Functions, Jobs, Templates, Config

Main | Meta-Schema | Job Details

Name: Meta-Schema | Status: Production | New Job

Title: Meta-Schema | Type: Meta-Schema File | Copy

Description: You can generate a Forms spreadsheet, a Fields spreadsheet, or text files with interpreted columns. Specify a Form name with a \* as a terminating wild-card. For example, **HPD.\*** for only the Help Desk forms, or **\*\*** for all forms on the server.

Instructions: Fill in the needed arguments: Output file name. Inspect all other arguments and select the type of output you'd like.  
Forms: Schema, Aliases, Type, Indexes, Associations-to-follow, Results list, Sort order  
Fields: Schema, ID, Label, ARS Type, Required/Display-only, Menu, Interpreted Type, Default, Schema Type, Help  
Workflow: A text file containing all the above plus interpreted descriptions of Active Links, Filters, Guides

Script Categorization

Tier 1: Report  
Tier 2: Forms  
Tier 3: Schema Structure

File Name	Max Size	Attach Name
		Doc1
		Doc2
		Doc3

Add

**Figure 25 The Sample Meta-Schema Template**

There are three Meta-Schema **Types** available:

- Form** Produces a CSV with these columns  
Schema, Aliases, Type, Indexes, Associations-to-follow, Results list, Sort order.
- Fields** Produces a CSV with these columns  
Schema, ID, Name, Label, ARS Type, Required / Display Only, Menu, Type, Default, Schema Type, Help
- Workflow** Produces a single text file with all the information in both Form and Fields and additionally, all Filters, Active Links, Guides.

# Form Type

	A	B	C	D	E	F	G	H	I	J	K	L
	Schema	Type	Default VUI	Alias	Alias Plural	Alias Short	Alias Short Plural	Audit	Help	Num Indexes	Indexes	Join Map
354	ASE-Admin-ServerSettings	Dialog	Default Administrator View						This dialog box	0		
	ASE-AssignAssoc_AssignForm	Join	Default Admin View							0		Type: Inner, Referential integrity: enforced, Delete option: None; Qual: (\$instanceID25 = 'Form Instance ID');
355	ASE-AssignAssoc_AssignRules	Join	Default Admin View							0		Type: Inner, Referential integrity:
356	ASE-Assignment Association	Regular	Default Admin View							2	1: non-Unique, Name: 1265_500000200_1, 2 Fields, 500000200 - Process InstanceId, 500000300 - InstanceId2; 2: Unique, Name: 1265_179_1, 1 Fields, 179 - Association Instance ID	
357	ASE-Assignment Process	Regular	Default Admin View	Assignment Processes						1	1: Unique, Name: 1264_179_1, 1	
358	ASE-Assignment Rules	Regular	Default Admin View	Assignment Rules						1	1: Unique, Name: 1263_179_1, 1	
360	ASE-AssignmentDetail	Join	Default Admin View	Assignment Details						0		Type: Inner, Referential integrity:
412	AST-Attributes	Regular	Default Administrator View					Log Disabled,		2	1: Unique, Name: 1966_179_1, 1	
413	AST-AttributesAuditFilters	Regular	Default Admin View							1	1: Unique, Name: 11530_179_1, 1	
414	AST-AUD_AssetAssociations	Regular	Administrator							3	1: non-Unique, Name:	
415	AST-Audit Filters	Regular	Default Admin View							1	1: Unique, Name: 11846_179_1, 1	
416	AST-AuditLogAttributes	Regular	Default Admin View					Log Shadow,		0		
417	AST-AuditLogSystem	Regular	Default Admin View					Log Shadow,		1	1: non-Unique, Name: 11979_450_1, 1	
418	AST-BaseElement	Join	Managem Base Elemei Base Elemei Base Element							0		Type: Outer, Referential integrity:
419	AST-BIOSElement	Join	Managem BIOS Elemei BIOS Elemei BIOS Elements							0		Type: Outer, Referential integrity:
420	AST-BMCAssetBaseJoin	Join	Administrator							0		Type: Inner, Referential integrity:
421	AST-BMCAssetBaseJoinComponent	Join	Administrator							0		Type: Inner, Referential integrity:
422	AST-BOAnalysisSrch_dlg	Dialog	Default Admin View	CI Available, CI Available, CI Available, CI Available/Unavailable Times						0		
423	AST-BulkInventory	Join	Managem Bulk Invent; Bulk Invent; Bulk Invent; Bulk Inventory							0		Type: Outer, Referential integrity:
424	AST-BulkUpdate-Display	Dialog	Managem Bulk Update Bulk Update Bulk Update Bulk Update							0		
425	AST-BulkUpdateRelocate-Display	Dialog	Managem Relocate CI Relocate CI Relocate CI Relocate CIs							0		
426	AST-BusinessProcess	Join	Managem Business Prc Business Prc Business Prc Business Processes							0		Type: Outer, Referential integrity:
427	AST-BusinessService	Join	Managem Business Ser Business Ser Business Ser Business Service							0		Type: Outer, Referential integrity:
428	AST-Card	Join	Managem Card	Card	Card	Card	Card			0		Type: Outer, Referential integrity:

Figure 26 The Schemas Sheet from our ITSM 9.1.02 Freebies

## Fields Type

	A	B	C	D	E	F	G	H	I	J	K
	Schema	ID	Name	Label	ARS Type	Required / Display Only	Menu	Type	Default	Schema Type	Help
16	AAS:Activity	60989	Assignee Groups_parent	Assignee Groups_parent	4			Char 255	\$NULLS	Regular	
17	AAS:Activity	1712800	z1G_Company	z1G_Company	4	D		Char 254	\$NULLS	Regular	
18	AAS:Activity	1713900	z1G_TaskViewPreference	z1G_TaskViewPreference	6	D		Enum: 10: Full, 20: Minimal	\$NULLS	Regular	
19	AAS:Activity	1714000	z1G_DefaultCompany	z1G_DefaultCompany	4	D		Char 0	\$NULLS	Regular	
20	AAS:Activity	1716500	z1G_UnrestrictedAccessMember	z1G_UnrestrictedAccessMember	6	D		Enum: 0: false_0, 1: true_1	\$NULLS	Regular	
21	AAS:Activity	3000600	z1G_ChangeIDPrefix	z1G_ChangeIDPrefix	4	D		Char 0	\$NULLS	Regular	
22	AAS:Activity	10000000	RootRequestInstanceID	RootRequestInstanceID	4			Char 38	\$NULLS	Regular	
23	AAS:Activity	10000001	RootRequestName(TMS)	RootRequestName(TMS)	4			Char 30	\$SCHEMAS	Regular	Enter the name of the parent that you want
24	AAS:Activity	10000005	RootRequestFormName	RootRequestFormName	4			Char 254	\$NULLS	Regular	
25	AAS:Activity	10000006	RootRequestID	Request ID	4			Char 38	\$NULLS	Regular	
26	AAS:Activity	10000523	z2TH_TaskWorkInfo	Task Work Info	33	D		Table: (MaxR: 0, Cols: 14) Svr: @ Sch: TMS:WorkInfo Qual: (TaskOrTaskGroupInstanceID = \$z1D_SelectedTaskIds)	\$NULLS	Regular	
27	AAS:Activity	10000545	z2TF_WorkInfoInstanceID	InstanceID	34	D		Column (Table 10000523-z2TH_TaskWorkInfo, on TMS:WorkInfo) Source: Data, Field: 179	\$NULLS	Regular	
28	AAS:Activity	10000546	z2TF_TaskWorkInfoType	Type	34	D		Column (Table 10000523-z2TH_TaskWorkInfo, on TMS:WorkInfo) Source: Data, Field: 10001950	\$NULLS	Regular	
	AAS:Activity	10000547	z2TF_TaskWorkInfoSummary	Summary	34	D		Column (Table 10000523-z2TH_TaskWorkInfo, on TMS:WorkInfo) Source:	\$NULLS	Regular	

Figure 27 The Fields Sheet from our ITSM 9.1.02 Freebies

## Workflow Type

The workflow type produces a text file with all the form and field information of the other two types and interpreted listings of Active Links, Filters, and Guides.

These three images show, in order, the form information, a few selected fields and filters, and, a few active link guides.

```

schema: AAS:Activity
-----
Assoc:      None,
Labels:     Activity, Activities, Activity, Activities
Help:
Q Results:  1000000182 - Activity ID 15 '';
            1000000082 - Company 25 '';
            7 - Status 21 '';
            1000000000 - Summary 50 '';
            1000000164 - Priority 10 '';

Sort Order: 1000000182 - Activity ID -descending;

Indexes:    6;
Index      1: non-Unique, Name: I2125_0_1000000001, 1 Fields,
           Field 1: 1000000001 - Location Company;
Index      2: non-Unique, Name: I2125_0_7, 1 Fields,
           Field 1: 7 - Status;
Index      3: Unique, Name: I2125_0_179, 1 Fields,
           Field 1: 179 - InstanceID;
Index      4: non-Unique, Name: I2125_0_10000000348, 1 Fields,
           Field 1: 10000000348 - Actual Start Date;
Index      5: non-Unique, Name: I2125_0_10000000350, 1 Fields,
           Field 1: 10000000350 - Scheduled Start Date;
Index      6: non-Unique, Name: I2125_0_10000000182, 1 Fields,
           Field 1: 10000000182 - Activity ID;

FieldId  Name(Label)  Req'd/Disply  Menu  Default
-----
1 4 Request ID 0 Char 15 $NULL$
2 4 Submitter(Submitter*) R 0 Char 254 $USERS

```

**Figure 28 The Form information for AAS:Activity**

```

1 4 Request ID 0 Char 15 $NULL$
2 4 Submitter(Submitter*) R 0 Char 254 $USERS$
3 7 Submit Date 0 Time $NULL$
Enum: --> 10: Draft, 20: Assigned, 30: Pending, 40: In Progress, 50: Completed, 70: Cancelled, 80: Closed
1000000018 4 Last Name() R 0 Char 30 $NULL$
1000000063 4 Categorization Tier 1 1 Char 60 $NULL$
Menu: --> Char Menu: CFG:CTL:S1-PBM-Q; Query: on Server: @ CFG:Service Catalog LookUp; Value: 1000000063; I
--> Change, Failure, Request
schema-filters: AAS:Activity has 108 filters

SHR:SHR:Social_SkipSmartitInstalled Enabled 0000 Set, Create, Service; dev_pdho Remedy A 2020/04/08 17:08
Qual: ('zID_SkipIfSmartITInstalled' = $NULL$)
Actn: 1 of 1: Set Fields: Num fields: 1 Server: @; Schema: SHARE:Application Properties
Qual: ('Application GUID' = "SMPIT4073beb410981200cf08b7494b9")
Fld 1 of 1: 304415281 zID_SkipIfSmartITInstalled = $400081600-Application GUID$

AAS:ATV:Status_Check 101 Enabled 0101 Set; Remedy Action R 2020/02/02 19:08
Qual: (('TR.Status' != $NULL$) AND ('TR.Status' != 'DB.Status'))
Actn: 1 of 2: Set Fields: Num fields: 1 Server: @; Schema: AAS:Activity
Qual: ('Request ID' = $Request ID$)
Fld 1 of 1: 301608900 zID Current Database Status = $7-Status$
Actn: 2 of 2: GdeCall: AAS:ATV:Status_Check

AAS:ATV:Status_Check 102 Enabled 0102 Remedy Action R 2020/02/02 19:08
Qual: ('TR.Status' != 'TR.zID Current Database Status')
Actn: 1 of 1: Set Fields: Num fields: 1 Server: @; Schema: SYS:Status Transition Rules
Qual: (((($zID_Current_Database_Status$ = 'From Status Int') AND ($Status$ =

```

**Figure 29 Selected Fields and Filters from AAS:Activity**

```

schema-FilterGuides: AAS:Activity has 14 filter guides

```

Filter Guide	Label	Filters
AAS:ATV:Status_Check	AAS:ATV:Status_Check	AAS:ATV:Status_Check_102 AAS:ATV:Status_Check_103
AAS:ATV:CancelTasks	AAS:ATV:CancelTasks	
AAS:ATV:AssignmentEngine_GUIDE		AAS:ATV:Assignment_SetAssignmentProcess_1 AAS:ATV:Assignment_SetAssignmentProcess_Default AAS:ATV:Assignment_UpdatePrevAssignee_DecrementNumTix`! AAS:ATV:Assignment_RunAssignmentEngine
AAS:ATV:PostAssignmentEngine_GUIDE		AAS:ATV:Assignment_UpdatePrevAssignee_DecrementNumTix`! AAS:ATV:Assignment_PostAssignmentSetAssignee AAS:ATV:CFGRules_GetAASNotificationRules_Assignee
AAS:ATV:ManuallAssignment_GUIDE		AAS:ATV:Assignment_UpdateAssignee_IncrementNumTix`! AAS:ATV:Assignment_UpdatePrevAssignee_DecrementNumTix`! AAS:ATV:Assignment_UpdateSetPrevAssignee AAS:ATV:Assignment_UpdateSetStatusAssigned AAS:ATV:CFGRules_GetAASNotificationRules_Assignee

**Figure 30 Selected Filter Guides from AAS:Activity**

## Meta-Update Script

With Meta-Update, you can automate any mass data change or migration. The Meta-Update Job Console is a Meta-Update script and runs Jobs simultaneously in as many slots as you like.

The Meta-Update distribution contains many useful samples. These are included in the Job Templates and are ready to be run as needed.

Meta-Update scripts are developed on a server or workstation – separate to the Meta-Update Job Console application.

Please see the Meta-Update User's Guide for script development. Remember that all samples can also be reviewed, copied, and edited at will.

All Meta-Update script used in the Job Console reside on the server or workstation that runs the Queue process..

All fields are required on the Main panel except for Categorization (strongly suggested) and attachments.

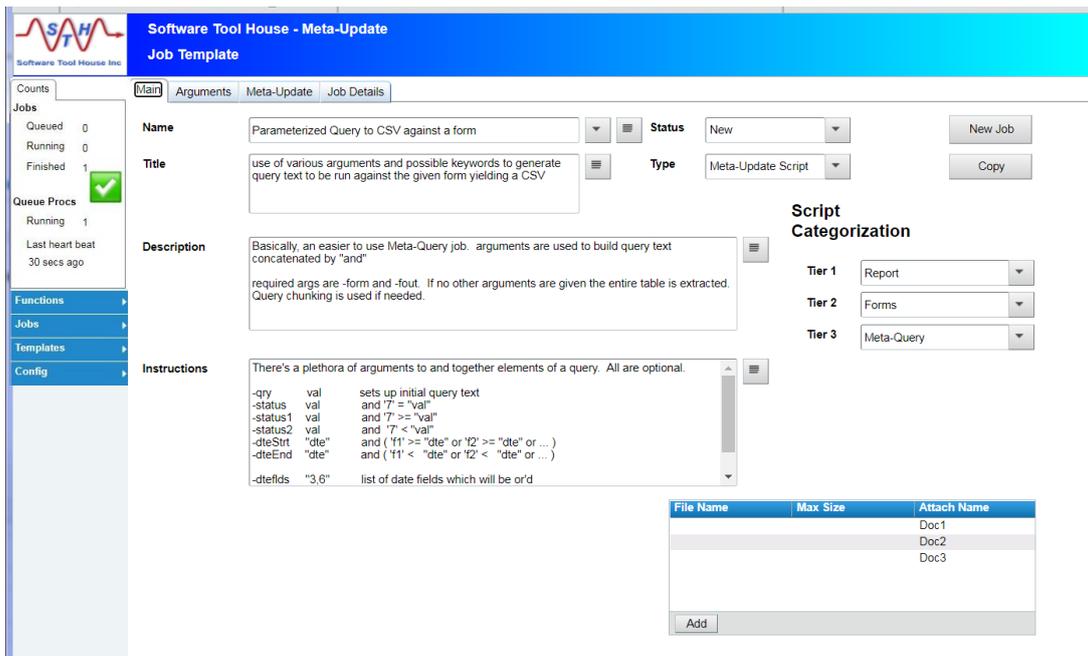
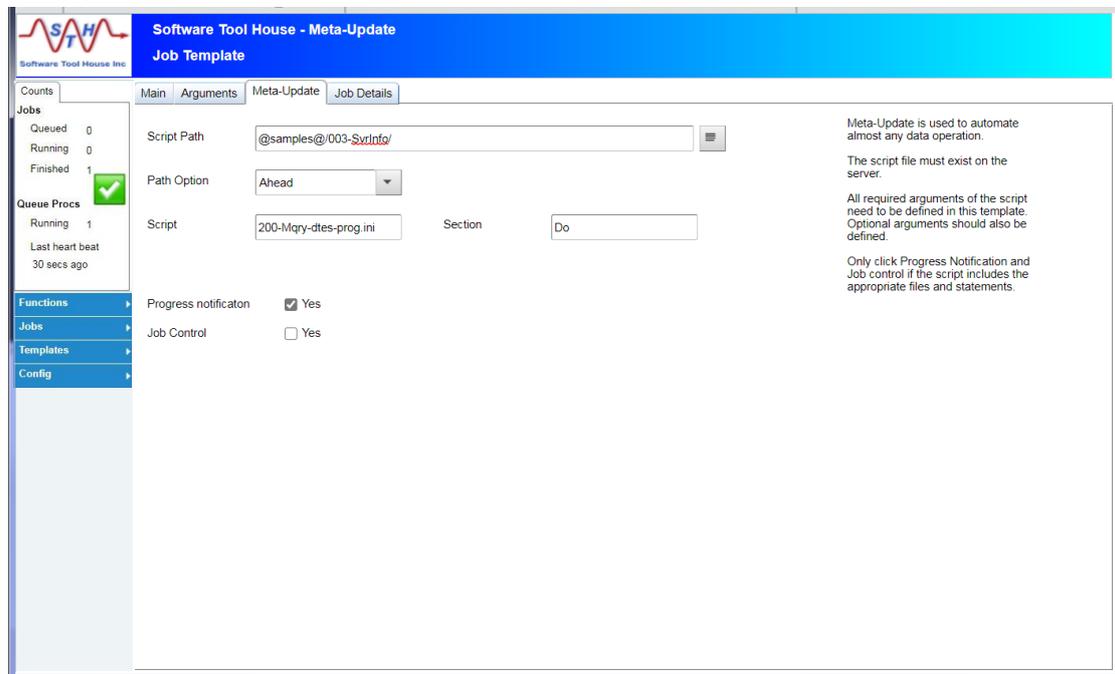


Figure 31 Main Panel for sample script, Parameterized Meta-Query

## Script Name and Location

For Meta-Update scripts, the script, script location, and any needed script path information is required in the Meta-Update tab. An indicator is set if the script include “Job Progress” or “Job Control”.



**Figure 32 Meta-Update panel for Parameterized Meta-Query**

Fields identifying the script:

<b>Script Path</b>	Can be set to the path of the script and any other paths required by the script.
<b>Path Option</b>	Ahead of the standard paths Behind the standard paths
<b>Script</b>	The script file name
<b>Section</b>	The first section to run in the script
<b>Progress notification</b>	The script includes the file 100-JobProgress-Upd.ini
<b>Job Control</b>	The script includes the file 000-Jct1-Sync.ini



## Script Path

Path separators can be coded as either a semi-colon or a colon corresponding to Windows and Linux.

Directory separators can also be coded as either a back-slash or a slash.

Specific keywords can be used in the script path.

@root@	The Meta-Update root path – the install directory.
@scripts@	The Meta-Update root/scripts/ path.
@include@	The Meta-Update root/include/ path.
@conf@	The Meta-Update root/conf/ path.
@samples@	The Meta-Update root/samples/ path.
@my-root@	a customer “root” path
@my-scripts@	a customer scripts path
@my-conf@	a customer configuration path

These keywords are replaced with values from the Job Configuration.

The Queue Process will convert the script path according to its OS and its configuration. Please see Queue Process ahead for more information on adjusting keywords, and path conversions.

## Progress Notification

When the script includes the file, `100-JobProgress-Upd.ini`, it will report its progress on the Job form. As the job progresses, a refresh of the Job detail will have a progress bar that grows until completion.

Templates for these scripts should have the **Progress notification** checked.

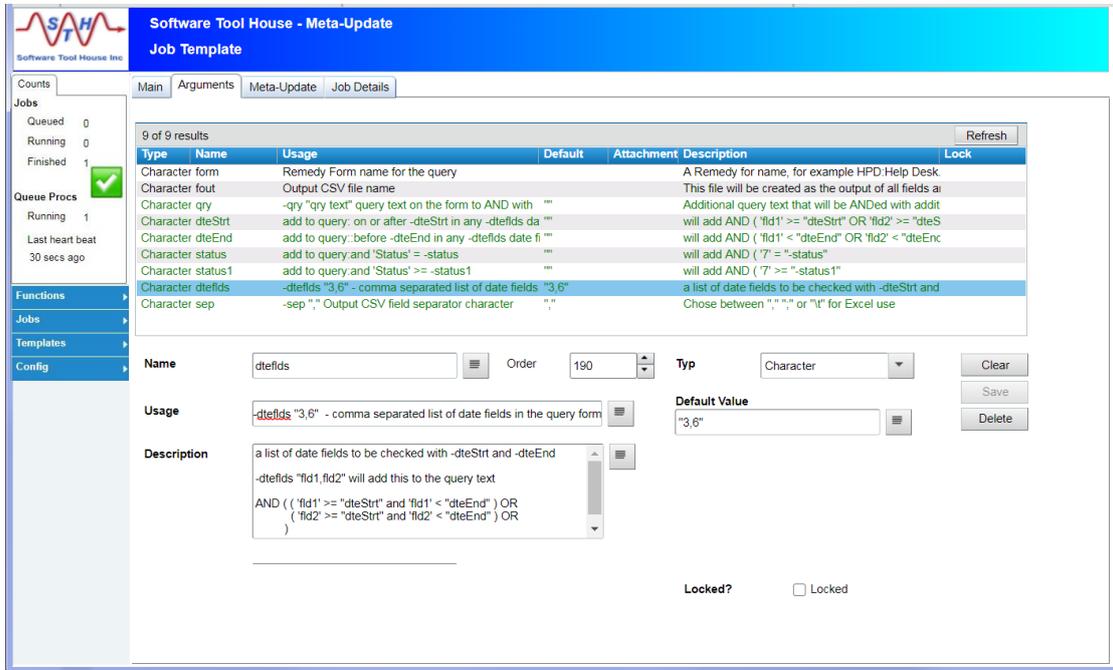
Please note that not all scripts are suitable for progress notifications. For example, the total number of records in a file is not known until the last record is read.

Please see **XXX** for using the JobProgress include file in scripts.

## Script Arguments

Scripts generally but not necessarily have arguments. All sample scripts do have arguments.

Sticking to the Parameterized Meta-Query example, you will see many arguments with only two required and all others having default values.



The screenshot shows the 'Arguments' tab of the 'Meta-Update' job template. It displays a table with 9 arguments and a detailed view for the 'dteflds' argument.

Type	Name	Usage	Default	Attachment	Description	Lock
Character	form	Remedy Form name for the query			A Remedy for name, for example HPD:Help Desk.	
Character	fout	Output CSV file name			This file will be created as the output of all fields a	
Character	qry	-qry "qry text" query text on the form to AND with	""		Additional query text that will be ANDed with addit	
Character	dteStrt	add to query: on or after -dteStrt in any -dteflds da	""		will add AND ("fld1" >= "dteStrt" OR "fld2" >= "dteS	
Character	dteEnd	add to query: before -dteEnd in any -dteflds date fi	""		will add AND ("fld1" < "dteEnd" OR "fld2" < "dteEnc	
Character	status	add to query:and 'Status' = -status	""		will add AND ("?" = "-status"	
Character	status1	add to query:and 'Status' >= -status1	""		will add AND ("?" >= "-status1"	
Character	dteflds	-dteflds "3,6" - comma separated list of date fields	"3,6"		a list of date fields to be checked with -dteStrt and	
Character	sep	-sep "" Output CSV field separator character	""		Chose between "" "" or ""\t" for Excel use	

The detailed view for the 'dteflds' argument shows:

- Name:** dteflds
- Order:** 190
- Type:** Character
- Usage:** -dteflds "3,6" - comma separated list of date fields in the query form
- Default Value:** "3,6"
- Description:** a list of date fields to be checked with -dteStrt and -dteEnd. -dteflds "fld1,fld2" will add this to the query text. AND ( ("fld1" >= "dteStrt" and "fld1" < "dteEnd" ) OR ( "fld2" >= "dteStrt" and "fld2" < "dteEnd" ) OR )
- Locked?:**  Locked

Figure 33 Arguments Panel for sample script, Parameterized Meta-Query

Fields identifying an argument:

<b>Name</b>	This is the <b>Arg=</b> name in the script's <b>[Main]</b> section eg: <b>dteflds</b>
<b>Type</b>	Chose between <b>Logical</b> , <b>Character</b> , <b>Integer</b> , <b>Real</b> , <b>File</b> .
<b>Usage</b>	This is a short description of the usage. eg: <b>-dteflds "comma separated list of date field names or ids"</b>
<b>Description</b>	This is a longer description of the usage. eg: <b>-dteflds "comma separated list of date field names or ids"</b>
<b>Default Value</b>	This is a default value (or null if required) for all types except <b>File</b> . eg: <b>-dteflds "3,6"</b>
<b>Locked?</b>	When checked, the user cannot change the default value.
<b>Order</b>	An integer used to simply order the table of arguments.

## Notes on File arguments

When an argument is a **File**, an **Attachment** field becomes visible. You can use the normal Remedy facilities to attach a file or delete a file.

This attached file name with no path information is supplied with the argument name to the Meta-Update job. It is also set in the read-only Attached File Name field. This is simply to allow the default Attachment column in the displayed table of arguments..

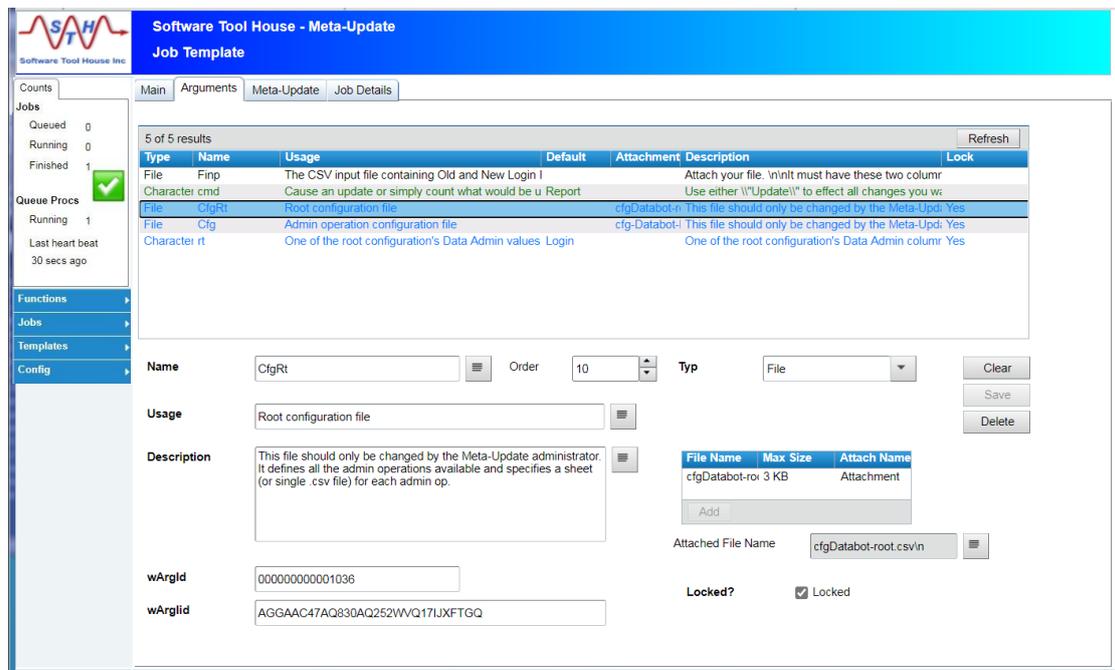
The file will be extracted from the attachment for the Meta-Update job.

If there is no default, this becomes a required argument for the Job.

If you want to use a file on the Queue server, select argument type as **Character** and:

- set the default value to the file name on the Queue server
- set the argument to **Locked**
- ensure the Script Path – in the Meta-Update Tab – will allow the file to be found.

Here's an example of a File argument from a Meta-Databot Login Id sample:



The screenshot shows the 'Software Tool House - Meta-Update Job Template' interface. The 'Arguments' tab is active, displaying a table of 5 arguments. The 'CfgRt' argument is selected, and its configuration is shown in the form below.

Type	Name	Usage	Default	Attachment	Description	Lock
File	Filep	The CSV input file containing Old and New Login I			Attach your file. w\it must have these two columr	
Character	cmd	Cause an update or simply count what would be u Report			Use either \\\"Update!\" to effect all changes you wr	
File	CfgRt	Root configuration file		cfgDatabot-ro	This file should only be changed by the Meta-Upd. Yes	
File	Cfg	Admin operation configuration file		cfg-Databot!	This file should only be changed by the Meta-Upd. Yes	
Character	rt	One of the root configuration's Data Admin values Login			One of the root configuration's Data Admin columr Yes	

The configuration form for the 'CfgRt' argument includes the following fields:

- Name:** CfgRt
- Order:** 10
- Typ:** File
- Usage:** Root configuration file
- Description:** This file should only be changed by the Meta-Update administrator. It defines all the admin operations available and specifies a sheet (or single .csv file) for each admin op.
- Attached File Name:** cfgDatabot-root.csv\n
- Locked?:**  Locked
- wArgId:** 000000000001036
- wArgItId:** AGGAAC47AQ830AQ252WVQ17LJXFTGQ

Figure 34 -CfgRt File argument from Meta-Databot - Login ID Template sample

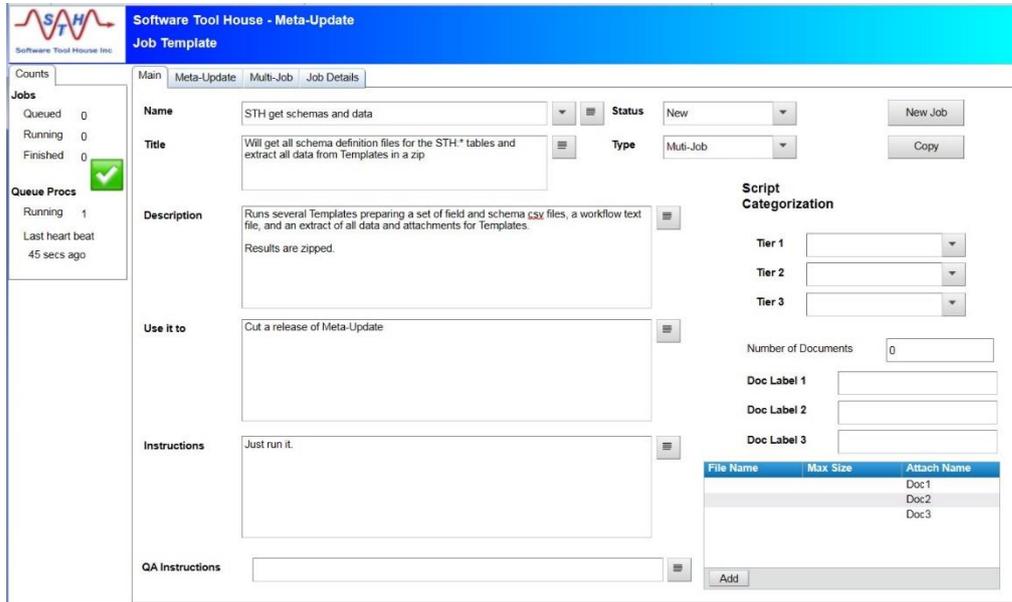
## Notes on Script Arguments defaults

A Meta-Update template effectively overrides any argument defaults of the script.

This is because all arguments defined in the script are defined in the Template and these arguments are all specified on the Meta-Update job. As there are no missing arguments, the script defaults have no meaning.

# Multi-Job

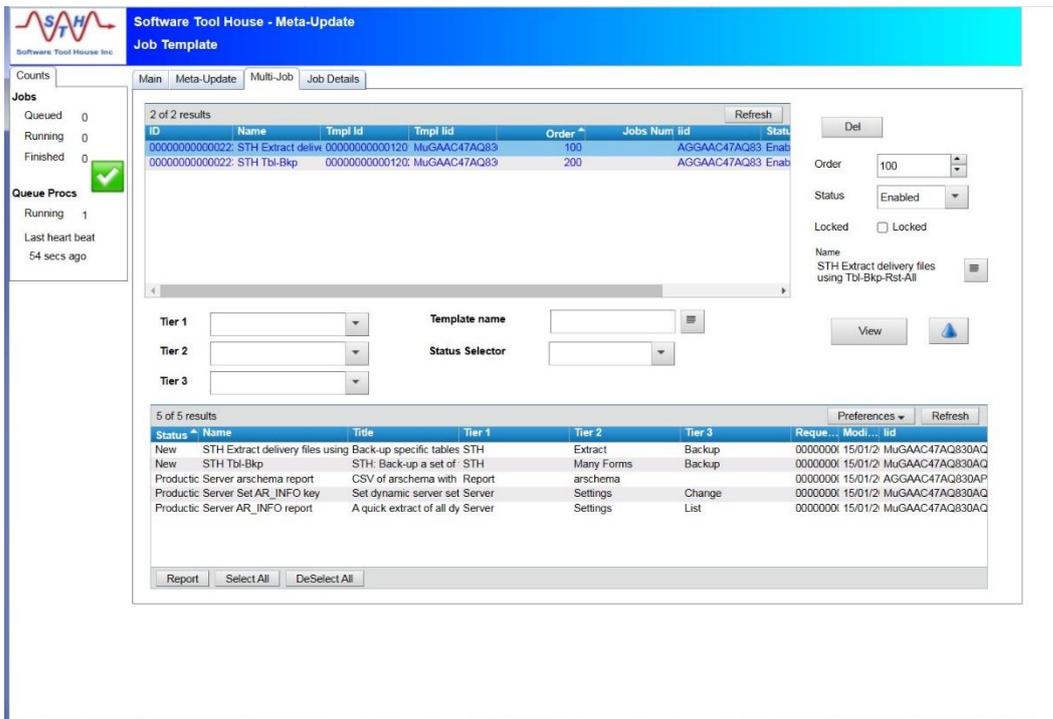
A Multi-Job allows you to select several Templates and run them as a single Job.



The screenshot shows the 'Multi-Job' configuration interface. On the left, there are 'Counts' for Queued (0), Running (0), and Finished (0) jobs, and 'Queue Procs' for Running (1) processes. The main area is titled 'Job Template' and contains several fields:
 

- Name:** STH get schemas and data
- Title:** Will get all schema definition files for the STH.\* tables and extract all data from Templates in a zip
- Description:** Runs several Templates preparing a set of field and schema csv files, a workflow text file, and an extract of all data and attachments for Templates. Results are zipped.
- Use it to:** Cut a release of Meta-Update
- Instructions:** Just run it.
- QA Instructions:** (empty)
- Status:** New
- Type:** Multi-Job
- Script Categorization:** Tiers 1, 2, and 3 are set to empty.
- Number of Documents:** 0
- Doc Label 1, 2, 3:** (empty)
- Attachments:** A table with columns 'File Name', 'Max Size', and 'Attach Name'. It lists Doc1, Doc2, and Doc3.

The Multi-Job tab allows you to bring templates into the group.



This screenshot shows the 'Multi-Job' configuration screen with a list of templates. The 'Jobs' section on the left shows 1 running process. The main area displays a table of 2 results for the job template:
 

ID	Name	Tmpl Id	Tmpl lid	Order	Jobs Num lid	Status
00000000000022	STH Extract deliv...	00000000000120	MuGAAC47AQ83	100	AGGAAC47AQ83	Enab
00000000000022	STH Tbl-Bkp	00000000000120	MuGAAC47AQ83	200	AGGAAC47AQ83	Enab

 Below the table, there are controls for 'Tier 1', 'Tier 2', and 'Tier 3', and a 'Template name' field. A 'Del' button is also present. At the bottom, there is a table with 5 results:
 

Status	Name	Title	Tier 1	Tier 2	Tier 3	Reque...	Modi...	lid
New	STH Extract delivery files using Back-up specific tables	STH	Extract	Extract	Backup	00000000	15/01/2	MuGAAC47AQ830AQ
New	STH Tbl-Bkp	STH Back-up a set of	STH	Many Forms	Backup	00000000	15/01/2	MuGAAC47AQ830AQ
Productic	Server arschema report	CSV of arschema with	Report	arschema	Change	00000000	15/01/2	AGGAAC47AQ830AP
Productic	Server Set_AR_INFO key	Set dynamic server set	Server	Settings	Change	00000000	15/01/2	MuGAAC47AQ830AQ
Productic	Server AR_INFO report	A quick extract of all dy	Server	Settings	List	00000000	15/01/2	MuGAAC47AQ830AQ

Only Templates that are "Complete" are available.

## Details Tab

All Templates and Jobs have a Details tab available.

Here you can select to zip output files. You can also enter command to be run before and after the job runs.



# Index

No index entries found.

