Software Tool House Inc.

Meta-Update

Installation Guide

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. The reader should 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

Updates

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

Comments

Your comments are welcome! Please see: www.softwaretoolhouse.com/support and click **Comments**, or email us at 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 *This document.*	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.
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.

Here's an overview of the Installation Guide sections:

Introduction

This is a short description of Meta-Update and the Job Console.

Licensing

This covers obtaining and using licenses and license files.

Installing

The installation of Meta-Update on LINUX and Windows.

Running

Running Meta-Update and interpreting the output. .

Installing the Job Console Application

This explains how to install or upgrade the Job Console application.

Distribution Contents

This lists the distribution files and their purpose.

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.
bmc	Applies to BMC Remedy ARS server sessions. Cannot be used for, or does not refer to ServiceNow sessions..
now	Applies to ServiceNow sessions. Cannot be used for, or does not refer to BMC Remedy ARS server sesions.
🛑	Caution. Failure to follow recommended actions may cause data loss.
Courier Bold	Courier Bold indicates a command you can enter. For example: ⊞ set SthApiRetry=90 - 92 0 60 93 0 30 🐧 export SthApiRetry=90 - 92 0 60 93 0 30

Table of Contents

Preface 3

- Document Library 4
- Organisation 5
- Document Conventions 6
- Table of Contents 7
- List of Figures 9
- Introduction 12

Licensing 15

- Requesting a Server License 16
- License Files 19
- Specifying The License Key 20

Installing Meta-Update 21

- About Installing 22
- Deciding the Install Locations 23
- Expanding The Distribution 24
  - Distribution Directories 25
- Preparing a ServiceNow instance 26
- Complete Installation 31
- About Job Console Installation 32
- Defining a Fixed User 34
- Run Time Environment 35
- Environment variables 36
  - Script Path Environment variable 36
  - API Retry Environment variable 37
  - License Environment Variable 38
  - Other Environment Variables 38
- User Password Encryption 40

Running Meta-Update 45

- Setting Up the Run-Time Environment 46
- Program Output 48
- Firing from Workflow 50
- Developing Scripts 51

Installing the Meta-Update Job Console 55

Job Queue Server 60

- Where to run Job Queue Server 61
- Evaluating Job Queue Server Performance 62
- Queue Server Config 63
- Starting The Job Queue Server 65

Distribution Contents 66

Directory Structure 67
bin 68
docs 70
include 71
install 72
samples 73
scripts 74

Index 78

List of Figures

Figure 1 Software Tool House Trial License Request page 16

Figure 2 Example of a BMC Remedy Server Platform Information page 17

Figure 3 Example of a ServiceNow Diagnostics page 18

Figure 4 The distribution root directories 24

Figure 5 Select CORS Rules by filtering 26

Figure 6 Defining a CORS Rule 27

Figure 7 Select Tables under System Definition by filtering 28

Figure 8 Filter Table list for `sys_properties` 29

Figure 9 Select `sys_properties` table 29

Figure 10 Enable All Application Access checkboxes 30

Figure 5 Overview of the Job Console Application 32

Figure 6 A Job Template referencing a sample script 33

Figure 7 Sample SthLicUpd Prompt session with two license files 41

Figure 8 Example of issuing the same query on two servers using SthLic.cmd 42

Figure 9 Generated SthLic.cmd Usage 42

Figure 10 Setting The Path, Licensing and Script Path in Windows 47

Figure 11 Overview of The Job Console and Queue Servers 56

Figure 12 Install Script Usage 57

Figure 13 Job Configuration 63

Figure 14 Local Job Config CSV file 63

Introduction

Introduction

Thank you for choosing Meta-Update.

With Meta-Update, creating repeatable imports, migrations and batch operations on your BMC Remedy and ServiceNow data is a snap.

The Meta-Update distribution is repleate with samples. The samples are also available as Meta-Update Job Console Templates.

Meta-Update also comes with a Remedy application which can be optionally installed on a Remedy server. The application allows multiple Remedy administrators to run jobs under their user logins through a Queue server.

This document covers:

- Licensing
  - How to request a license
  - Using License files
  - Specifying the License Key
- Downloading
- Installing
  - Deciding on the installation root
  - Expanding the distribution
  - Distribution Directories
  - Complete Installation
  - About Job Console Installation
  - Run -time Environment
  - Environment Variables
  - Defining a Fixed ARS or ServiceNow User
  - Using SthLic.cmd or SthLic.sh
- Installing the Meta-Update Job Console
- Running Meta-Update
- Distribution Contents

Installing

Licensing

Requesting a Server License

Meta-Update is licensed on a server by server basis.

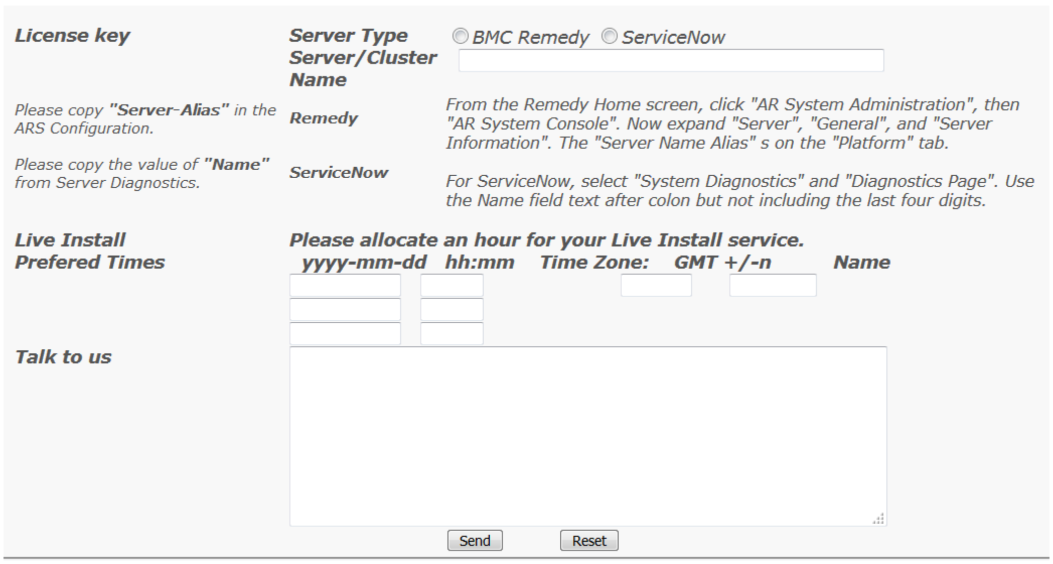
Licenses are dated and may be indefinite or limited term. Evaluation licenses and migration project licenses are examples of limited term licenses.

Once a Meta-Update license is granted for use against a BMC Remedy or ServiceNow Server, Meta-Update scripts can be run against that Server at any time or on any server or workstation and by any users with access and a sign-in to the licensed server.

License keys may be requested from the Software Tool House's web site at http://www.softwaretoolhouse.com.

When requesting license keys, the ARS Server Alias Name must be supplied. This ARS Server Alias Name is matched against the supplied license key.

For ServiceNow instances, the Name is the Cluster Name (see below).



License key	***Server Type***	◯ *BMC Remedy* ◯ *ServiceNow*
	Server/Cluster Name	
*Please copy **"Server-Alias"** in the ARS Configuration.*	***Remedy***	*From the Remedy Home screen, click "AR System Administration", then "AR System Console". Now expand "Server", "General", and "Server Information". The "Server Name Alias" s on the "Platform" tab.*
*Please copy the value of **"Name"** from Server Diagnostics.*	***ServiceNow***	*For ServiceNow, select "System Diagnostics" and "Diagnostics Page". Use the Name field text after colon but not including the last four digits.*
Live Install Prefered Times	***Please allocate an hour for your Live Install service.*** ***yyyy-mm-dd hh:mm Time Zone: GMT +/-n Name***	
Talk to us		

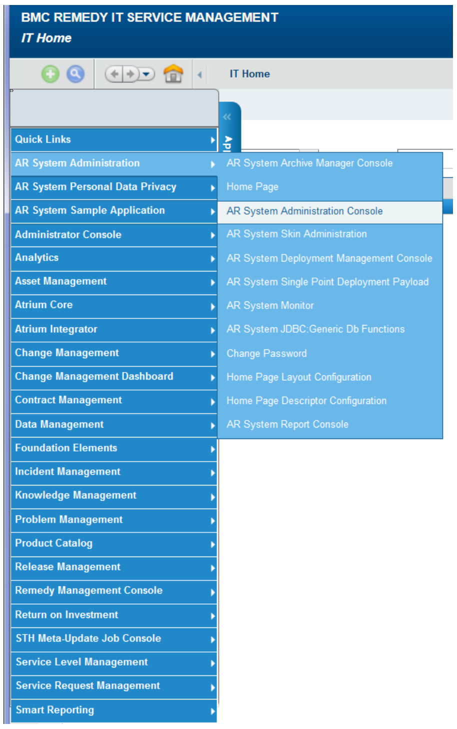
Figure 1 Software Tool House Trial License Request page

This name is specified in the ARS Server's server configuration file, `ar.conf`, or, `ar.cfg`.

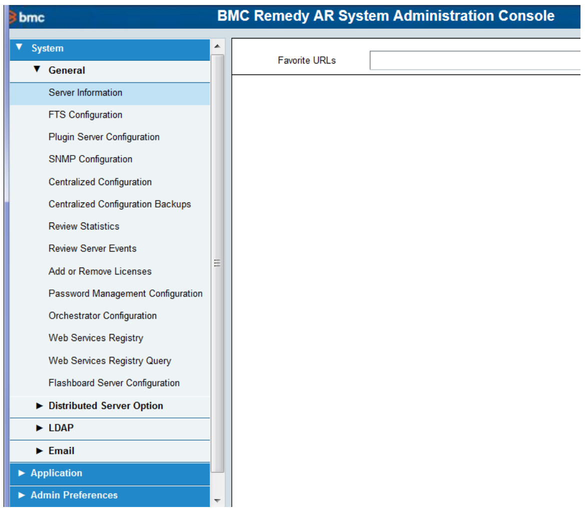
It is given by the `Server-Name` value.

Or, through the interface:

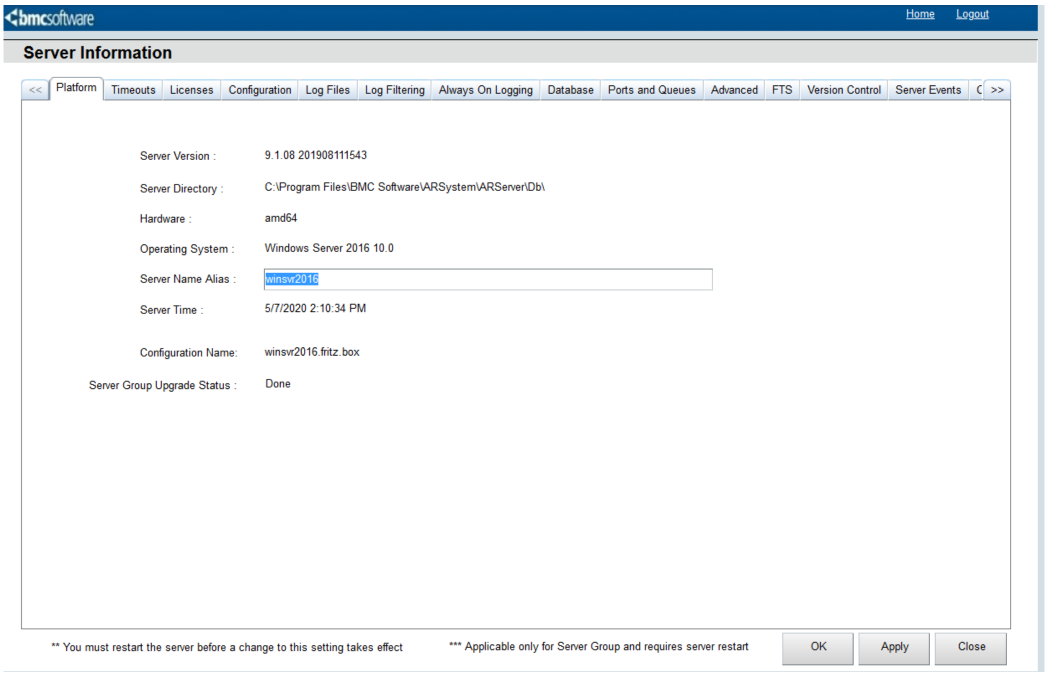
From the standard “Home” menu, expand “AR System Administration” and click “AR System Administration Console”.


BMC REMEDY IT SERVICE MANAGEMENT
IT Home
IT Home
Quick Links
AR System Administration
AR System Personal Data Privacy
AR System Sample Application
Administrator Console
Analytics
Asset Management
Atrium Core
Atrium Integrator
Change Management
Change Management Dashboard
Contract Management
Data Management
Foundation Elements
Incident Management
Knowledge Management
Problem Management
Product Catalog
Release Management
Remedy Management Console
Return on Investment
STH Meta-Update Job Console
Service Level Management
Service Request Management
Smart Reporting
AR System Archive Manager Console
Home Page
AR System Administration Console
AR System Skin Administration
AR System Deployment Management Console
AR System Single Point Deployment Payload
AR System Monitor
AR System JDBC:Generic Db Functions
Change Password
Home Page Layout Configuration
Home Page Descriptor Configuration
AR System Report Console

In the Administration Console, expand “System” and “General” and click “Server Information”.


bmc
BMC Remedy AR System Administration Console
System
General
Server Information
FTS Configuration
Plugin Server Configuration
SNMP Configuration
Centralized Configuration
Centralized Configuration Backups
Review Statistics
Review Server Events
Add or Remove Licenses
Password Management Configuration
Orchestrator Configuration
Web Services Registry
Web Services Registry Query
Flashboard Server Configuration
Distributed Server Option
LDAP
Email
Application
Admin Preferences
Favorite URLs

Once, the “Server Information” form comes up, use the ***Server Name Alias*** from the “Platform” tab.



Server Version :	9.1.08 201908111543
Server Directory :	C:\Program Files\BMC Software\ARSystem\ARServer\Db\
Hardware :	amd64
Operating System :	Windows Server 2016 10.0
Server Name Alias :	winsvr2016
Server Time :	5/7/2020 2:10:34 PM
Configuration Name:	winsvr2016.fritz.box
Server Group Upgrade Status :	Done

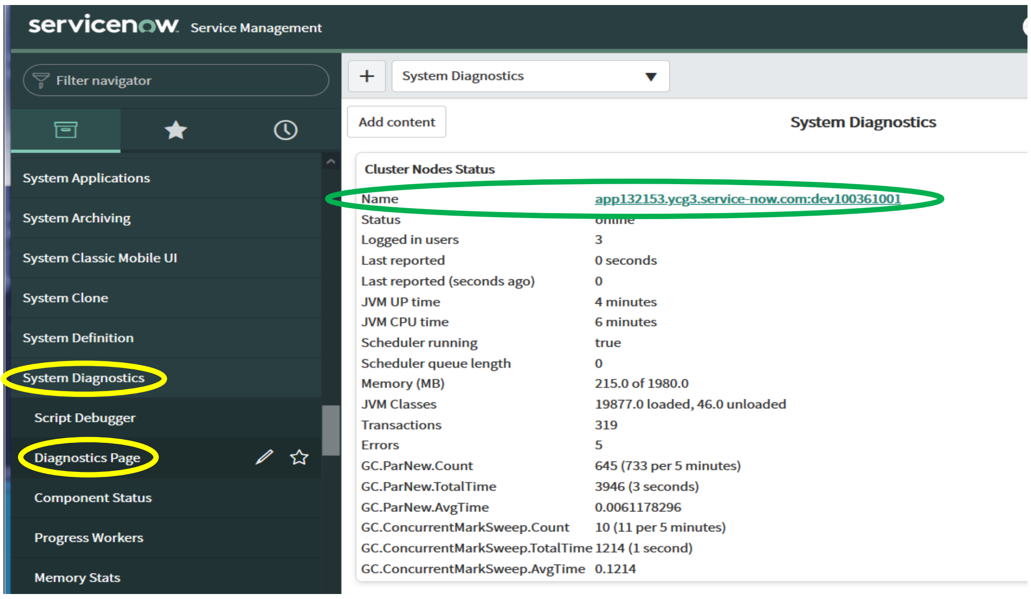
** You must restart the server before a change to this setting takes effect

*** Applicable only for Server Group and requires server restart

Figure 2 Example of a BMC Remedy Server Platform Information page

For ServiceNow instances, Expand the Menu, “System Diagnostics” and click “Diagnostics Page”.

The ***Name*** field text after colon but not including the last four digits is used to generate the license.



System Diagnostics

Cluster Nodes Status

Name	app132153.ycg3.service-now.com:dev100361001
Status	online
Logged in users	3
Last reported	0 seconds
Last reported (seconds ago)	0
JVM UP time	4 minutes
JVM CPU time	6 minutes
Scheduler running	true
Scheduler queue length	0
Memory (MB)	215.0 of 1980.0
JVM Classes	19877.0 loaded, 46.0 unloaded
Transactions	319
Errors	5
GC.ParNew.Count	645 (733 per 5 minutes)
GC.ParNew.TotalTime	3946 (3 seconds)
GC.ParNew.AvgTime	0.0061178296
GC.ConcurrentMarkSweep.Count	10 (11 per 5 minutes)
GC.ConcurrentMarkSweep.TotalTime	1214 (1 second)
GC.ConcurrentMarkSweep.AvgTime	0.1214

Figure 3 Example of a ServiceNow Diagnostics page

In the image above, for example, it would be: “`dev10036`”.

License Files

A license file contains the keys for all Meta-Update binaries for a single server.

It is generally named, ***server*.lic**, where the "**server**" is the name supplied for the server.

It is a simple text file that contains the licenses and expiration dates for that supplied server name.

This file should be placed in the Meta-Update installation's bin directory.

The file may be freely copied to other servers and workstations that will use Meta-Update against the licensed server.

You may have multiple license files in a bin directory. This is especially useful if the server or workstation can connect to many servers.

Examples:

- On a development workstation, you may have a bin/ with 6 license files: delopment, QA, and production servers for both Remedy and ServiceNow
- On a QA workstation, you may have two license files: Remedy QA and ServiceNow QAr

Meta-Update provides a convenient utility to generate a batch file or shell script to easily switch between servers and encrypt sign-on passwords to those servers.

SthLic.cmd or .sh is generated once and will allow you to easily switch between servers. It simply sets default server access environment variables.

SthLicUpd.exe will generate the above file. It will read server licenses and generate a batch file or shell script to set both the target server and the licensing in the environment.

For example:

```
SthLic.cmd    prod
SthMqry.exe  -f  hpd-prod-march.csv
             -S "HPD:Help Desk"  "'6' <   \"05/01/2020\"
                                 and '6' >=  \"04/01/2020\""
SthLic.cmd    dev
SthMqry.exe  -f  hpd-dev-march.csv
             -S "HPD:Help Desk"  "'6' <   \"05/01/2020\"
                                 and '6' >=  \"04/01/2020\""
```

SthLicUpd.exe will also encrypt the server user password so no passwords can be discovered by reading the generated **SthLic.cmd** or **SthLic.sh** file.

See below for information on running **SthLicUpd.exe**.

Specifying the License Key

The license key can be given to Meta-Update in the following ways:

- In the script file's `[Main] License=` keyword.
  This can be a reference to an environment variable.
- On the command line with the **`-lic`** argument
- In the environment variable, `SthMupdLic=`
- Using SthLic.cmd or SthLic.sh

The above list is in priority order. The first license encountered going down the list is used.

Specifying the License Key with Environment Variables:

On Windows, environment variables may be set on a single DOS session, for a specific user's complete Windows sessions, or for all users' Windows system environment.

To set a single DOS box's environment, open a Command Prompt, then, use the set command to assign the License Key to the expected environment variables. For example,

```
set SthMupdLic=QF143G6-PL95SQ
```

Or, on Linux:

```
export SthMupdLic=QF143G6-PL95SQ
```

Specifying the License Key in the Script

The Meta-Update License may be specified in the **`[Main]`** section of a script file as an alternative to using environment variables. To specify the license key in the **`[Main]`** section, code the **`License=`** keyword with the license key as the value.

`License =` This is the Software Tool House supplied License key for a server.

It must be specified exactly as was specified when the license was requested (no case changes).

The following script file addition would accomplish the same thing as the environment variable example above:

```
[Main]
License        =        QF143G6-PL95SQ
```

This practise is not recommended as it makes scripts more difficult to move between environments.:

Installing Meta-Update

About Installing

Meta-Update is distributed as a single zip or gzip file.

As soon as the distribution is unzipped, Meta-Update is ready to use. Further steps are required for using the Meta-Update Job Console application.

This chapter covers the following points:

Deciding the install locations	
Preparing a ServiceNow instance	
Complete Installation	
About Job Console Installation	
Defining a fixed user	
Run Time Environment	
Environment Variables	
Preparing SthLic.cmd encrypted passwords	
Distribution Contents`	

:

Deciding the Install Locations

On Windows one normally installs programs in

```
C:\Program Files\
```

The space in the name necessitates the use of quotes in any individual command line, any batch file, and any references within scripts designed to control processes.

It is generally more convenient when you use an alternative path, even a second drive. For example:

```
D:\apps\sth\
```

On Linux, we also recommend an alternative location be used. For example:

```
/apps/sth/
```

The location of the install location does not matter and is your decision.

Different locations can be installed opn different workstations or servers.

Of course, standardization brings advantages.

Expanding the Distribution

Meta-Update is distributed as a single zip or g-zip file.

The file may be expanded in your applications area. For example,

On Windows, you could unzip the distribution in:

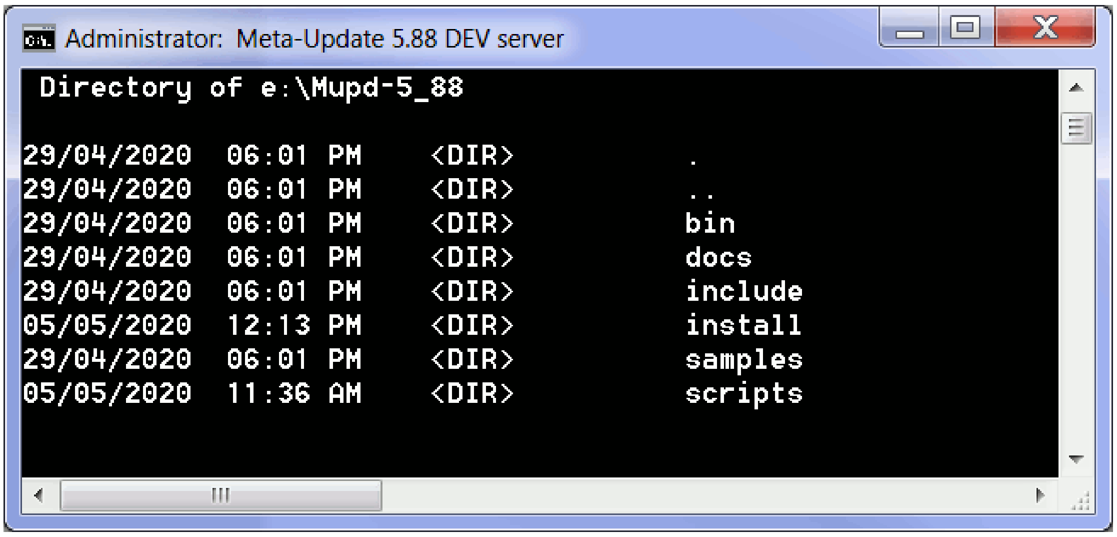
```
C:\Program Files\STH\
D:\apps\STH\
```

On Linux, the distribution is a gzipped tarball.

```
cd /apps/STH/
gunzip   SthMupd-5.92-918-lnx-lx64.tar.gz
tar   xvf   SthMupd-5.92-918-lnx-lx64.tar
```

Once, the distribution is expanded, the root directory is called Mupd_Xxx where X.xx is the Meta-Update release. This allows you to test different versions and APIs of Meta-Update easily.

This root Mupd_Xxx directory will contain these folders or sub-directories.



```
 Directory of e:\Mupd-5_88

29/04/2020  06:01 PM    <DIR>          .
29/04/2020  06:01 PM    <DIR>          ..
29/04/2020  06:01 PM    <DIR>          bin
29/04/2020  06:01 PM    <DIR>          docs
29/04/2020  06:01 PM    <DIR>          include
05/05/2020  12:13 PM    <DIR>          install
29/04/2020  06:01 PM    <DIR>          samples
05/05/2020  11:36 AM    <DIR>          scripts
```

Figure 4   The distribution root directories

Distribution Directories

The distribution is contained in the single directory is called **`Mupd_X_xx`** where **`Mupd_X_xx`** is the Meta-Update release. This allows you to test different versions and APIs of Meta-Update easily.

This directory contains directories and files that is the Meta-Update distribution.

Directory	Contents
`bin`	Contains all required 64-bit binaries (.exe) and libraries (.dll) for Meta-Update, the Job Console, and bundled utilities. It is recommended that this path be added to the system Environment Variables for use by a Server, and in the User's Environment Variables for workstations.
`docs`	This directory contains the Meta-Update and the Job Console user manuals and the Trace Facility Administration Guide. These are distributed as PDF files.
`install`	Contains scripts, data, ARS def files used a first time to install the GUI application and data and thereafter to create a local configuration file for use with an off-server Queue Process.
`include`	These are included script fragments used by the Meta-Update Job Console, samples, and script packages...
`scripts`	These are scripts used by the Meta-Update Job Console, Meta-Archive, and Meta-Databot.
`samples`	Contains sample Meta-Update scripts and a sample Trace configuration file.
`conf`	These are different configurations used by the Meta-Update scripts, the Job Console, Meta-Archive, and Meta-Databot.

Meta-Update supports the 64-bit architecture. All binaries, dlls, shared objects, and executables in the "**`bin`**" only run on 64 bit machines.

```
"C:\Program Files\SoftwareToolHouse\Mupd_6_00\bin\"
/apps/sth/mupd_6_00/bin/
```

This "**`bin`**" directory contains all required binaries (**`.exe`**) and libraries (**`.dll`** / **`.so`**) for Meta-Update and associated utilities.

It is recommended that this path be added to the system Environment Variables for use by a Server, and in the User's Environment Variables for workstations.

All Meta-Update binaries print usage instructions when entered with no arguments.

Preparing a ServiceNow Instance

For current releases of ServiceNow, a CORS Rule is required to allow a client to issue REST API calls from a non-ServiceNow domain.

The default ServiceNow behaviour is to return all records if a query is in error. The change causes ServiceNow to return no records on invalid queries. An example is a query with a field name misspelled.

Step 1 – Define a CORS Rule

From the main menu, select All and filter for "**`CORS`**". Then select **`CORS Rules`**.

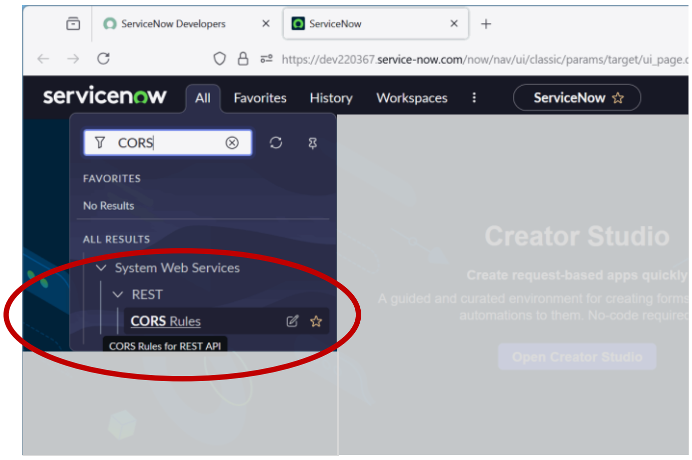


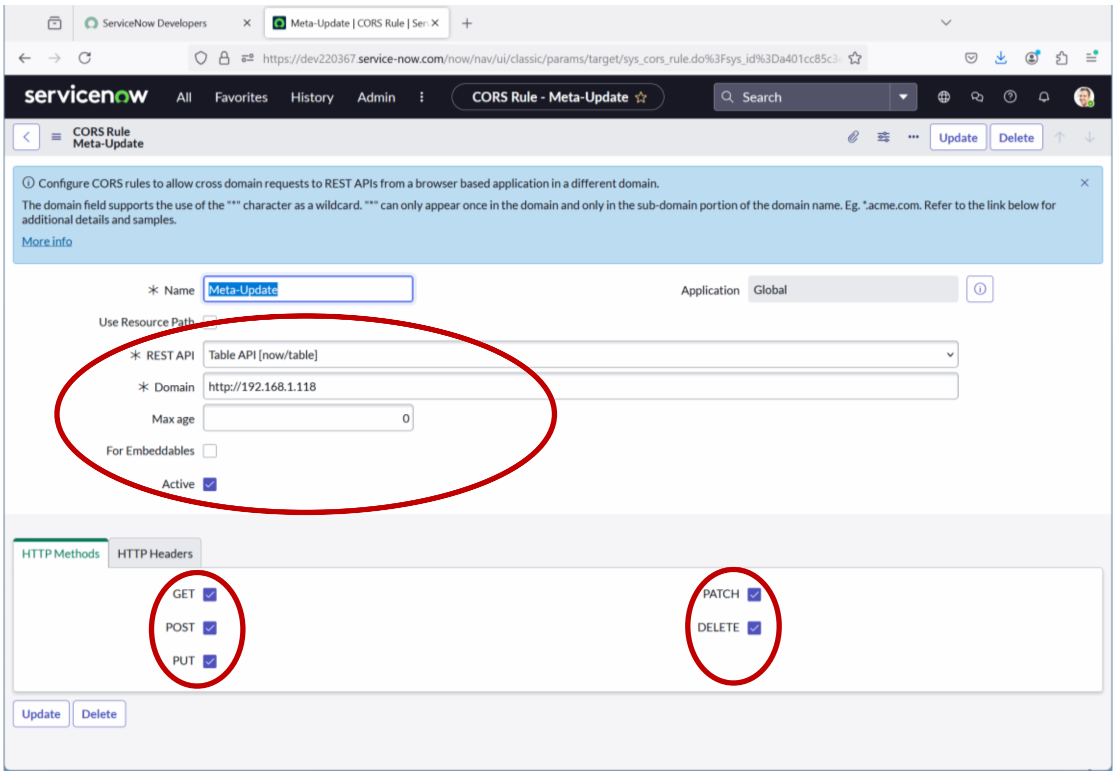
Figure 5 Select CORS Rules by filtering

Under the list, select **`New`** to create a new CORS Rule.

The easiest CORS Rule is to specify the IP address of the client machine that will be running **Meta-Update**. It does not matter if the client machine is not addressable from the ServiceNow instance.

Ensure the `Table API  [now/table]` is selected.

Ensure all methods are checked.



ServiceNow Developers
Meta-Update | CORS Rule | Ser
https://dev220367.service-now.com/now/nav/ui/classic/params/target/sys_cors_rule.do%3Fsys_id%3Da401cc85c3
servicenow
All
Favorites
History
Admin
CORS Rule - Meta-Update
Search
CORS Rule
Meta-Update
Update
Delete

ⓘ Configure CORS rules to allow cross domain requests to REST APIs from a browser based application in a different domain.

The domain field supports the use of the "*" character as a wildcard. "*" can only appear once in the domain and only in the sub-domain portion of the domain name. Eg. *.acme.com. Refer to the link below for additional details and samples.

More info

* Name	Meta-Update	Application	Global
Use Resource Path	☐		
* REST API	Table API [now/table]		
* Domain	http://192.168.1.118		
Max age	0		
For Embeddables	☐		
Active	✓		

HTTP Methods | HTTP Headers

GET	✓	PATCH	✓
POST	✓	DELETE	✓
PUT	✓		

Figure 6   Defining a CORS Rule

Step 2 – Change table `sys_properties` to allow writes and insertions.

By default, the REST API will return all records of a table when a query on that table is invalid.

To change this behaviour, this specific **`sys_properties`** row is required.

```
name           glide.invalid_query.returns_no_rows
sys_name       glide.invalid_query.returns_no_rows
value          TRUE
type           BOOL
```

By default, the **`sys_properties`** table can only be read. As an administrator, you can not insert a record manually as well.

To add the required property, this table ***must allow insertions***.

You can then manually insert this row through the normal ServiceNow list **`New`** button. Alternatively, you can use a simple argument to set it:

```
          -snQryChk     set | quit | ignore
```

```
SthLic         SN
SthMqry       -snQryChk   set   -S sys_properties ^
              sys_nameLIKEglide.invalid_query.returns_no_rows
```

Once the property is set, it is perfectly fine to revert this table's allowed Application access.

To allow insertions and edits on the **`sys_properties`** table:

From the main menu, select All and filter for "**`Tables`**". Then select **`Tables`** under **`System Definition`**

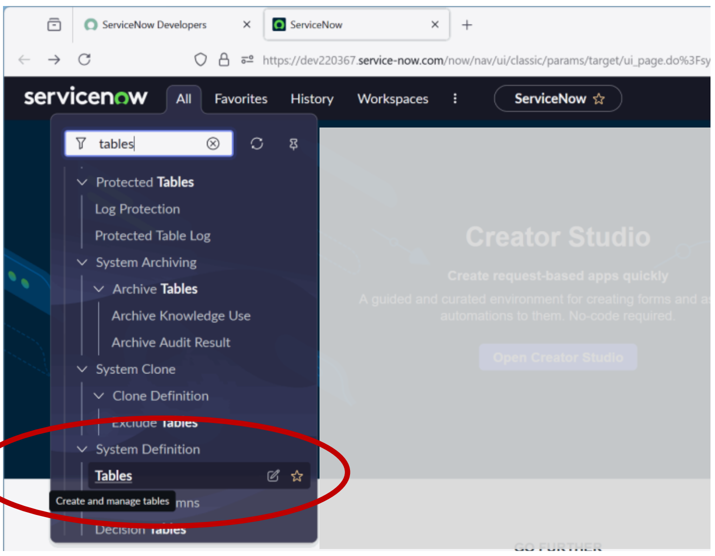
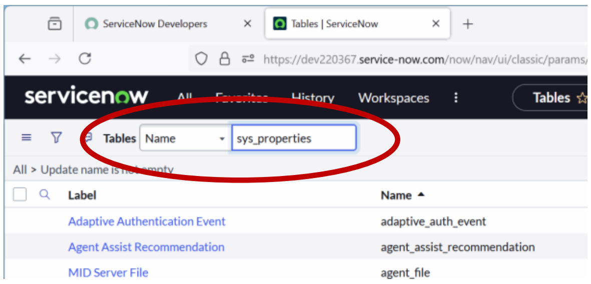


Figure 7   Select Tables under System Definition by filtering

When presented with the list, type **`sys_properties`** into the list search box.

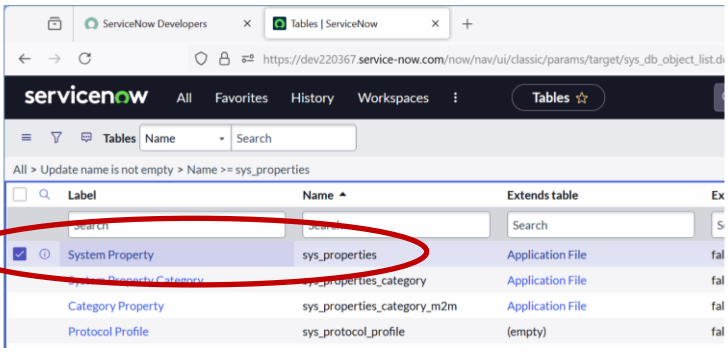


All > Update name is not empty

Label	Name
Adaptive Authentication Event	adaptive_auth_event
Agent Assist Recommendation	agent_assist_recommendation
MID Server File	agent_file

Figure 8 Filter Table list for `sys_properties`

This will scroll the **`sys_properties`** table to the first. Click it and select the **`Application Access`** tab.

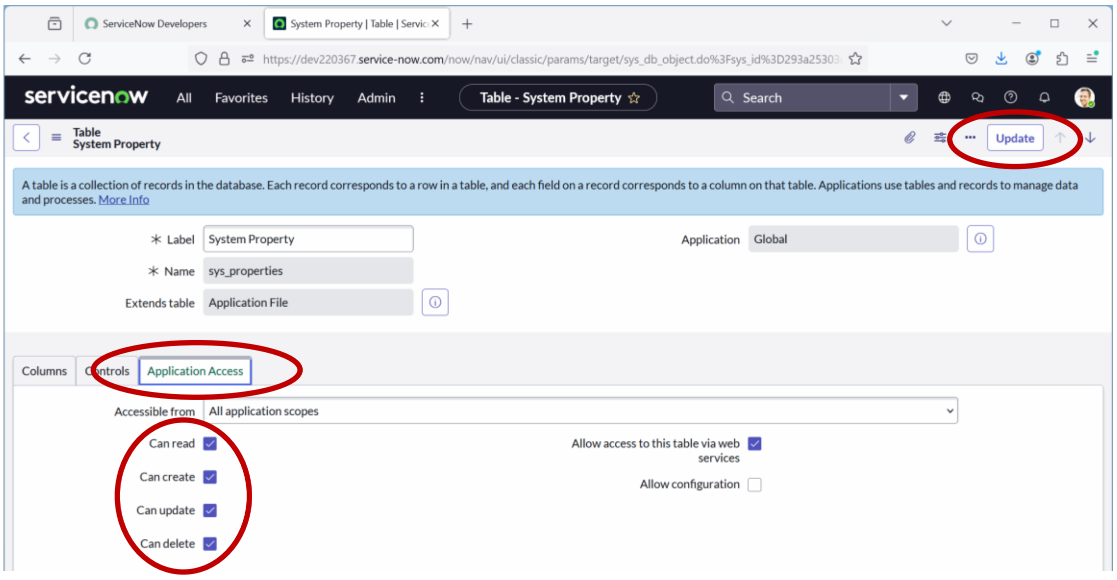


All > Update name is not empty > Name >= sys_properties

Label	Name	Extends table	Ex
Search	Search	Search	S
System Property	sys_properties	Application File	fal
System Property Category	sys_properties_category	Application File	fal
Category Property	sys_properties_category_m2m	Application File	fal
Protocol Profile	sys_protocol_profile	(empty)	fal

Figure 9 Select `sys_properties` table

Ensure all Application Access checkboxes are checked and click **`Update`**.



ServiceNow Developers
System Property | Table | Servic
https://dev220367.service-now.com/now/nav/ui/classic/params/target/sys_db_object.do%3Fsys_id%3D293a25303
servicenow
All
Favorites
History
Admin
Table - System Property
Search
Update
Columns
Controls
Application Access

Table
System Property

A table is a collection of records in the database. Each record corresponds to a row in a table, and each field on a record corresponds to a column on that table. Applications use tables and records to manage data and processes. More Info

* Label	System Property	Application	Global
* Name	sys_properties		
Extends table	Application File		

Accessible from	All application scopes		
Can read	✓	Allow access to this table via web services	✓
Can create	✓	Allow configuration	☐
Can update	✓		
Can delete	✓		

Figure 10 Enable All Application Access checkboxes

Complete Installation

There is no formal installation for **Meta-Update**.

Meta-Update is meant to be run from a command line or shell or fired by ARS workflow, or other automated processes, include the Meta-Update Job Control application.

By expanding the distribution file, you have completed the installation of Meta-Update.

You may choose to include the Meta-Update distribution directory on you path and your library path. See Runtime Environment below for more information.

The Meta-Update Job Console application is an optional add-on to Meta-Update that allows multiple administrators to run and develop Meta-Update jobs and templates. See overleaf.

About Job Console Installation

The **Meta-Update Job Console** is a Remedy Application that you may choose to install.

When this application is installed, Job Templates are also installed. You can build templates for your own scripts and designated users may fire these Meta-Update jobs.

These will be controlled by a Job Queue server which can be run on any server or workstation.

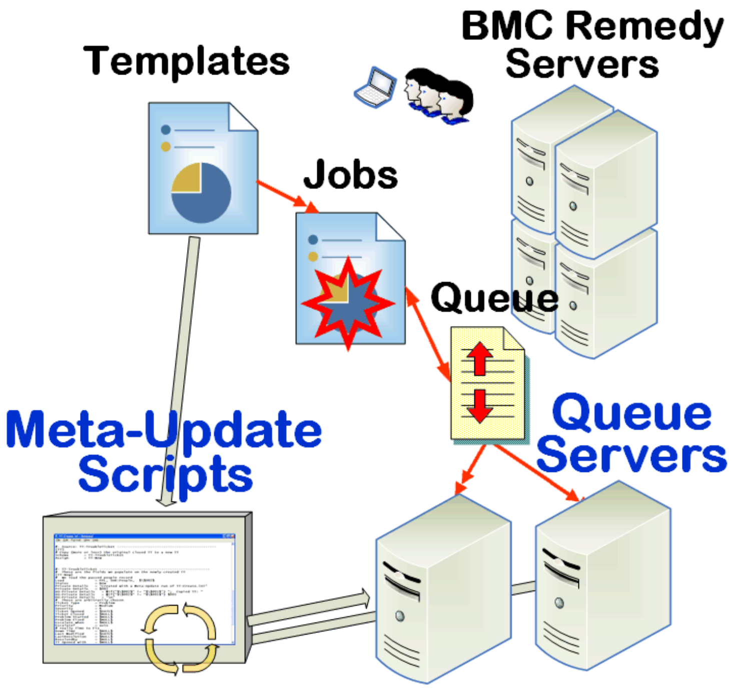


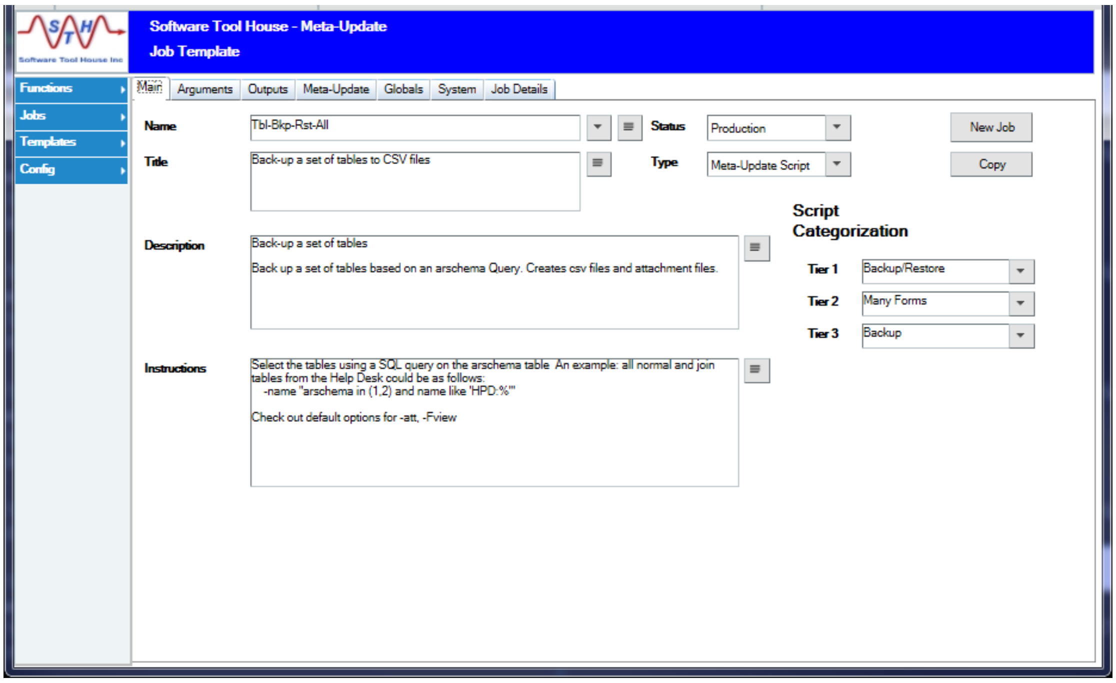
Figure 11   Overview of the Job Console Application

The installation of the Console application is done by executing a Meta-Update script. That script, when run for the first time, will import the application, set up the Configuration, and load the sample data based on your arguments.

Note that you do ***not*** need to install the Job Console to gain benefit from Meta-Update or to create and use Meta-Update scripts.

However, if you chose to do so, you can open Meta-Update for a set of users and allow them to run your own and supplied scripts.

You can control where Job Queue servers run and how many threads they have.



Name: Tbl-Bkp-Rst-All

Status: Production

Title: Back-up a set of tables to CSV files

Type: Meta-Update Script

Description: Back-up a set of tables

Back up a set of tables based on an arschema Query. Creates csv files and attachment files.

Instructions:

```
Select the tables using a SQL query on the arschema table  An example: all normal and join
tables from the Help Desk could be as follows:
   -name "arschema in (1,2) and name like 'HPD:%'"

Check out default options for -att, -Fview
```

Script Categorization

Tier 1: Backup/Restore

Tier 2: Many Forms

Tier 3: Backup

Figure 12   A Job Template referencing a sample script

Defining a Fixed User

Meta-Update runs against a Remedy or ServiceNow server and so must sign-on to that server.

Most installations of Meta-Update use a Meta-Update user similar to an automatically created Remedy Administrator user used for system functions.

Certain functions of the API such as SQL Queries cannot be performed without the Administrator privilege. Furthermore, without the Administrator privilege, some data may not be readable to the Meta-Update user.

Similarly, in ServiceNow, an Administrator has access to all fields and all data.

Specifying a user such as “Meta-Update-Robot”, causes the Modified-by and Modified date to be updated and easily searchable.

With the Job Control application, this fixed user is used to initiate a job and to load the results of the job.

The user firing the job is used for his own job’s data changes.

This fixed user is also used for the Job Queue server.

Run Time Environment

Meta-Update runs in a Windows "Command Prompt" or UNIX shell. It is a simple process that can be fired by workflow, batch files, shell scripts, even Meta-Update scripts.

Scripts and files developed and referenced may be interchanged freely between Window and UNIX.

Meta-Update scripts can be run

- through the Meta-Update Job Console application
- manually in a shell or command prompt
- in a filter with the $PROCESS$ actions
- through a batch file or shell or Perl script
- through an OS scheduler like **cron** or **at**

The runtime environment is the same for workflow, script, and manual operation.

The Meta-Update "**`bin`**" directory contains all required Meta-Update binaries or executable programs, shared objects, dlls, Windows batch files and Linux shell scripts.

The Meta-Update **`bin`** directory should be on the path.

On Windows, the Meta-Update "**`bin`**" directory needs to be in the PATH= environment variable.

```
Set PATH=D:\Apps\Sth\Meta-Update-6.00\;%PATH%
```

The program operates in a Command Prompt, or "DOS Box", or as a fired process. Local trace files are written in the current working directory by default.

On Solaris or Linux, the Meta-Update "**`bin`**" directory needs to be in the PATH= and LD_LIBRARY_PATH= environment variables.

```
export PATH=/apps/Sth/Meta-Update-6.00/bin/:$PATH
export LD_LIBRARY_PATH=/Apps/Sth/Meta-Update-
6.00/bin/:$LD_LIBRARY_PATH
```

The program operates under any of the available shells or as a spawned or background process. Local trace files are written in the current working directory when not specified.

Environment Variables

Both Meta-Update and the BMC Remedy API can be affected by using Environment Variables[1]. This section defines the Meta-Update environment variables and the values and behaviours associated with them.

BMC Remedy documentation is the accurate source for documentation on the BMC API environment variables. We summarize them here because they affect Meta-Update behaviour.

Meta-Update environment variables are fully defined below:

Environment Variable	Description
`SthScriptPath`	A path-like environment variable for finding Meta-Update scripts and files.
`SthApiRetry`	Allows Meta-Update to retry API operations on any BMC Remedy API errors or during server outages.
`SthMupdLic`	Specifies the Meta-Update license key for the main server.

BMC Remedy API environment variables are specified in the BMC provided documentation. The usage of these variables may change at any time. This list is included for convenience and because it affects and overrides Meta-Update behaviours.

Environment Variable	Description
`ARAPILOGGING`	Generates one or two files in the current working directory of the running Meta-Update process. Conflicts will occur when multiple Meta-Update processes with this environment variable are run. Use **`ARAPILOGGING=61`** to create a single file, apires.log, in the current working directory, containing all API calls, statuses, timings, and result data. **`ARAPILOGGING=29`** is as above but will not log result data. Note that native Meta-Update logging allows you granular control
`ARTCPPORT`	Sets **all connections** TCP Port to the servers. ***Overrides*** the Meta-Update **`Port=`** keyword which can be different for different servers.
`ARRPC`	Specifies a private RPC port for all server connections.

Script Path Environment Variable

[1] "**Environment variables** are a set of dynamic named values that can affect the way running processes will behave on a computer." - Wikipedia

Scripts may be specified on the command line or may be found by searching an **SthScriptPath** environment variable.

SthScriptPath is set the same way as PATH according to the OS that Meta-Update is running on.

On Windows, one could set the script path like this:

```
set SthScriptPath=E:\Projects\ITSM\Scripts;D:\Apps\STH\include\;
```

On LINUX, one could set the path like so:

```
export SthScriptPath=/Projects/ITSM/Scripts/:/Apps/STH/samples:
```

Note the difference in the path and directory separators.

Subdirectories in the paths are not searched. However if the script passed to the command line contains a relative path, that relative path will be checked against the **SthScriptPath** and the first matching file will be opened.

API Retry Environment Variable

A Meta-Update job normally returns any errors received from the ARS server during any of its API calls and cancels the single record it was processing. It would then continue with the next record.

It is useful to protect the Meta-Update run from a server timeout, crash, or restart. Meta-Update can retry some API calls to the server based on configurable ARERR codes, a maximum number of retries, and a delay between retries.

The environment variable **SthApiRetry** may be used to specify these retry settings.

Without this environment variable, all API calls that fail cause an error in Meta-Update that can result in a record being lost, not found, or the Meta-Update job terminating before processing all records of a query.

The **SthApiRetry=** string is either a single or multiple sets of three numbers:

```
start_ARERR_number [ - stop_ARERR_number ] Retries Delay
```

start_ARERR_number [- stop_ARERR_number]	Single or ranges of ARERR numbers can be specified.
Retries	A **Retry** count of 0 means infinite number of retries.
Delay	The **Delay** is in seconds. A **Delay** of 0 means no delay.

The following example illustrates its use to protect against server crashes and server time-outs.

```
set SthApiRetry=90-92 24 300 93 10 180
```

```
export SthApiRetry=90-92 24 300 93 0 180
```

This example retries API calls resulting in ARERR 90, 91, 92 – connection or server down: 24 times with a 5 minute delay, for a total of 2 hours, and, in ARERR 93 – timeout due to busy server: 10 times with a 3 minute delay for a total of 30 minutes.

Note that for Query timeouts (94), retries will generally not resolve the problem.

Instead use the **`TimeOutLong=`** keyword of the **`[Main]`** section, or the **`-TmeOutL`** argument to increase the default.

fs

License Environment variable

```
    SthMupdLic      =      license-key
```

If this environment variable is defined, the license check is made against the value associated unless overridden.

This is set automatically with the **`SthLic.cmd`** or **`SthLic.sh`** batch file / shell script.

Other Environment variables

```
    AnyVar          =      Value
```

Any environment variable may be used in a Meta-Update script. All defined environment variables are referenced by the reserved tag, **`ENV`**. The field name is the environment variable name.

Environment variables, like all other field names are case sensitive.

```
       Loop         =      String, Pth, ";", $ENV, PATH$
```

The above example loops for every directory in the PATH environment variable.

As another example, the environment variable, **`ArsGlobals = 5`**, could be used to load a site-specific set of values and keys to other records.

```
    LoadQ    =      Tag, Schema, '1' = $ENV, ArsGlobals$
```

Using SthLic.cmd or SthLic.sh files

A batch file or shell script is generated that sets Environment variables to Server, User, Password, Licensing values.

When generated, the batch file will support any number of ARS and ServiceNow servers.

This allows to easily switch servers and run the same commands. Here, we save the product catalogue CTI only on three different servers:

```
SthLic.cmd     dev
SthMqry.exe    -f  prod-cat-dev.csv -S "PCT:Product Catalog"
                "'Manufacturer' = \"\""
SthLic.cmd     qa
SthMqry.exe    -f  prod-cat-qa.csv -S "PCT:Product Catalog"
                "'Manufacturer' = \"\""
SthLic.cmd     prod
SthMqry.exe    -f  prod-cat-prod.csv -S "PCT:Product Catalog"
                "'Manufacturer' = \"\""
```

You now have three files for three environments that can be easily compared.

On Linux, the generated file must be set to be executable once. To do so, enter some variation of **chmod**

```
>  chmod     +x               ./SthLic.sh
>  chmod     +--x--x--x       ./SthLic.sh
```

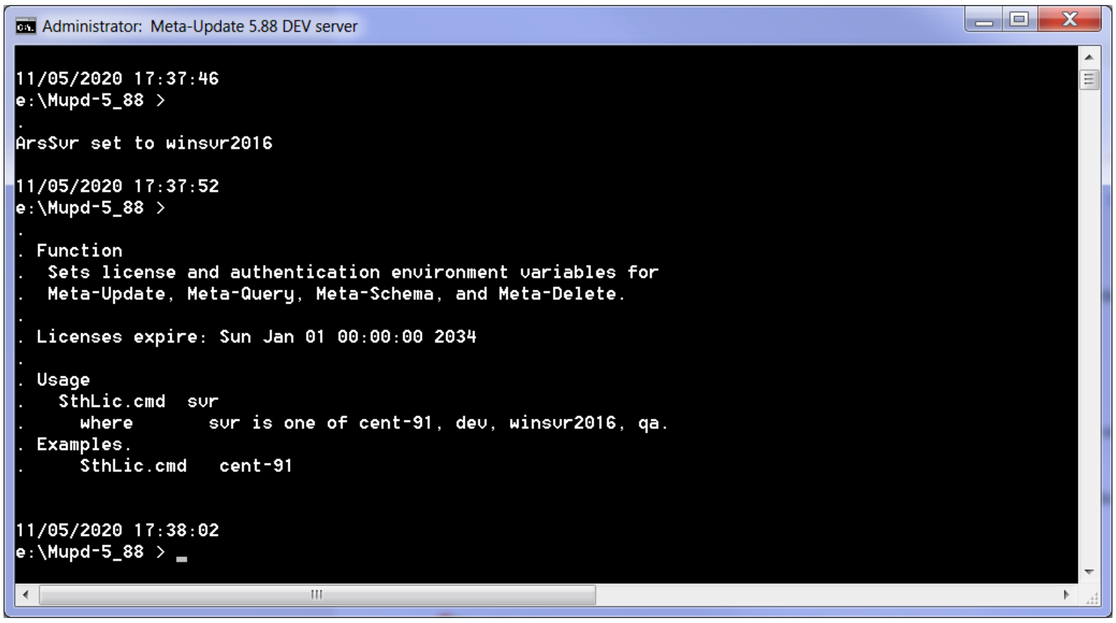
The shell script needs to be "Sourced" or any changes made to environment variables will be lost upon its completion.

When executing the shell script, "source it" by prefixing the command invocation with a dot, as follows:

```
> . ./SthLic.sh
```

On Windows, simply execute the batch file normally:

> .\SthLic.cmd



```
11/05/2020 17:37:46
e:\Mupd-5_88 >
.
ArsSvr set to winsvr2016

11/05/2020 17:37:52
e:\Mupd-5_88 >
.
. Function
.   Sets license and authentication environment variables for
.   Meta-Update, Meta-Query, Meta-Schema, and Meta-Delete.
.
. Licenses expire: Sun Jan 01 00:00:00 2034
.
. Usage
.    SthLic.cmd  svr
.      where       svr is one of cent-91, dev, winsvr2016, qa.
. Examples.
.      SthLic.cmd   cent-91


11/05/2020 17:38:02
e:\Mupd-5_88 > _
```

User Password Encryption

`SthLicUpd.exe`. generates `SthLic.cmd` or `SthLic.sh` and encrypts ARS and ServiceNow user passwords.

Once encrypted, only the same OS user that encrypted the password can use that password.

`SthLic.cmd` and `SthLic.sh` set environment variables for a licensed server and authentication can be automatically generated on all supported platforms – see below.

These files will allow the setting of environment variables by specifying the desired server, so that for example, a query can be run against one server and then run against another server.

All utilities bundled with Meta-Update will accept either plain text or encrypted passwords in all the methods that passwords may be specified: on the command line, in scripts, or, in environment variables.

If using ARS password encryption, the supplied passwords must be encrypted for each Windows or Unix user that will use Meta-Update. The encryption / decryption is dependent on the currently signed on user.

A new version of the files `SthLic.cmd` and `SthLic.sh` must be generated for each Windows or Unix user even if the ARS User is the same.

This means that when using ARS Password Encryption, the files, `SthLic.cmd` and `SthLic.sh` cannot be copied from machine to machine, and, if a single machine is used by more than one user, different `SthLic.cmd` and `SthLic.sh` files will need to be used by each user.

SthLicUpd.exe Utility

This utility can be used to encrypt user passwords and to generate an `SthLic.cmd` and `SthLic.sh` scripts based on license files.

The utility is available on all supported platforms and may be run in prompt mode where it will ask you for all needed information.

Easy server names, Server IPs, Ports, Users, and Passwords can be set. ARS and ServiceNow passwords are by default encrypted.

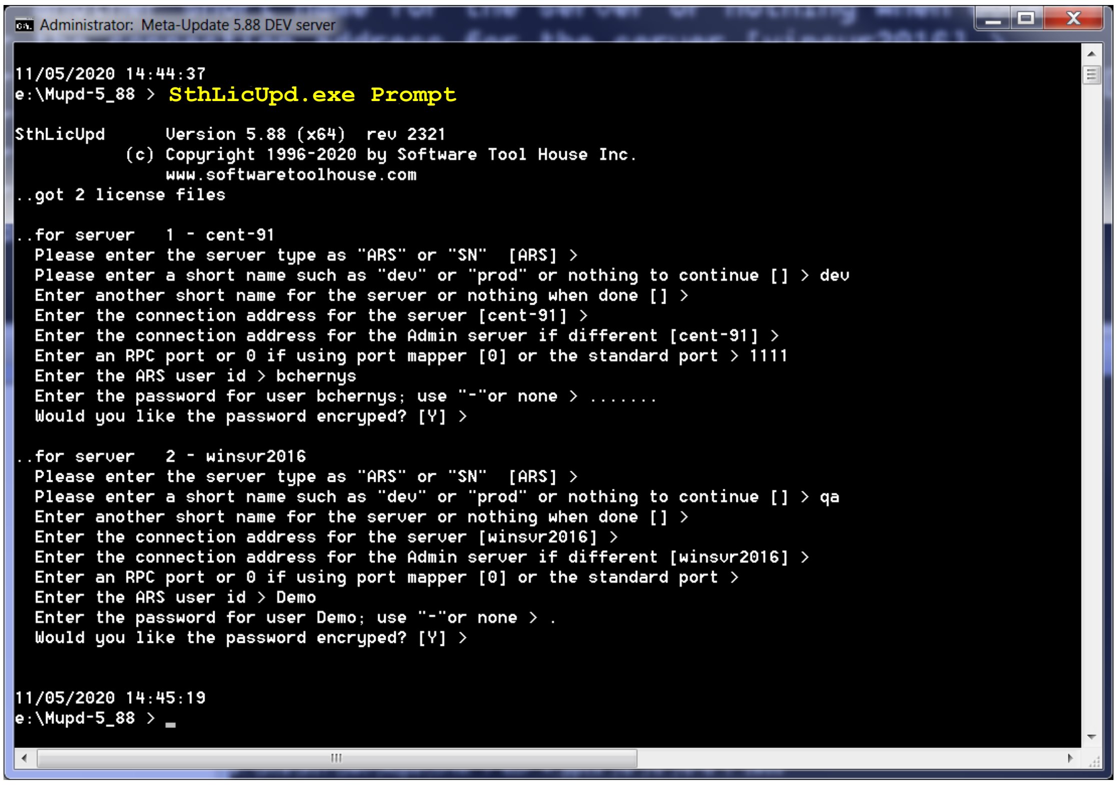
The **`SthLicUpd.exe`** utility will generate only one of the `SthLic.cmd` and `SthLic.sh` files as appropriate for the system that it is being run on.

Files produced by **`SthLicUpd`** containing encrypted passwords are not transferrable across platforms or users. You have the choice to encrypt a user password.

Once an `SthLic.cmd` and `SthLic.sh` are built, they can be used to set Environment Variables that define the target server and licensing. Of-course, they can be manually edited, adding paired servers for example. For example, when archiving from one server to another.

Steps need to generate the `SthLic.cmd` and `SthLic.sh` file:

1 Copy license file(s) into the bin directory of the Meta-Update installation
2 Decide and have available the Remedy or ServiceNow User that Meta-Update will login with
3 Run **`SthLicUpd.exe Prompt`** as in this example with two license files:

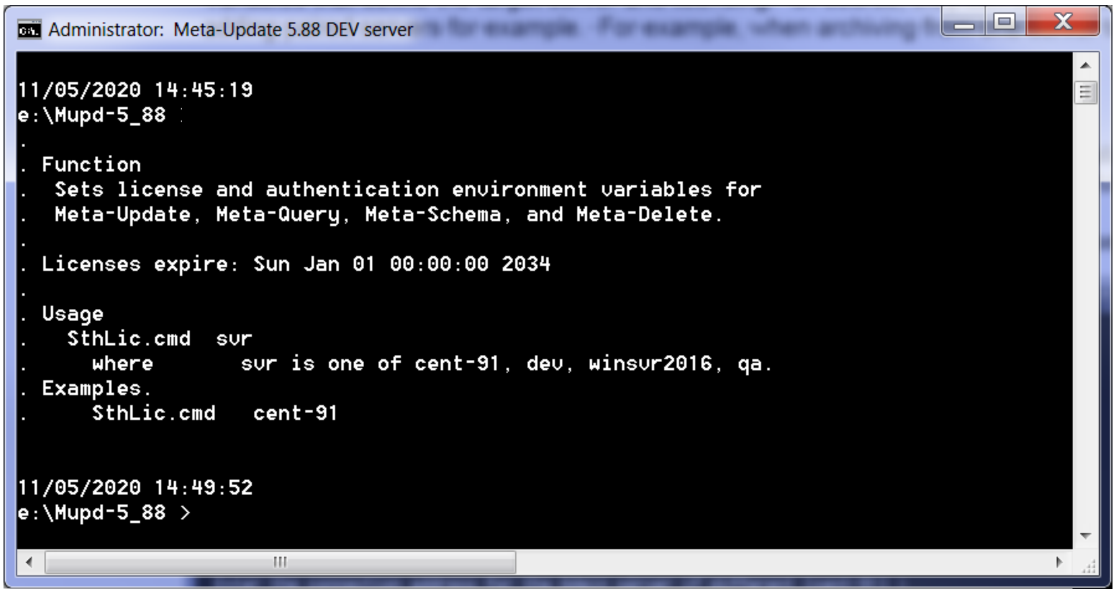


Administrator: Meta-Update 5.88 DEV server

```
11/05/2020 14:44:37
e:\Mupd-5_88 > SthLicUpd.exe Prompt

SthLicUpd      Version 5.88 (x64)  rev 2321
           (c) Copyright 1996-2020 by Software Tool House Inc.
               www.softwaretoolhouse.com
..got 2 license files

..for server    1 - cent-91
  Please enter the server type as "ARS" or "SN"  [ARS] >
  Please enter a short name such as "dev" or "prod" or nothing to continue [] > dev
  Enter another short name for the server or nothing when done [] >
  Enter the connection address for the server [cent-91] >
  Enter the connection address for the Admin server if different [cent-91] >
  Enter an RPC port or 0 if using port mapper [0] or the standard port > 1111
  Enter the ARS user id > bchernys
  Enter the password for user bchernys; use "-"or none > .......
  Would you like the password encryped? [Y] >

..for server    2 - winsvr2016
  Please enter the server type as "ARS" or "SN"  [ARS] >
  Please enter a short name such as "dev" or "prod" or nothing to continue [] > qa
  Enter another short name for the server or nothing when done [] >
  Enter the connection address for the server [winsvr2016] >
  Enter the connection address for the Admin server if different [winsvr2016] >
  Enter an RPC port or 0 if using port mapper [0] or the standard port >
  Enter the ARS user id > Demo
  Enter the password for user Demo; use "-"or none > .
  Would you like the password encryped? [Y] >


11/05/2020 14:45:19
e:\Mupd-5_88 > _
```

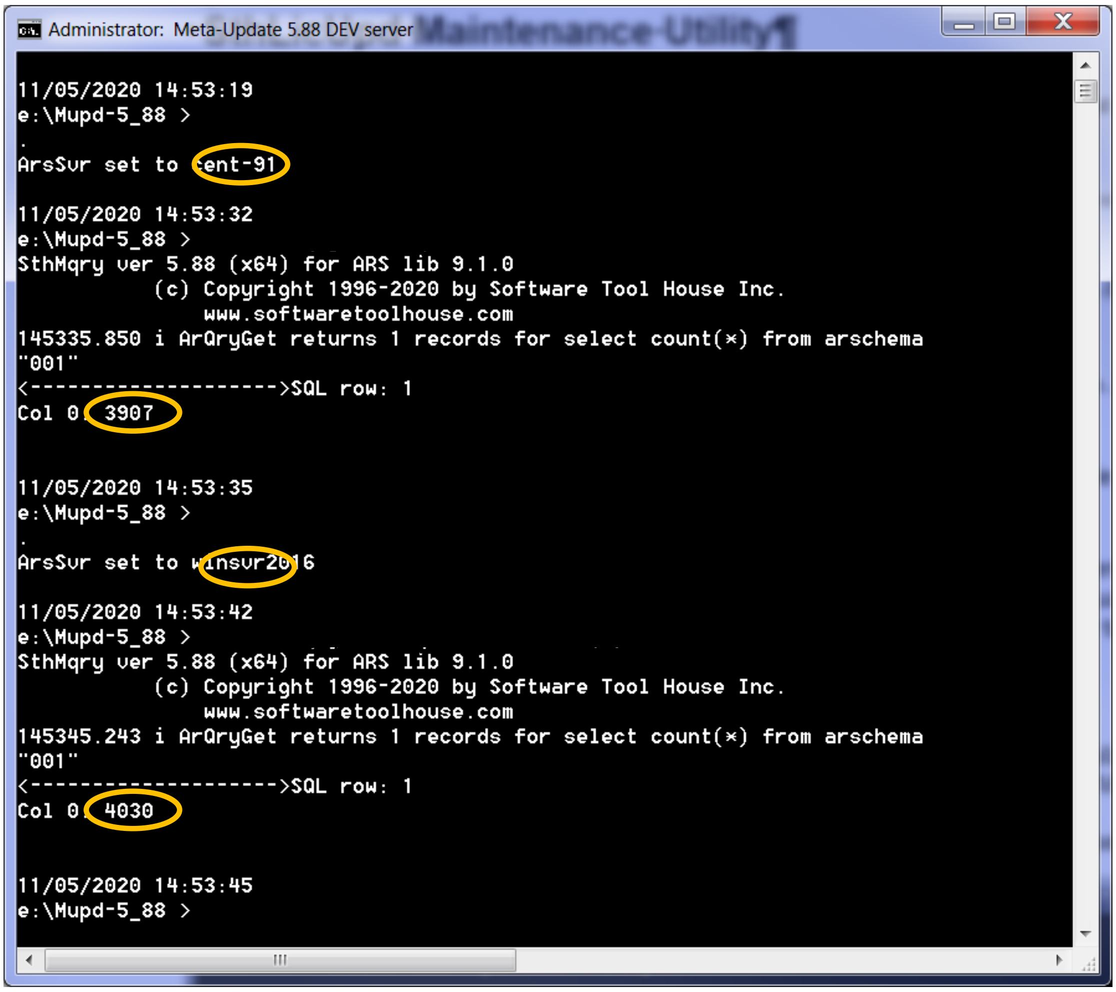
Figure 13 Sample SthLicUpd Prompt session with two license files

The generated file will give usage instructions:


Administrator: Meta-Update 5.88 DEV server

```
11/05/2020 14:45:19
e:\Mupd-5_88
.
. Function
.  Sets license and authentication environment variables for
.  Meta-Update, Meta-Query, Meta-Schema, and Meta-Delete.
.
. Licenses expire: Sun Jan 01 00:00:00 2034
.
. Usage
.   SthLic.cmd  svr
.     where       svr is one of cent-91, dev, winsvr2016, qa.
. Examples.
.     SthLic.cmd   cent-91


11/05/2020 14:49:52
e:\Mupd-5_88 >
```

Figure 15 Generated SthLic.cmd Usage


Administrator: Meta-Update 5.88 DEV server

```
11/05/2020 14:53:19
e:\Mupd-5_88 >
.
ArsSvr set to cent-91

11/05/2020 14:53:32
e:\Mupd-5_88 >
SthMqry ver 5.88 (x64) for ARS lib 9.1.0
           (c) Copyright 1996-2020 by Software Tool House Inc.
               www.softwaretoolhouse.com
145335.850 i ArQryGet returns 1 records for select count(*) from arschema
"001"
<-------------------->SQL row: 1
Col 0: 3907


11/05/2020 14:53:35
e:\Mupd-5_88 >
.
ArsSvr set to winsvr2016

11/05/2020 14:53:42
e:\Mupd-5_88 >
SthMqry ver 5.88 (x64) for ARS lib 9.1.0
           (c) Copyright 1996-2020 by Software Tool House Inc.
               www.softwaretoolhouse.com
145345.243 i ArQryGet returns 1 records for select count(*) from arschema
"001"
<-------------------->SQL row: 1
Col 0: 4030


11/05/2020 14:53:45
e:\Mupd-5_88 >
```

Figure 14 Example of issuing the same query on two servers using SthLic.cmd

SthLicUpd.exe Usage

```
SthLicUpd      Version 5.88 (x64)  rev 2321
           (c) Copyright 1996-2020 by Software Tool House Inc.
               www.softwaretoolhouse.com

Function:
  SthLicUpd is used to generate Meta-Update SthLic.cmd files and
    encrypt ARS Users' passwords.

Modes:
  SthLicUpd can be run in different modes:
      Prompt     will scan license files, prompt for needed info,
                   and generate a new SthLic.cmd.
      Pwd        will encrypt ARS users passwords
      -pwd  xx   will encrypt the single suppied ARS pwd and
                   write only the encrypted value to stdout
Synopsis:
  SthLicUpd   mode   [ switches ]

  where  mode is one of:     Prompt or  Pwd

      Prompt             Run in interactive mode to scan licenses
      Pwd                Will encrypt ARS users' passwords

    switches             are as follows:
      -licpath   path    The directory path of license files
      -out       file    Output file name if not SthLic.cmd/sh
               Default for -out, -licpath is the bin directory
                 where this utility was found and SthLic.cmd/sh
    other switches
      -d                 Specifies full tracing

Examples:
      SthLicUpd Prompt  -licpath /apps/STH/bin/
                        -out ~/bin/SthLic.sh
      SthLicUpd Prompt  -licpath E:\apps\STH\bin\
                        -out C:\Users\apps\bin\SthLic.cmd
```

Prompt mode will find all available *.lic files in a single directory and ask for any needed information. It will then generate a new `SthLic.cmd` or `SthLic.sh` file.

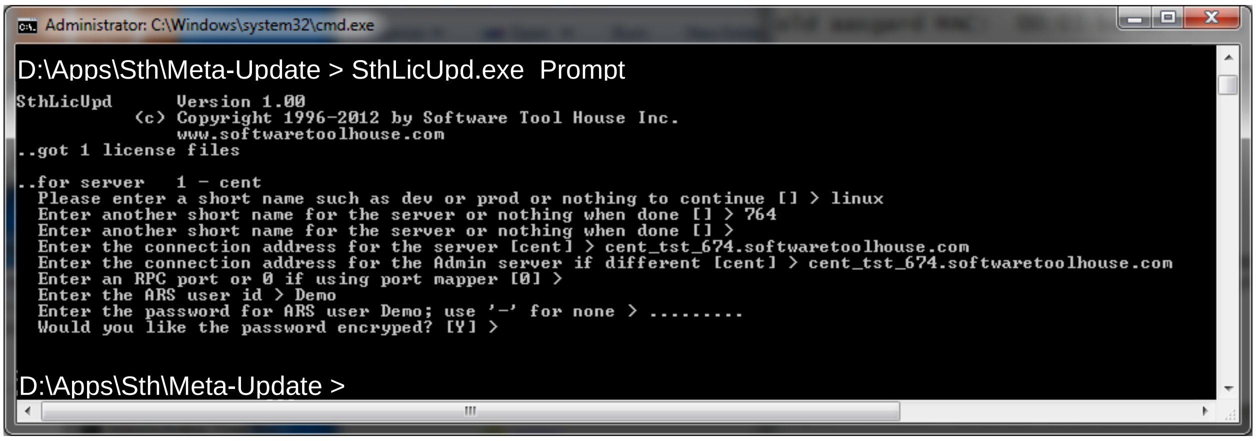
These files will set the environment variable for ARS server connectivity and authentication which all Meta-Update utilities will automatically pick up. These become defaults and switches can be used to override them.

Password mode will simply allow you to encrypt any number of ARS User passwords. You can then use these encrypted password strings in any of the Meta-Update utilities to authenticate to the ARS server.

Strategies for using SthLic & SthLicUpd.

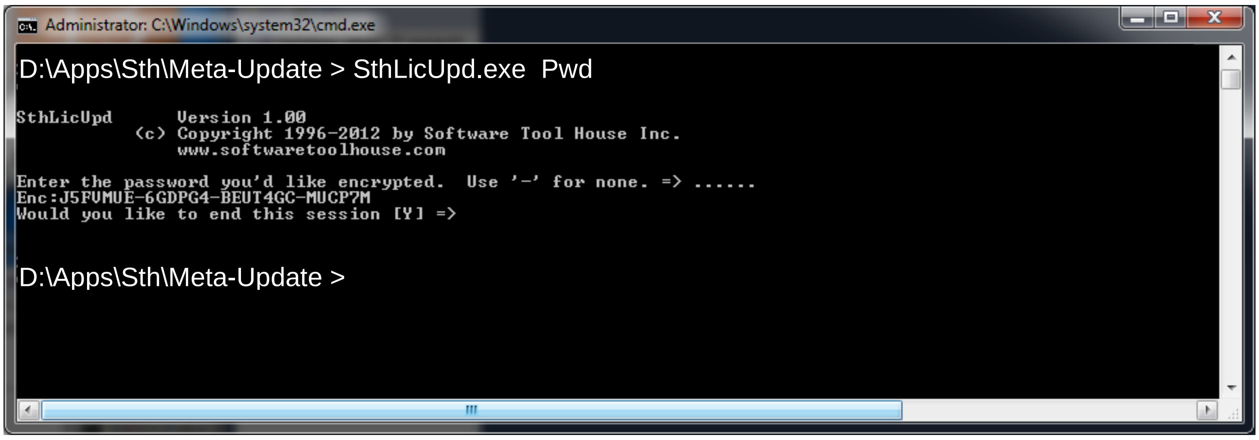
Different customers had different strategies for managing users and SthLic files. Some are outlined here to give an idea of the possibilities.

Sample Prompt Session:



```
D:\Apps\Sth\Meta-Update > SthLicUpd.exe  Prompt
SthLicUpd       Version 1.00
           (c) Copyright 1996-2012 by Software Tool House Inc.
               www.softwaretoolhouse.com
..got 1 license files

..for server    1 - cent
  Please enter a short name such as dev or prod or nothing to continue [] > linux
  Enter another short name for the server or nothing when done [] > 764
  Enter another short name for the server or nothing when done [] >
  Enter the connection address for the server [cent] > cent_tst_674.softwaretoolhouse.com
  Enter the connection address for the Admin server if different [cent] > cent_tst_674.softwaretoolhouse.com
  Enter an RPC port or 0 if using port mapper [0] >
  Enter the ARS user id > Demo
  Enter the password for ARS user Demo; use '-' for none > .........
  Would you like the password encrypted? [Y] >

D:\Apps\Sth\Meta-Update >
```

Sample Password Session:



```
D:\Apps\Sth\Meta-Update > SthLicUpd.exe  Pwd

SthLicUpd       Version 1.00
           (c) Copyright 1996-2012 by Software Tool House Inc.
               www.softwaretoolhouse.com

Enter the password you'd like encrypted.  Use '-' for none. => ......
Enc:J5FVMUE-6GDPG4-BEUT4GC-MUCP7M
Would you like to end this session [Y] =>

D:\Apps\Sth\Meta-Update >
```

Running Meta-Update

In this section, we will cover:

- Setting up the run time environment
- BMC Remedy API versions
- Meta-Update program versions
- Using the license keys
- Environment variables
- The Meta-Update command line usage
- Meta-Update output and return values
- Meta-Update Tracing

Setting Up the Run-Time Environment

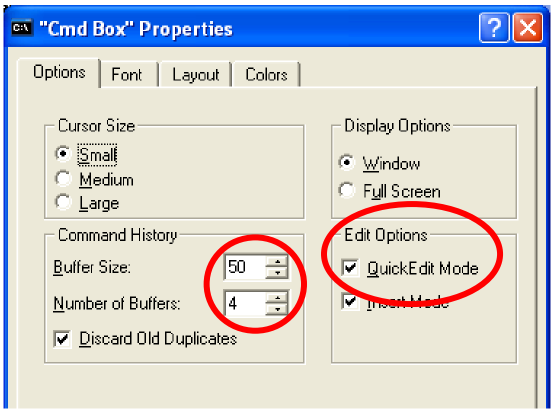
Meta-Update runs in a Windows "Command Prompt" or UNIX shell.

Ideal Command Prompt Settings

In Windows, it is helpful to change the Command prompt properties so that all buffers are bigger and “Quick Edit” is turned on.

Under the “Command Prompt” properties, “Layout” tab, increase the buffer width and depth. Adjust the Window size to taste.

In “Options”, make sure “QuickEdit” is turned on.



"Cmd Box" Properties

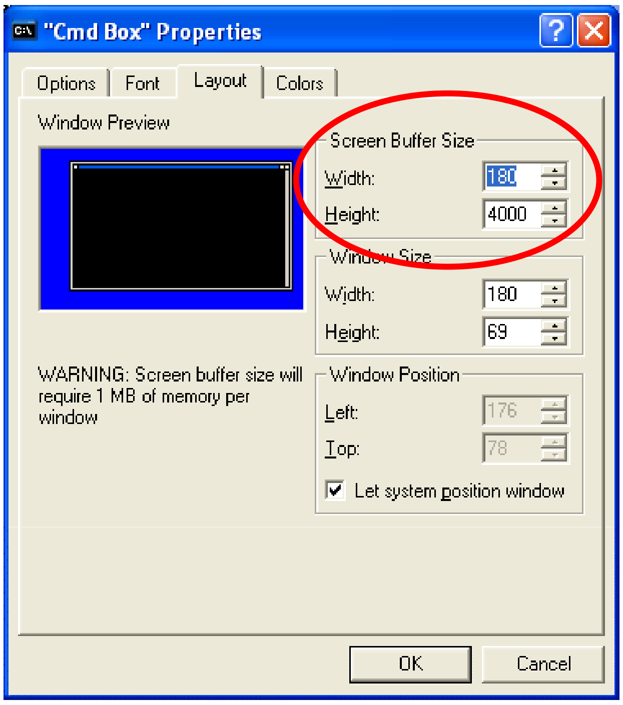
Options | Font | Layout | Colors

Cursor Size: [✓ Small] Medium Large

Display Options: [✓ Window] Full Screen

Command History
Buffer Size: 50
Number of Buffers: 4
[✓] Discard Old Duplicates

Edit Options
[✓] QuickEdit Mode
[✓] Insert Mode



"Cmd Box" Properties

Options | Font | Layout | Colors

Window Preview

Screen Buffer Size
Width: 180
Height: 4000

Window Size
Width: 180
Height: 69

WARNING: Screen buffer size will require 1 MB of memory per window

Window Position
Left: 176
Top: 78
[✓] Let system position window

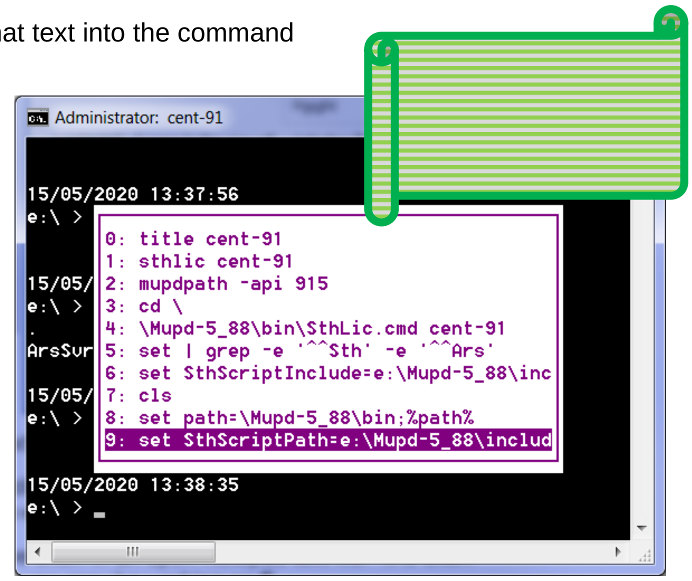
OK | Cancel

With QuickEdit, you can click and drag a rectangle of text in the Command Prompt and then right click to copy the text to the Clipboard. A single right click will paste that text into the command prompt.

In addition, Function Key **F7** will bring up a window with your old commands. Select one and press Enter to execute it. Press an arrow key to edit it before execution.

Tip: Keep commonly used commands and save often in an open Notepad text file. It is often useful to have one open for each different think you may be working on.

For Linux users, only note that the default Bash shell under some configuration have a low text buffer space.



Administrator: cent-91

```
15/05/2020 13:37:56
e:\ >            0: title cent-91
                 1: sthlic cent-91
15/05/           2: mupdpath -api 915
e:\ >            3: cd \
.                4: \Mupd-5_88\bin\SthLic.cmd cent-91
ArsSvr           5: set | grep -e '^^Sth' -e '^^Ars'
                 6: set SthScriptInclude=e:\Mupd-5_88\inc
15/05/           7: cls
e:\ >            8: set path=\Mupd-5_88\bin;%path%
                 9: set SthScriptPath=e:\Mupd-5_88\includ

15/05/2020 13:38:35
e:\ > _
```

Set the Path

It is often useful to have a path setting batch file or add the Meta-Update “bin” to the User or System environment variables.

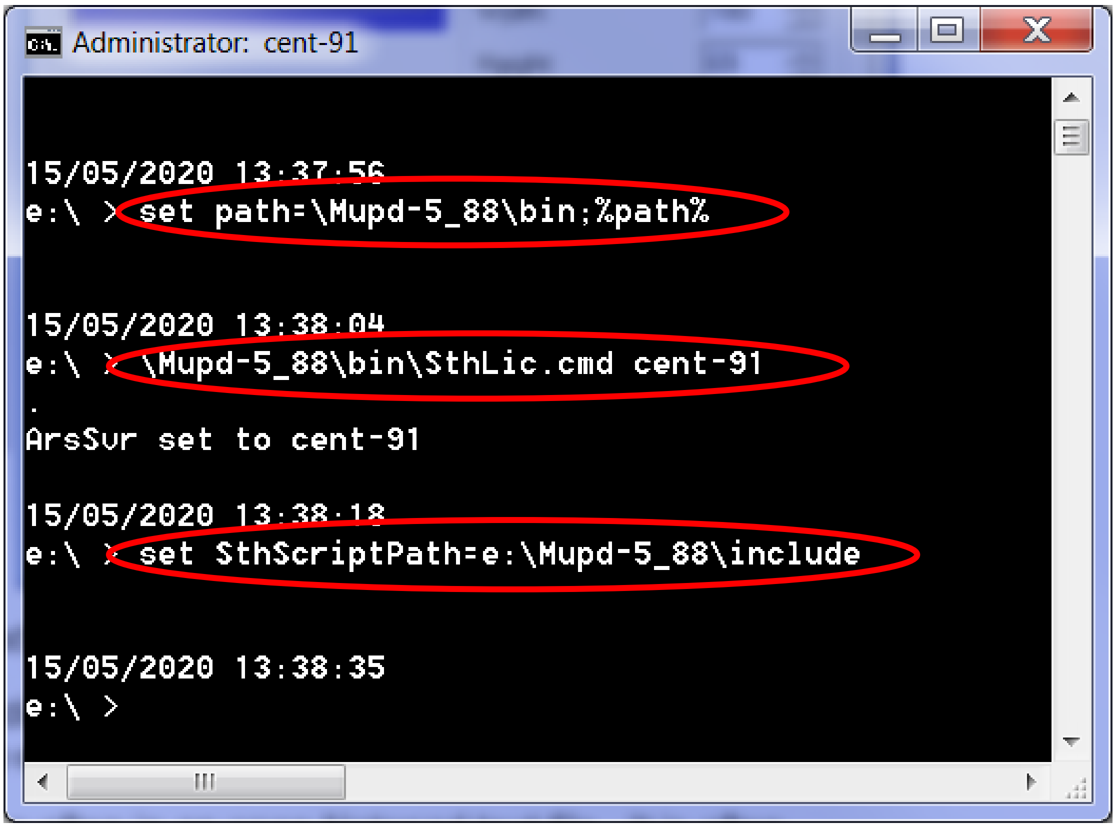
```
set  path-D:\apps\STH\Mupd_6.00\bin

export PATH=/apps/sth/mupd_6.00/bin/:$PATH
export LD_LIBRARY_PATH=/apps/sth/mupd_6.00/bin/:$ LD_LIBRARY_PATH
```

Set the Script Path

Many scripts use an include library. The standard include directory needs to be added to the `SthScriptPath` Environment variable.

Set Licensing and Server Defaults

The easiest way to set a target server with a user is through the use of SthLic.cmd or SthLic.sh.



```
15/05/2020 13:37:56
e:\ > set path=\Mupd-5_88\bin;%path%

15/05/2020 13:38:04
e:\ > \Mupd-5_88\bin\SthLic.cmd cent-91
.
ArsSvr set to cent-91

15/05/2020 13:38:18
e:\ > set SthScriptPath=e:\Mupd-5_88\include

15/05/2020 13:38:35
e:\ >
```

Figure 16   Setting The Path, Licensing and Script Path in Windows

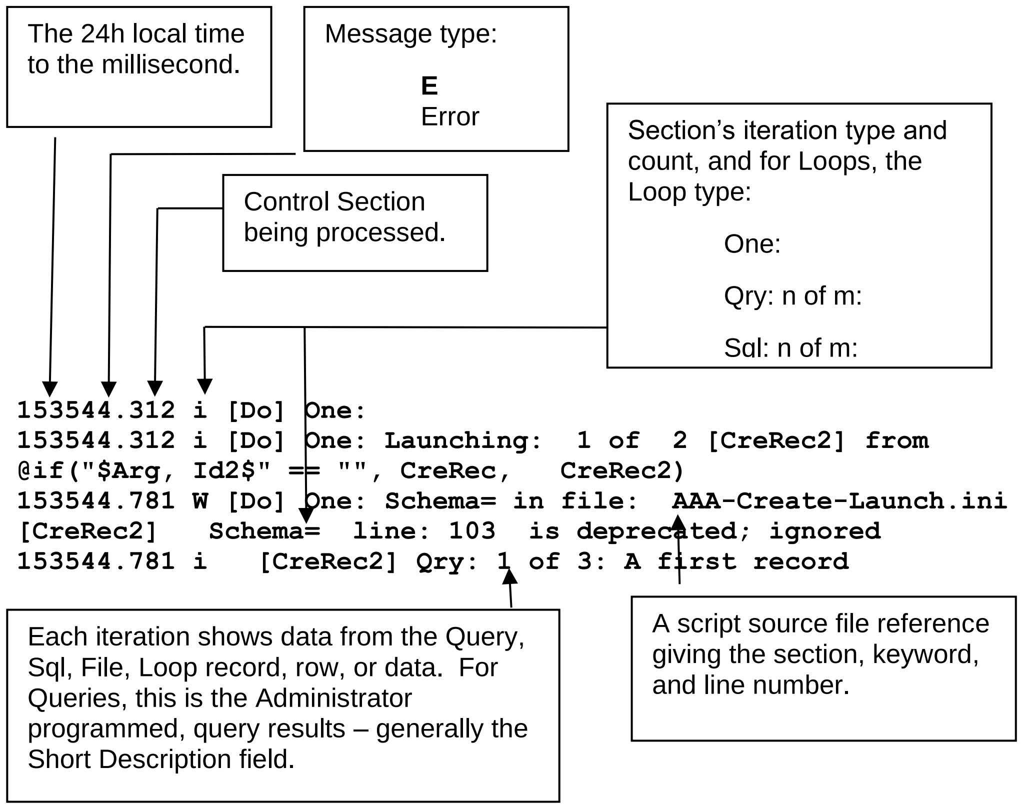
You are now ready to install the Meta-Update Job Console.

Program Output

Unless the –q switch is used, Informational, Warning, and Error messages are echoed to the console. These messages tell you what section is working on what record and lists outputs to ARS tables. These messages are also captured in the trace logs.

An example:

```
E:\Dta\_wrk\ > SthMupd.exe AAA-Create-Launch.ini Do -p 426 429
Meta-Update    Version 5.56 (x64) for ARS lib 8.1.2
           (c) Copyright 1996-2015 by Software Tool House Inc.
               www.softwaretoolhouse.com
153544.312 i [Do] One:
153544.312 i [Do] One: Launching:  1 of  2 [CreRec2] from @if("$Arg, Id2$" == "",
CreRec,   CreRec2)
153544.781 W [Do] One: Schema= in file:  AAA-Create-Launch.ini  [CreRec2]   Schema=
line: 103  is deprecated; ignored
153544.781 i   [CreRec2] Qry: 1 of 3: A first record
153544.890 i   [CreRec2] Qry: 1 of 3: Merged schema: _Test, Id: 000000000004474 OldId=
153544.921 i   [CreRec2] Qry: 2 of 3: and now, only seconds lat
153544.968 i   [CreRec2] Qry: 2 of 3: Merged schema: _Test, Id: 000000000004475 OldId=
153544.968 i   [CreRec2] Qry: 3 of 3: A second entry made a few
153545.031 i   [CreRec2] Qry: 3 of 3: Merged schema: _Test, Id: 000000000004476 OldId=
153545.031 i   [CreRec2] Qry: eof 3 record OK; 0 records with errors; total: 3.
153545.031 i [Do] One: Launching:  2 of  2 [CopyRec2] from @if("$Arg, Id2$" == "",
CopyRec,  CopyRec2)
153545.031 W [Do] One: Update0= in file:  AAA-Create-Launch.ini  [CopyRec2]   Update0=
line: 98  is deprecated.  Use AssignNew=
153545.031 i   [CopyRec2] Qry: 1 of 3: A first record
153545.125 i   [CopyRec2] Qry: 1 of 3: Merged schema: _Test, Id: 000000000004477
OldId=
153545.125 i   [CopyRec2] Qry: 2 of 3: and now, only seconds lat
153545.187 i   [CopyRec2] Qry: 2 of 3: Merged schema: _Test, Id: 000000000004478
OldId=
153545.187 i   [CopyRec2] Qry: 3 of 3: A second entry made a few
153545.234 i   [CopyRec2] Qry: eof 3 record OK; 0 records with errors; total: 3.
153545.234 i [Do] One: 1 record OK; 0 records with errors; total: 1.
153545.234 i Statistics:
153545.234 i        Sections:                    3
153545.234 i        Maximum section depth:       2
153545.234 i        Assignment Sections:         6
153545.234 i        Singleton Sections:          1   errors:      0
153545.234 i        Queries:                     2
153545.234 i        Query records:               6   errors:      0
153545.234 i        Output Schemas:              0
153545.250 i        Output Schema records:       6   created
153545.250 i        Output Schema records:       0   updated  (with 0 skipped)
153545.250 i        Outputs OK:                  6
153545.250 i        Outputs Errors:              0
153545.250 i        Outputs Aborts:              0
153545.250 i        Input   Errors:              0
153545.250 i terminating successfully in 1 sec.

E:\Dta\_wrk\ >
```



The 24h local time to the millisecond.

Message type:

E
Error

Control Section being processed.

Section's iteration type and count, and for Loops, the Loop type:

One:

Qry: n of m:

Sql: n of m:

```
153544.312 i [Do] One:
153544.312 i [Do] One: Launching:  1 of  2 [CreRec2] from
@if("$Arg, Id2$" == "", CreRec,   CreRec2)
153544.781 W [Do] One: Schema= in file:  AAA-Create-Launch.ini
[CreRec2]   Schema=  line: 103  is deprecated; ignored
153544.781 i   [CreRec2] Qry: 1 of 3: A first record
```

Each iteration shows data from the Query, Sql, File, Loop record, row, or data. For Queries, this is the Administrator programmed, query results – generally the Short Description field.

A script source file reference giving the section, keyword, and line number.

Firing from Workflow

Meta-Update may be fired from workflow as Run Process or Set Fields $PROCESS$ filter or active link.

When firing from workflow on the server, the environment is that of the ARS server process. It is prudent to code a script or batch file in the workflow and then have that script or batch file set up the environment for the run, invoke Meta-Update, and possibly do some termination activities.

The environment generally includes a path to the executable and to any required shared libraries or dlls, other environment variables, parameters, and the working directory.

As workflow is fired at independent times, it is possible for multiple copies of Meta-Update to be running simultaneously. If so, the Server based tracing version is highly recommended to properly serialise log files.

Developing Scripts

Normal Meta-Update runs will report script errors with an ‘E’ level message echoed to the console. That message will print the script file name, section, line number, and, if appropriate, the keyword being processed.

```
114159.531 E [Do] [asg-init] AssignInit apply was aborted in file:
  FD-SupGrp-Ren.ini  [asg-init]   @Cmd=  line: 74
```

Errors may be caused by different things:

- Syntax errors
- ARS reported errors such as unrecognised schema names or field names or labels
- LookUp or Load failures
- User Aborts

Meta-Update has several switches that will aid in script development which would normally not be used in production runs.

-e   single Error   With this switch, any error in any section will stop the run.

We recommend you use this switch when you develop and test scripts. You will generally not want it on production runs.

-v   Verbose   This prints all query qualifications and results to the console and to the log file.

We recommend you use this switch when you develop and test scripts. You will generally not want it on production runs.

-n   null   This switch prevents any ARS updates or creates.
This is only useful for the most simple of scripts as generally launched sections depend on access to a previous sections updated and reread record reference.

-d   Logging: Debug This should not normally be needed. It is intended to be used when using Meta-Update support. It provides complete debug level information on the job and generates masses of logs. You can also specify you want ARS client logging with this switch. See ***Tracing*** above for more information.

-g   Script Debugger   This invokes the Meta-Update script debugger. The script debugger allows you to set breakpoints and single step though your script’s operation. You can get debugging help, print your script, examine references, control breakpoints, and resume normal execution.

See **Script Debugging** below for more information about using the Meta-Update Script Debugger.

In this example, a script **Abort=** was set by an **AssignInit=** section that ensured there was at least one matching Support Organisation.

Another example where a bad value is passed as a script argument:

The **-v** switch echoes the exact query qualifications sent to the Remedy Server

```
E:\> SthMupd -v -e FD-SupGrp-Ren.ini Do -Org "Qelp Desk"
```

The script issues several "E" messages and then an abort.

Meta-Update tells you the script issued an Abort.

```
Meta-Update    Version 5.56 (x64) for ARS lib 8.1..
           (c) Copyright 1996-2015 by Software Too
               www.softwaretoolhouse.com
114159.515 q [Do] QuerySql: Svr: sthv1
114159.515 q [Do] QuerySql: Qualification: : 0000: select
count(*) from   CTM_Support_Group where  Support_Organiza
114159.515 q [Do] QuerySql: Qualification: : 0040: tion = 'Qelp
Desk'
114159.515 q [Do] QuerySql: returned 1 records of 1.
114159.515 i [Do] Msg: Found 0 records with:  'Support
Organization' == "Qelp Desk"
114159.515 E [Do] Msg: The Support Organisation argument must
match 1 or more records of CTM:Support Group"
114159.515 E [Do] Msg: Please check the spelling of your command
line argument."
114159.531 E [Do] Abort: ..aborting."
114159.531 E [Do] [asg-init] AssignInit apply was aborted in
file:  FD-SupGrp-Ren.ini  [asg-init]   @Cmd=  line: 74
114159.531 E IniRdo of FD-SupGrp-Ren.ini [Do] failed with 3 -
ArPutIini: Parm error 3
114159.531 i Statistics:
114159.531 i        Sections:                   1
114159.531 i        Maximum section depth:      1
114159.531 i        Output Schemas:             0
114159.531 i        Output Schema records:      0   created
114159.531 i        Output Schema records:      0   updated
(with 0 skipped)
114159.546 i        Outputs OK:                 0
114159.546 i        Outputs Errors:             0
114159.546 i        Outputs Aborts:             0
114159.546 i        Input   Errors:             0
114159.546 E error: some errors occurred.  Check for errors above
this message.
114159.546 E terminating unsuccessfully in 0 sec.
```

In this next example, the script file's **Query=** at line 65 referenced a ReadServer tag which was not defined as the script didn't need use additional servers.

```
    Query      = @Itsm6, User, User, $V, Qry$
```

```
E:\> SthMupd  QQQ-TblRpt-User.ini Do sthv1 Demo -start 1 -max 10
```

Error: Server reference not found.

Script line number in error.

Source line in error.

```
Meta-Update    Version 5.56 (x64) for ARS lib 8.1/2
           (c) Copyright 1996-2015 by Software Tool House Inc.
               www.softwaretoolhouse.com
113416.785 i [Do] One:
113416.785 i [Do] One: Launching:  1 of  1 [Do1]
113416.785 E [Do] One: FlIniFindCtl: Server Tag: Itsm6 not found
113416.785 E [Do] One: ArIiniQuery: FlIniRefFindCtl for Itsm6
failed at file:  QQQ-TblRpt-User.ini  [Do1]   Query=  line: 65
113416.785 E [Do] One: ArPutIiniRinit: ArIiniQuery failed (rc=4)
in file:  QQQ-TblRpt-User.ini  [Do1]   Query=  line: 65
113416.785 E [Do] One: ArPutIiniRinit for Do1 returned 3 -
ArPutIini: Parm error 3
113416.785 E [Do] One: ArPutIiniRdo: DoLaunch failed!
113416.801 E [Do] One: 0 record OK; 1 records with errors; total:
1.
113416.801 E IniRdo of QQQ-TblRpt-User.ini [Do] failed with 3 -
113416.801 i Statistics:
113416.801 i         Sections:                    1
113416.801 i         Maximum section depth:       1
113416.801 i         Singleton Sections:          1   errors:
0
113416.801 i         Output Schemas:              0
113416.801 i         Output Schema records:       0   created
113416.801 i         Output Schema records:       0   updated
(with 0 skipped)
113416.801 i         Outputs OK:                  0
113416.817 i         Outputs Errors:              0
113416.817 i         Outputs Aborts:              0
113416.817 i         Input   Errors:              0
113416.817 E error: some errors occurred.  Check for errors above
this message.
113416.817 E terminating unsuccessfully in 0 sec.
```

Installing the Meta-Update Job Console

Understanding the Job Console application

A **Job Template** defines a use of a Meta-Update script, or a "Job". A single script can be used in many Job Templates. The Template specifies arguments, configurations, files needed, the script's location, etc.

For example, when the Meta-Update distribution is expanded, a samples directory is created. These samples, have templates defined to use them.

A **Job** is based on a Job Template. The user selects a Job Template, creates a new Job, adds / changes / inspects arguments and then "Fires" the job, placing it on a Queue.

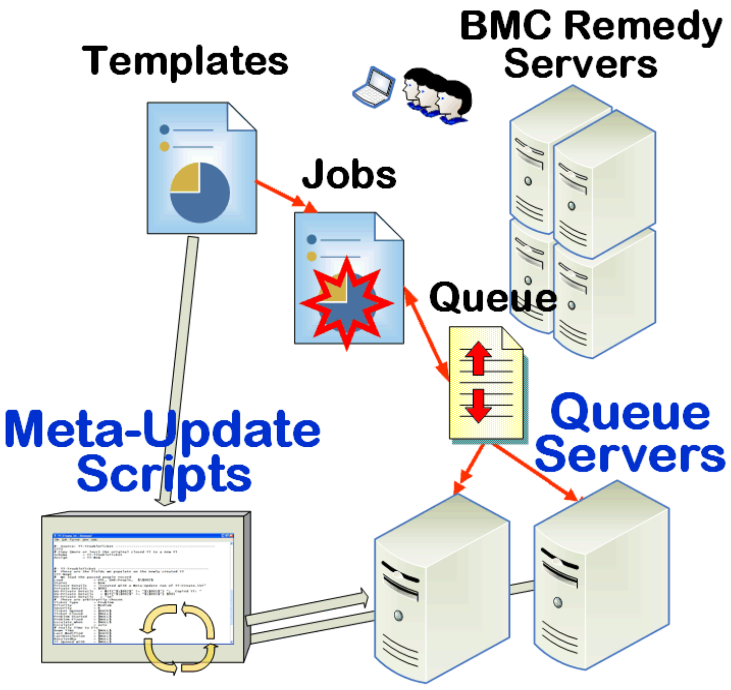


Figure 17 Overview of The Job Console and Queue Servers

A **Queue Server** reads requests from the queue and fires the job in its available slots. As jobs complete, slots become available for more jobs. When a job completes, results are loaded into back into the Job form and is marked as complete.

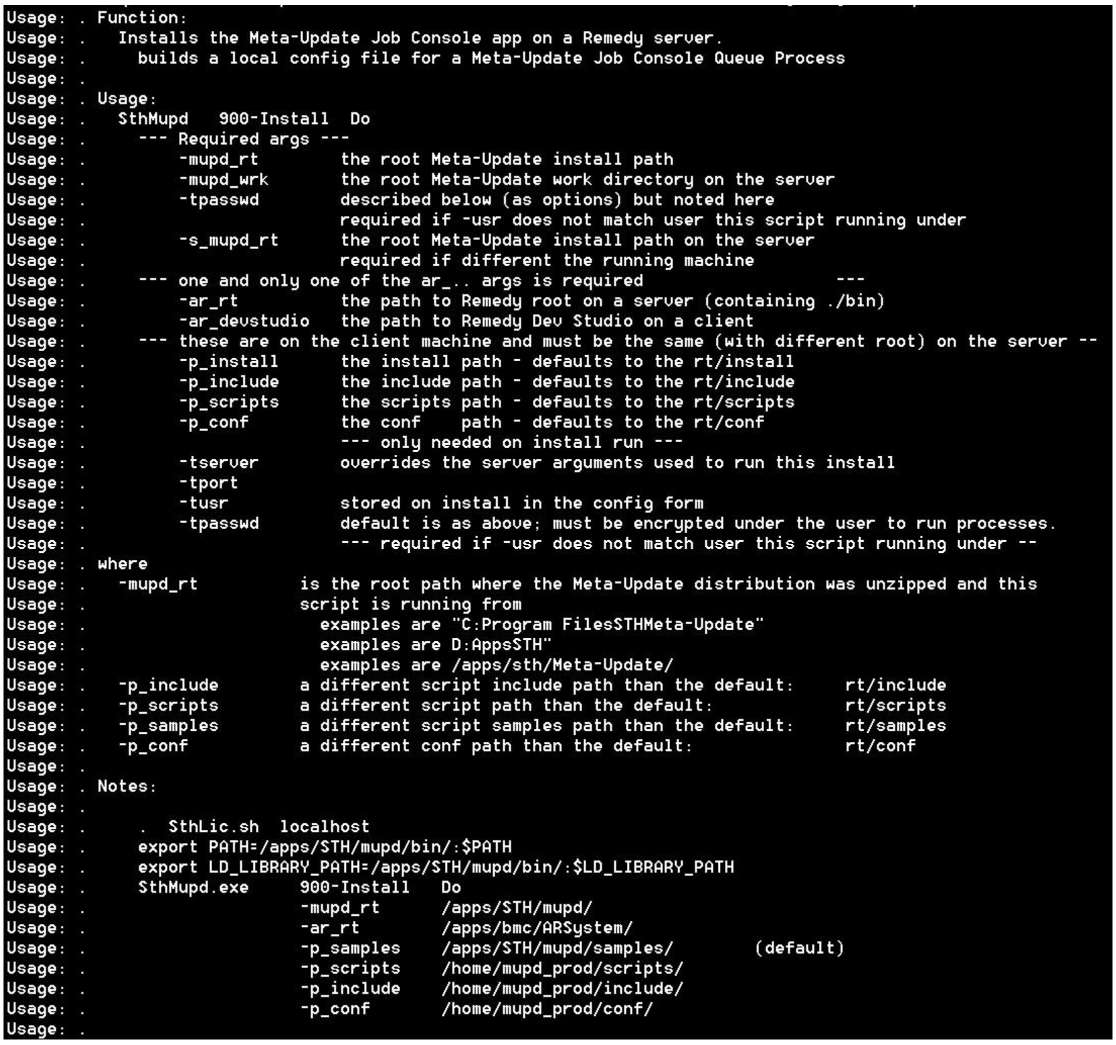
Running the Install script

To install the Job Console application, run the Meta-Update Install script,

    scripts/800-Install.ini

The script may be run on a workstation or server, on Windows or Linux.

When run with no arguments, the following usage appears:



```
Usage: . Function:
Usage: .   Installs the Meta-Update Job Console app on a Remedy server.
Usage: .     builds a local config file for a Meta-Update Job Console Queue Process
Usage: .
Usage: . Usage:
Usage: .   SthMupd   900-Install  Do
Usage: .     --- Required args ---
Usage: .         -mupd_rt        the root Meta-Update install path
Usage: .         -mupd_wrk       the root Meta-Update work directory on the server
Usage: .         -tpasswd        described below (as options) but noted here
Usage: .                         required if -usr does not match user this script running under
Usage: .         -s_mupd_rt      the root Meta-Update install path on the server
Usage: .                         required if different the running machine
Usage: .     --- one and only one of the ar_.. args is required                    ---
Usage: .         -ar_rt          the path to Remedy root on a server (containing ./bin)
Usage: .         -ar_devstudio   the path to Remedy Dev Studio on a client
Usage: .     --- these are on the client machine and must be the same (with different root) on the server --
Usage: .         -p_install      the install path - defaults to the rt/install
Usage: .         -p_include      the include path - defaults to the rt/include
Usage: .         -p_scripts      the scripts path - defaults to the rt/scripts
Usage: .         -p_conf         the conf    path - defaults to the rt/conf
Usage: .                         --- only needed on install run ---
Usage: .         -tserver        overrides the server arguments used to run this install
Usage: .         -tport
Usage: .         -tusr           stored on install in the config form
Usage: .         -tpasswd        default is as above; must be encrypted under the user to run processes.
Usage: .                         --- required if -usr does not match user this script running under --
Usage: . where
Usage: .   -mupd_rt          is the root path where the Meta-Update distribution was unzipped and this
Usage: .                     script is running from
Usage: .                       examples are "C:Program FilesSTHMeta-Update"
Usage: .                       examples are D:AppsSTH"
Usage: .                       examples are /apps/sth/Meta-Update/
Usage: .   -p_include        a different script include path than the default:     rt/include
Usage: .   -p_scripts        a different script path than the default:             rt/scripts
Usage: .   -p_samples        a different script samples path than the default:     rt/samples
Usage: .   -p_conf           a different conf path than the default:               rt/conf
Usage: .
Usage: . Notes:
Usage: .
Usage: .     .  SthLic.sh  localhost
Usage: .     export PATH=/apps/STH/mupd/bin/:$PATH
Usage: .     export LD_LIBRARY_PATH=/apps/STH/mupd/bin/:$LD_LIBRARY_PATH
Usage: .     SthMupd.exe     900-Install   Do
Usage: .                     -mupd_rt      /apps/STH/mupd/
Usage: .                     -ar_rt        /apps/bmc/ARSystem/
Usage: .                     -p_samples    /apps/STH/mupd/samples/       (default)
Usage: .                     -p_scripts    /home/mupd_prod/scripts/
Usage: .                     -p_include    /home/mupd_prod/include/
Usage: .                     -p_conf       /home/mupd_prod/conf/
Usage: .
```

Figure 18 Install Script Usage

The install script can be run on either the server or a workstation that can connect to the server.

The script does the following:

> Check if the application is installed
> If not, it runs a def file import through the DevStudio API or RiK depending on the server release and whether install is being run on the server
> Abort if the application failed to install
> Add a single configuration record
> Add the sample scripts template data

To import the application, the script uses either the RIK binary from the Remedy Server, or the DevStudio Import API Java class from either a workstation with the BMC DevStudio client installed or as supplied with the Server for releases 9.1.05 / 1805 and beyond.

The import is run synchronously. The script will wait for the completion of the import

As it is a BMC “requirement”, using the DevStudio import API requires that the current working directory be that of the DevStudio installation. For that reason, a batch file is created to do the import.

For RIK installations, the batch file is not needed and RIK is called directly. However, the files that RIK uses to load need to be placed in the current directory.

Note that for DevStudio **`java`** must be on the PATH.

Arguments

`-mupd_rt`	Required. This is the root of the Meta-Update install on the machine running the install script. Examples: **`/apps/sth/mupd_600/`** **`D:\apps\sth\mupd_6.00\`** It is where the distribution package was unzipped.
`-s_mupd_rt`	Required if not running on the server. Else cannot be used. Use only if the machine running the install is not the server. Points to the root path on the server. Used when adding data only.

`-ar_rt`	Use only if the install script is running on the server. It is the root path of the BMC Remedy installation. Examples:
	`/apps/bmc/ARSystem/`
	`D:\apps\BMC Software\ARSystem\`

`-ar_devstudio`	Required if not running on the server. Else cannot be used. Use only if the install script is not running on the server. This machine must have the BMC DevStudio client installed. This is the root path of the BMC Remedy DevStudio installation. Example: **`d:\Apps\BMC\ARSystem\DeveloperStudio\`**
`-p_include`	Optional. Use only to override the default of the Meta-Update root path with **`./include/`** as a subdirectory.
`-p_scripts`	Use only to override the default of the Meta-Update root path with **`./scripts/`** as a subdirectory.

`-p_samples`	Use only to override the default of the Meta-Update root path with **`./samples/`** as a subdirectory.
`-upd`	Optional. Set 1 to upgrade the application and data for samples. Use only when installing an upgrade downloaded from Software Tool House. **`-upd` implies both `-updAdd`** and **`-updData`**
`-updApp`	Optional. Set 1 to upgrade the application. Use only when installing an upgrade downloaded from Software Tool House.
`-updData`	Optional. Set 1 to upgrade the sample scripts. Use only when installing an upgrade downloaded from Software Tool House.

If run on the server, use the **`-mupd_rt`** and **`-ar_rt`** arguments and not **`-ar_devstudio`** or **`-s_mupd_rt`**. If the server is above release 9.1.05/1805, DevStudio is used with the **`-ar_rt`** argument.

If ***not*** run on the server, use the **`-mupd_rt`**, **`-s_mupd_rt`** and **`-ar_devstudio`** arguments and not **`-ar_rt`**.

Job Queue Server

The Job Queue server is simply a Meta-Update script.

This topic covers

- Deciding where the Job server should run.
- When to use a local Job Server config file
- Deciding on the number of slots
- Setting up the environment
- Starting the Job Queue Server

Where to run Job Queue Server

The Job Queue Service is a simple Meta-Update script.

As such, any environment set up for running Meta-Update will work for the Job Queue server.

Well perhaps not so simple. But as a script, you can inspect it, change it, use the debugger on it, watch it in action.

The Job Queue server is a long lived Meta-Update process. It should basically be started when its host is started and ended when its host is brought down.

A Job Queue server runs Meta-Update jobs in “slots”. A Queue server with four jobs running in four slots can use

Considerations of where to run a Job Queue server.

- IP access to the Remedy server whose Queue is being servered
- IP access to any other servers that the script or Templates may offer
- Performance needs to be evaludated:
    - if run on the server, may affect Remedy performance while taking resources away from Remedy, and may limit slots
    - if run on a separate machine, much less affect on Remedy performace

- May be run on a workstation for testing

Evaluating Performance

Evaluating Job Queue Server Performance

Simply ensure no queue server is running. Then initiate the following six jobs

1. arschema report
2. SRD Report
3. UDM Jobs Spreadsheet
4. server workflow report
5. server fields
6. UDM Transforms Spreadsheet
7. server forms report

Now, initiate a Queue server with say, 4, slots. The first fours jobs will fill the slots and three will be left queued.

Check the CPU and Memory of the machine running the Queue server.
Check the CPU and Memory of the Remedy Server

Jobs 2 and 3 should complete quickly and jobs 5 and 6 should initiate.
Job 6 is also quick and job 7 should initiate.

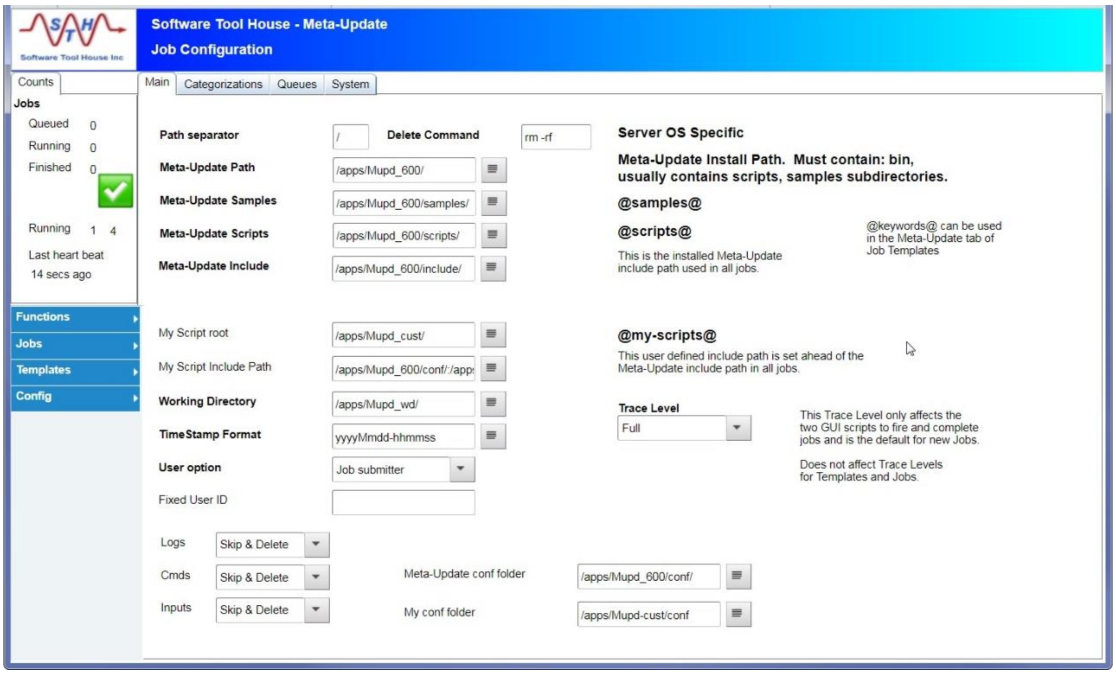
Make the time that all jobs ere done. This can also be done by reporting on the Jobs themselves.

In all, this test should take no more than 15 minutes.

Queue Server Config

The Job Queue server can be started on a server or workstation. The server holds a single record in the STH:JobCfg form.

When the Job Queue server is started on the server, it can use the configuration held in the STH:JobCfg form.



Software Tool House - Meta-Update
Job Configuration

Counts

Jobs

Queued	0
Running	0
Finished	0
Running	1 4

Last heart beat
14 secs ago

Functions
Jobs
Templates
Config

Main | Categorizations | Queues | System

Path separator	/	**Delete Command**	rm -rf
Meta-Update Path	/apps/Mupd_600/		
Meta-Update Samples	/apps/Mupd_600/samples/		
Meta-Update Scripts	/apps/Mupd_600/scripts/		
Meta-Update Include	/apps/Mupd_600/include/		
My Script root	/apps/Mupd_cust/		
My Script Include Path	/apps/Mupd_600/conf/:/app:		
Working Directory	/apps/Mupd_wd/		
TimeStamp Format	yyyyMmdd-hhmmss		
User option	Job submitter		
Fixed User ID			
Logs	Skip & Delete		
Cmds	Skip & Delete	Meta-Update conf folder	/apps/Mupd_600/conf/
Inputs	Skip & Delete	My conf folder	/apps/Mupd-cust/conf

Server OS Specific

Meta-Update Install Path. Must contain: bin, usually contains scripts, samples subdirectories.

@samples@

@scripts@

This is the installed Meta-Update include path used in all jobs.

@keywords@ can be used in the Meta-Update tab of Job Templates

@my-scripts@

This user defined include path is set ahead of the Meta-Update include path in all jobs.

Trace Level: Full

This Trace Level only affects the two GUI scripts to fire and complete jobs and is the default for new Jobs.

Does not affect Trace Levels for Templates and Jobs.

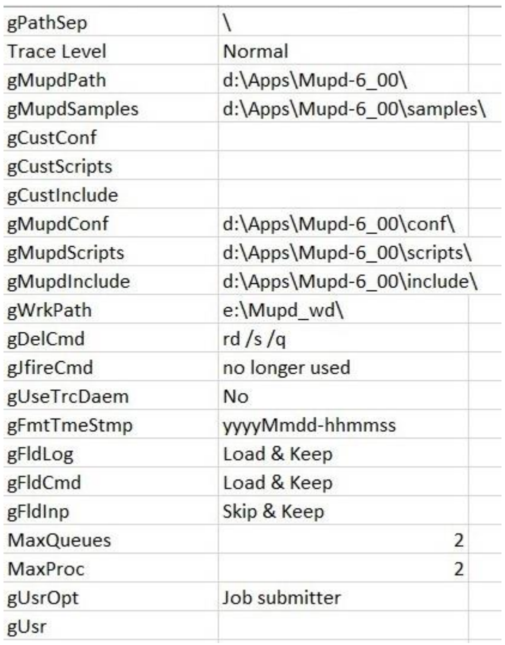
Figure 19 Job Configuration

A local configuration file is used instead of the above record when the Job Queue server is run in an environment with different paths than on the server.

Job Templates use keywords such as `@my-conf@` which are translated to values in either the server or local configuration.

Paths may use forward or backslashes no matter what platform the Queue server is running on.

Inner directories, for example `100-Misc` in the samples, must be the same on the server and on the Queue server.



gPathSep	\
Trace Level	Normal
gMupdPath	d:\Apps\Mupd-6_00\
gMupdSamples	d:\Apps\Mupd-6_00\samples\
gCustConf	
gCustScripts	
gCustInclude	
gMupdConf	d:\Apps\Mupd-6_00\conf\
gMupdScripts	d:\Apps\Mupd-6_00\scripts\
gMupdInclude	d:\Apps\Mupd-6_00\include\
gWrkPath	e:\Mupd_wd\
gDelCmd	rd /s /q
gJfireCmd	no longer used
gUseTrcDaem	No
gFmtTmeStmp	yyyyMmdd-hhmmss
gFldLog	Load & Keep
gFldCmd	Load & Keep
gFldInp	Skip & Keep
MaxQueues	2
MaxProc	2
gUsrOpt	Job submitter
gUsr	

Figure 20 Local Job Config CSV file

Job Template Keywords

The following table lists keywords that reference these caonfigurations and are available to templates.

`@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

My Script Include Path

If this is used, every job has this path included in its script path. Templates default to their script path being placed ahead of all other paths.

Starting The Job Queue Server

The Job Queue server is a Meta-Update script.

As such, the same environment used to run scripts can be used to run the Job Queue Server.

```
. Function:
.   Monitors a single STH:JobQueue firing and controlling a number of
.     waiting jobs.
.
. A Job Queue server can be run on a Remedy Server or any client
    machine.  On the server, configuration is controlled by an
    STH:JobCfg record.  On a client machine, configuration is
    controlled by the -cfg specified local file as follows:
.       Key            Val
.       gPathSep
.       gMupdPath
.       gMupdSamples
.       gMupdInclude
.       gWrkPath
.       gDelCmd
.       gFleInp
.       gInclude
  It is separated by a comma and does not have field headers.
.
. Usage:
.   SthMupd   800-JobQueue  Do
                -jobsmax      2
.               -stop         mm-dd-yyy hh:mm:ss
.               -cfg          required if process host != server host
.               -jdbg         1 increases tracing levels of the
JobFire and JobDone scripts
.
. where
.   -jobsmax      max num of simultaenous jobs on this server
.   -stop         A timestamp at which all jobs will quiesce -
                    each on the next root request
.                 Mm/dd/yyyy [hh:mm:ss]
.   -cfg          a config file for the localhost
.   -jdbg         0 or 1; 0 Normal tracing   (default)
.                         1 mass tracing for JobFire, JobDone jobs
.
```

Only **`-jobsmax`** is required.

Distribution Contents

Directory Structure

The Meta-Update distribution is a single zip file that has a single directory at the root.

Linux distributions are gzipped tar balls. They are named SthMupd-6.00-918_lx64.tar.gz where 6.00 is the Meta-Update release and 918 is the Remedy API library version.

The distribution is contained in the single directory is called **`Mupd_X_xx`** where **`Mupd_X_xx`** is the Meta-Update release. This allows you to test different versions and APIs of Meta-Update easily.

This directory contains directories and files that is the Meta-Update distribution.

Directory	Contents
`bin`	Contains all required 64-bit binaries (.exe) and libraries (.dll) for Meta-Update, the Job Console, and bundled utilities. It is recommended that this path be added to the system Environment Variables for use by a Server, and in the User's Environment Variables for workstations.
`docs`	This directory contains the Meta-Update and the Job Console User manuals and the Trace Facility Administration Guide. These are distributed as PDF files.
`install`	Contains scripts, data, ARS def files used a first time to install the GUI application and data and thereafter to create a local configuration file for use with an off-server Queue Process.
`include`	These are included script fragments used by the Meta-Update Job Console, samples, and script packages...
`scripts`	These are scripts used by the Meta-Update Job Console.
`samples`	Contains sample Meta-Update scripts and a sample Trace configuration file.
`conf`	Contains sample Meta-Update configuration files.

Meta-Update supports the 64-bit architecture. All binaries, dlls, shared objects, and executables in the "**`bin`**" only run on 64 bit machines.

```
"C:\Program Files\SoftwareToolHouse\Mupd_6_00\bin\"
/apps/sth/mupd_5_88/bin/
```

This "**`bin`**" directory contains all required binaries (**`.exe`**) and libraries (**`.dll`** / **`.so`**) for Meta-Update and associated utilities.

It is recommended that this path be added to the system Environment Variables for use by a Server, and in the User's Environment Variables for workstations.

All Meta-Update binaries print usage instructions when entered with no arguments.

They may be unzipped and tar extracted into the directory of your choice.

bin

Meta-Update supports the 64-bit architecture. All binaries, dlls, shared objects, and executables in the "**`bin`**" only run on 64-bit machines.

The Meta-Update distribution contains the following binaries for 64-bit architectures:

File	Contents
SthMupd.exe	Meta-Update. - using local trace. This single executable is Meta-Update. It appends to a single log file – **`SthMupd.log`** – on each run.
SthMupdTrc.exe	Meta-Update. - using global trace. This single executable is also Meta-Update with a communication based for logging facility. It will either behave as **`SthMupd.exe`**, or, if the **`SthTrcDaem.exe`** process is running (see below), will send its logs and messages to that process. **`SthTrcDaem.exe`** can be configured and controlled. Note that Meta-Query, Meta-Delete, Meta-Schema are also offered in this trace facility by suffixing "Trc" to the binary names.
SthMqry.exe	Meta-Query This tool allows you to issue ARS and SQL queries through the ARS API and print or create CSVs from the results. Please See the Meta-Query User's Guide for more information.
SthMsch.exe	Meta-Schema This tool allows you to find information about the ARS forms and fields on your server and print or create CSVs from the results. Please See the Meta-Schema User's Guide for more information.
SthMdel.exe	Meta-Delete This tool allows you to delete records from ARS tables on your server. Please See the Meta-Schema User's Guide for more information.
SthLicUpd.exe	License Updater and Password Encryption tool This utility is used to generate an **`SthLic.cmd`** or **`SthLic.sh`** file that sets environment variables for ARS server, authentication and Meta-Update licensing. It is also used to encrypt ARS authentication passwords.

SthTrcDaem.exe	The Trace facility daemon. This binary can be started when the machine powers on and left to run until the powers off. See the Software Tool House, Trace Facility Administration Guide for more information.
SthTrcCtl.exe	The Trace daemon control program. This program is used to control the trace levels, files, sizes, and so on. It communicates with the **`SthTrcDaem.exe`** process.
SthTrcEcho.exe	The Tracy utility that can be used to write messages to the trace facility.
SthArsTime.exe	Allows easy conversion of Remedy time stamp values.
SthSiniGet.exe	Allows extracts of script files to be used in batch files and performs a simple validity test on scripts.

docs

Contains all reference manuals for Meta-Update as PDF files. Note that actual file names have underscores in place of spaces.

File	Contents
Meta-Update Installation Guide.pdf	Meta-Update and the Job Console installation guide.
Meta-Update Users Guide.pdf	This is a detailed reference on Meta-Update scripting. It is used by script developers. It covers developing and debugging scripts.
Meta-Update Job Console Users Guide.pdf	This is a detailed reference on developing templates and firing jobs using the Job Console.
Meta-Update Release Notes.pdf	This highlights changes made in this release of Meta-Update.
Trace Daemon Users Guide.pdf	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.pdf	This highlights changes made in this release of Meta-Update.

include

Contains all Meta-Update “include” scripts used by some of the samples, the Job Console application, and available for your scripts.

File	Contents
000-Jctl-Sync.ini	This is used to allow long running scripts to be terminated.
100-JobProgress-Upd.ini	This is used by samples to update Job Progress in the Job Console (if fired through the Job Console).
300-TmeCalc.ini	Allows you to do several time calculations including building a text string for elapsed time. Used by Job Console.
991-JobCfg.ini	These are used by the Job Console scripts. Placed here as it common to several scripts.
992-asgENV.ini	
993-asgCfg.ini	Contains actual script configurations.

install

Contains all Meta-Update “install” data.

File	Contents
mupd-job-console.def	This is the Job Console application.
data/*	This is used by the Job Console application to add all samples.

samples

Contains all Meta-Update sample scripts.

File	Contents
mupd-job-console.def	This is the Job Console application.

scripts

Contains all Meta-Update scripts used by the Job Console.

File	Contents
100-JobFire.ini	Fires a Job and waits for its completion.
200-JobDone.ini	Once a Job completes, sets values in the Job record and attaches all output files.
800-JobQueue.ini	Runs a Job Queue server launching and controlling jobs in up to 10 “threads”.
900-Install.ini	Installs or upgrades the Job Console application, and is used to create local config files for Job Queue servers

Running Meta-Update

Software Tool House Inc

Index

D

Developing

Scripts 44

Directory

Install root structure 26

Distribution

Contents 53, 54
Deciding the Install Locations 24
Expanding 25

E

Environment

Run Time 31
Run Time, Running 39
Running Meta-Update 38

Environment Variables 32

SthMupdLic 34
SthScriptPath 33

F

Fixed

Meta-Update User 30

I

Installing

About 23
Deciding the Install Locations 24
Directory Structure 26
Distribution Contents 53, 54
Expanding the Distribution 25

L

Licensing 15

O

Output

Program Output 41

P

Program

Meta-Update Output 41

R

Run Time Environment 31

Running 39

Running 38

Environment for Meta-Update 38
Firing from Workflow 43
Meta-Update Environment Variables . 32, 33, 34
Program Output 41

S

Scripts

Developing 44
SthMupdLic Environment Variable 34
SthScriptPath Environment Variable 33

SthMupdLic Environment Variable 34

SthScriptPath Environment Variable 33

U

User

Fixed Meta-Update User 30

W

Workflow

Running Meta-Update from 43