

# Cron Service User's Guide

Version 3.4.0

Last updated: May 26, 2011

---

## Table of Contents

### [Overview](#)

### [Installation and Configuration](#)

### [Request Syntax](#)

- [REGISTER](#)
- [UNREGISTER](#)
- [LIST](#)
- [TRIGGER](#)
- [ENABLE](#)
- [DISABLE](#)
- [VERSION](#)
- [HELP](#)

### [Persistent Registration Data](#)

### [Service Logging](#)

### [Using Python](#)

### [Cron Python Variables](#)

### [Cron User Interface \(CronUI\)](#)

- [Using the CronUI](#)
- [Submit a New Registration](#)
- [Registration Example using Python and Prepare Script](#)
- [Edit a Registration](#)
- [Copy a Registration](#)
- [Trigger a Registration](#)
- [View the Service Log](#)
- [Delete the Service Log](#)

### [Cron Error Codes](#)

### [Appendix A: References](#)

### [Appendix B: Jython and CPython Differences](#)

### [Appendix C: Licenses and Acknowledgements](#)

---

## Overview

The Cron service allows you to register STAF commands that will be executed at a specified time interval(s). The Cron service also provides a User Interface to simplify interaction with the Cron service

Note that Cron registration information is persistent data. This means that if you register with the Cron Service, if you shutdown STAF and restart it (even if you reboot the machine), the prior registration information will still be active. When STAF starts, it reads in the previous Cron registration information, and will execute the registered STAF commands at the specified time interval(s).

Note that the Cron service uses its machine's Operating System date/time information to determine the current date/time in relation to the registered requests. Users of the Cron service must ensure that the machine on which the Cron service is running has the correct Operating System date/time (and that it correctly updates the current date/time relative to Daylight Savings Time).

---

## Installation and Configuration

1. Install Java 1.5 (aka 5.0) or later.
2. Install STAF Version 3.3.3 or later by following the installation instructions in the STAF documentation.
3. Install the Cron service by downloading the CronV340.zip/tar file from [Get STAF Services](#) into a local directory (e.g. C:/STAF/services or /usr/local/staf/services) and extracting it.
4. Configure the Cron service by adding the following statement to your STAF configuration file:

```
SERVICE <Name> LIBRARY JSTAF EXECUTE <Cron Jar File Name>  
    [OPTION <Name[=Value]>]...  
    [PARMS OLDVARRESOLUTION]
```

where:

- SERVICE specifies the name by which the Cron service will be known on this machine.
- EXECUTE specifies the fully qualified name of the STAFcron.jar file. On Windows systems, this might be C:/STAF/services/cron/STAFcron.jar. On Unix systems, this might be /usr/local/staf/services/cron/STAFcron.jar. This option will resolve STAF variables, so you could use the STAF/Config/STAFRoot variable to specify the jar file name in a platform independent manner such as {STAF/Config/STAFRoot}/services/cron/STAFcron.jar.
- OPTION specifies a configuration option that will be passed on to the JSTAF Java service proxy library. This is typically used by service proxy libraries to further control the interface to the actual service implementation. You may specify multiple OPTIONS for a given service. See the STAF User's Guide for more information on options for the JSTAF Java service proxy library.
- OLDVARRESOLUTION specifies to perform the old way of variable resolution (e.g. the way variables resolution was done prior to Cron V3.3.1). This means that on a REGISTER request, options MACHINE, SERVICE, and REQUEST will resolve STAF variables when the REGISTER request is submitted (which means the variables are resolved on the Cron service machine). Note that in Cron V3.3.1, these options were changed to not resolve STAF variables at the time when a REGISTER request is submitted so that STAF variables will be resolved when the STAF service request is submitted.

**Examples:**

```
SERVICE Cron LIBRARY JSTAF EXECUTE {STAF/Config/STAFRoot}/services/cron/STAFcron.jar
```

```
SERVICE Cron LIBRARY JSTAF EXECUTE {STAF/Config/STAFRoot}/services/cron/STAFcron.jar \
    OPTION JVMName=Cron OPTION JVM=C:\j2sdk1.5.0\bin\java
```

```
SERVICE Cron LIBRARY JSTAF EXECUTE {STAF/Config/STAFRoot}/services/cron/STAFcron.jar
PARMS OLDVARRESOLUTION
```

## Request Syntax

The CRON service provides the following requests:

- REGISTER - Registers a STAF command to be executed at a specified time interval
- UNREGISTER - Unregisters a STAF command
- LIST - Lists registered STAF commands
- TRIGGER - Triggers a registered STAF command to be executed immediately (without matching the registered time triggers)
- ENABLE - Enables a STAF command.
- DISABLE - Disables a STAF command.
- HELP - Displays a list of requests for the Cron service and how to use them.

## REGISTER

REGISTER registers a STAF command to be executed at a specified time interval.

### Syntax

```
REGISTER MACHINE <machine> | PYTHONMACHINE <machine>
SERVICE <service> | PYTHONSERVICE <service>
REQUEST <request> | PYTHONREQUEST <request>
[DESCRIPTION <description>]
[PREPARE <script>]
[MINUTE <minute>] [HOUR <hour>]
[DAY <day>] [MONTH <month>]
[WEEKDAY <weekday>]
[ONCE]
[ENABLED | DISABLED]
```

MACHINE specifies the name of the machine where the command will be executed. This option will not resolve STAF variables (unless the OLDVARRESOLUTION parameter is specified when registering the Cron service). When the specified time interval occurs, the value of MACHINE will not be evaluated as a python string.

PYTHONMACHINE specifies the name of the machine where the command will be executed. This option will not resolve STAF variables (unless the OLDVARRESOLUTION parameter is specified when registering the Cron service). When the specified time interval occurs, the value of PYTHONMACHINE will be evaluated as a python string.

SERVICE specifies the name of the service to be executed. This option will not resolve STAF variables (unless the OLDVARRESOLUTION parameter is specified when registering the Cron service). When the specified time interval occurs, the value of SERVICE will not be evaluated as a python string.

PYTHONSERVICE specifies the name of the service to be executed. This option will not resolve STAF variables. When the specified time interval occurs, the value of PYTHONSERVICE will be evaluated as a python string.

**REQUEST** specifies the request to be executed. This option will not resolve STAF variables (unless the `OLDVARRESOLUTION` parameter is specified when registering the Cron service). When the specified time interval occurs, the value of `REQUEST` will not be evaluated as a python string. This option will handle private data.

**PYTHONREQUEST** specifies the request to be executed. This option will not resolve STAF variables. When the specified time interval occurs, the value of `PYTHONREQUEST` will be evaluated as a python string. This option will handle private data.

**PREPARE** specifies Python code which will be executed when the specified time interval occurs. This code will be executed prior to the `PYTHONMACHINE`, `PYTHONSERVICE`, and `PYTHONREQUEST` options being evaluated as python strings. This option will not resolve STAF variables. If the Python code sets the variable `STAFcronSubmit` to any string other than 'true', then the request will not be submitted.

**DESCRIPTION** specifies a description of the registration. It is for informational purposes only. This option will resolve STAF variables.

**MINUTE** specifies the exact minute(s) that the request executes. The valid values are 0 - 59. You can specify either \* (an asterisk) or ANY, meaning all valid values, or a list of elements separated by commas. An element is either a number or an inclusive range, indicated by two numbers separated by a minus sign (such as 1-15).

**HOUR** specifies the exact hour(s) that the request executes. The valid values are 0 - 23. You can specify either \* (an asterisk) or ANY, meaning all valid values, or a list of elements separated by commas. An element is either a number or an inclusive range, indicated by two numbers separated by a minus sign (such as 8-17).

**DAY** specifies the exact day(s) that the request executes. The valid values are 1 - 31. You can specify either \* (an asterisk) or ANY, meaning all valid values, or a list of elements separated by commas. An element is either a number or an inclusive range, indicated by two numbers separated by a minus sign (such as 1-2).

**MONTH** specifies the exact month(s) that the request executes. The valid values are 1 - 12. You can specify either \* (an asterisk) or ANY, meaning all valid values, or a list of elements separated by commas. An element is either a number or an inclusive range, indicated by two numbers separated by a minus sign (such as 5-6).

**WEEKDAY** specifies the exact weekday(s) that the request executes. The valid values are 0 - 6 (Sunday = 0, ...). You can also specify the textual weekday ("Sunday", etc.). You can specify either \* (an asterisk) or ANY, meaning all valid values, or a list of elements separated by commas. An element is either a number or an inclusive range, indicated by two numbers separated by a minus sign (such as 1-15).

**ONCE** indicates that the STAF command should only be executed one time. After the specified time interval has occurred and the STAF command has been submitted, the ID for this request will be unregistered.

**ENABLED** specifies that the command will be enabled when it is registered. This means that the command will be submitted when the specified time triggers occur. This is the default if the `ENABLED` or `DISABLED` options are not specified.

**DISABLED** specifies that the command will be disabled when it is registered. This means that the command will not be submitted when the specified time triggers occur.

**Note:** At least one of the time interval options (`MINUTE`, `HOUR`, `DAY`, `MONTH`, `WEEKDAY`) must be specified in a `REGISTER` request.

## Security

This request requires at least trust level 5.

## Results

Upon successful return, the result buffer contains the Cron ID.

## Examples

- **Goal:** Execute the command "start command notepad" at 1:00AM every day.

**Syntax:** STAF local CRON REGISTER MACHINE local SERVICE PROCESS REQUEST "START COMMAND notepad" HOUR 1

- **Goal:** Execute the command "start command notepad" at 1:00AM every Sunday.

**Syntax:** STAF local CRON REGISTER MACHINE local SERVICE PROCESS REQUEST "START COMMAND notepad" HOUR 1 WEEKDAY Sunday

- **Goal:** Execute a STAX job every December 1st.

**Syntax:** STAF local CRON REGISTER MACHINE local SERVICE stax REQUEST "EXECUTE FILE c:/tests/startall.xml" MONTH 12 DESCRIPTION "Kick off all of the regression tests"

Note that the registration in this example would only result in the command being executed the next time the month changes to 12. So, if you registered this command on Dec 5, 2003, it would not be executed until 12AM Dec 1 2004.

- **Goal:** Execute a STAX job every day in December.

**Syntax:** STAF local CRON REGISTER MACHINE local SERVICE stax REQUEST "EXECUTE FILE c:/tests/startall.xml" MONTH 12 DAY "\*"

- **Goal:** Execute a STAX job in file C:/automate/updateStatus.xml via the STAX service on machine server1 every week day (Monday through Friday) at the following times: 8:00AM, 12:00PM, 3:00PM, 4:00PM, 5:00PM, and midnight.

**Syntax:** STAF local CRON REGISTER MACHINE server1 SERVICE stax REQUEST "EXECUTE FILE c:/automate/updateStatus.xml" WEEKDAY "Monday-Friday" HOUR "8, 12, 15-18, 0"

- **Goal:** Execute a STAX job in file /tests/TestA.xml' via the STAX service on machine server1 every day at midnight. This STAX job takes the following map of arguments as input: {'machine': 'client1', 'testDir': '/tests'}

#### Syntax:

- This example uses the REQUEST option which will resolves STAF variables. A STAF variable reference is denoted by surrounding the variable in curly braces, e.g. {MyVar}. It is important to understand this because the value being specified for REQUEST contains a "{". So, because of this special significance of "{", if you do not want variable substitution performed, use a caret, "^", as an escape character for "{" and "^". This is talked about in section [8.19.5 RESOLVE](#) in the STAF User's Guide when discussing the VAR service. If you forget to escape the { used to denote the beginning of a map (aka dictionary) in Python, this request will fail with RC 13 (Variable Does Not Exist) and with an error message that says "Variable resolution failed" for the REQUEST value.

```
STAF local CRON REGISTER MACHINE server1 SERVICE STAX REQUEST "EXECUTE FILE /tests/TestA.xml ARGS \"^{ 'machine': 'client1', 'testDir': '/tests' }\" HOUR 0
```

On Windows, we need to also escape the "^" with another "^" if submitted from the Windows command prompt (cmd.exe) because cmd.exe eats one of the carets when enclosed in double quotes:

```
STAF local CRON REGISTER MACHINE server1 SERVICE STAX REQUEST "EXECUTE FILE /tests/TestA.xml ARGS \"^^{ 'machine': 'client1', 'testDir': '/tests' }\" HOUR 0
```

- This example uses the PYTHONREQUEST option which does not resolve STAF variables. It specifies the request using Python syntax. Because the PYTHONREQUEST value is enclosed in double quotes and single quotes, any quotes used within this value need to be escaped with a backslash.

```
STAF local CRON REGISTER MACHINE server1 SERVICE STAX PYTHONREQUEST "'EXECUTE FILE /tests/TestA.xml ARGS \"{'machine': 'client1', 'testDir': '/tests'}\"" HOUR 0
```

- o This example uses the PREPARE option in addition to the PYTHONREQUEST option to help construct the request in Python.

```
STAF local CRON REGISTER MACHINE server1 SERVICE STAX PYTHONREQUEST "'EXECUTE FILE /
tests/TestA.xml ARGS \"%s\"' % (args)" PREPARE "args = {'machine': 'client1',
'testDir': '/tests'}" HOUR 0
```

Actually, it's usually easier to register a more complex request like this that uses Python code via the CronUI instead of the command line because then you don't have to bother so much with escaping quotes. See the [Registration Example using Python and Prepare Script](#) section for an example of how to register this STAF command via the Cron User Interface.

## UNREGISTER

UNREGISTER unregisters a STAF command with the Cron Service.

### Syntax

```
UNREGISTER ID <registrationID>
```

ID specifies the Cron ID which is to be unregistered.

### Security

This request requires at least trust level 4.

### Results

Upon successful return, the result buffer will be empty.

## LIST

LIST lists information about the commands registered with the Cron service.

### Syntax

```
LIST [MACHINE <machine>] [LONG | SHORT]
```

MACHINE specifies the machine for which Cron requests should be listed.

LONG specifies to list more detailed information about each Cron request.

### Security

This request requires at least trust level 2.

### Results

Upon successful return:

- The result buffer for a LIST request will contain a marshalled <List> of <Map:STAF/Service/Cron/CronID> which represents a list of the matching registered Cron requests. The map is defined as follows:

Definition of map class STAF/Service/Cron/CronID
--

<b>Description:</b> This map class represents a registered Cron request.
--

Key Name	Display Name	Type	Format / Value
cronID	ID	<String>	
description	Description (Desc)	<String>   <None>	
machine	Machine	<String>	
service	Service	<String>	
request	Request	<String>	Private data will be masked.
minute	Minute	<String>   <None>	
hour	Hour	<String>   <None>	
dayOfMonth	Day of Month (Day)	<String>   <None>	
month	Month	<String>   <None>	
dayOfWeek	Day of Week (Weekday)	<String>   <None>	
once	Once	<String>	'true'   'false'

- The result buffer for a LIST SHORT request will contain a marshalled <List> of <Map:STAF/Service/Cron/CronIDShort> which represents a list of the matching registered Cron requests with general information. The map is defined as follows:

Definition of map class STAF/Service/Cron/CronIDShort			
<b>Description:</b> This map class represents a registered Cron request with general information.			
Key Name	Display Name	Type	Format / Value
cronID	ID	<String>	
description	Description	<String>   <None>	
machine	Machine	<String>	
service	Service	<String>	
request	Request	<String>	Private data will be masked.

- The result buffer for a LIST LONG request will contain a marshalled <List> of <Map:STAF/Service/Cron/CronIDLong> which represents a list of the matching registered Cron requests with details information. The map is defined as follows:

Definition of map class STAF/Service/Cron/CronIDLong			
<b>Description:</b> This map class represents a registered Cron request with detailed information.			
Key Name	Display Name	Type	Format / Value
cronID	ID	<String>	
description	Description	<String>   <None>	
machine	Machine	<String>	
machineType	Machine Type	<String>	'Literal'   'Python'
service	Service	<String>	
serviceType	Service Type	<String>	'Literal'   'Python'
request	Request	<String>	Private data will be masked.
requestType	Request Type	<String>	'Literal'   'Python'
prepareScript	Prepare Script	<String>   <None>	Private data will be masked.

minute	Minute	<String>   <None>	
hour	Hour	<String>   <None>	
dayOfMonth	Day of Month	<String>   <None>	
month	Month	<String>   <None>	
dayOfWeek	Day of Week	<String>   <None>	
once	Once	<String>	'true'   'false'
enabled	Enabled	<String>	'true'   'false'

## Examples

- **Goal:** List information about all the commands registered with the Cron service.

**Syntax:** STAF local CRON LIST

**Result:** If the request is submitted from the command line, the result, in the verbose format, could look like:

```

ID Desc      Machine Service Request      Minute Hour   Day   Month  Weekday Once
--  -
1  <None> client1 PROCESS START COMMA <None> 1     <None> <None> <None> false
    .austin  ND notepad
    .ibm.co
    m
2  Run the server1 STAX EXECUTE FIL <None> 8,12,1 <None> <None> 1,2,3,4 false
   e STAX     E c:/automa 5,16,1 <None> <None> ,5
   job t     te/updateSt 7,18,0
   o upda    atus.xml
   te sta
   te status
3  Run all server1 STAX EXECUTE FIL <None> 8     <None> <None> <None> false
   l regr
   ession
   tests
4  <None> local PROCESS START COMMA <None> 18    <None> <None> 5     true
    ND notepad

```

- **Goal:** List general information about all the commands registered with the Cron service.

**Syntax:** STAF local CRON LIST SHORT

**Result:** If the request is submitted from the command line, the result, in the verbose format, could look like:

```

ID Description Machine Service Request
--  -
1  <None> client1.austin.ibm.com PROCESS START COMMAND notepad
2  Run the STAX server1 STAX EXECUTE FILE c:/automate/updates
   job to upda tatus.xml
   te status
3  Run all regr server1 STAX EXECUTE FILE c:/tests/runAll.xml
   ession tests
4  <None> local PROCESS START COMMAND notepad

```

- **Goal:** List detailed information about all the commands registered with the Cron service with machine server1.

**Syntax:** STAF local CRON LIST LONG MACHINE server1

**Result:** If the request is submitted from the command line, the result, in the verbose format, could look like:

```
[
  {
    ID           : 2
    Description   : Run the STAX job to update status
    Machine      : server1
    Machine Type : Literal
    Service      : STAX
    Service Type : Literal
    Request      : EXECUTE FILE c:/automate/updateStatus.xml
    Request Type : Literal
    Prepare Script: <None>
    Minute       : <None>
    Hour         : 8,12,15,16,17,18,0
    Day of Month : <None>
    Month        : <None>
    Day of Week  : 1,2,3,4,5
    Once         : false
    Enabled      : true
  }
  {
    ID           : 3
    Description   : Run all regression tests
    Machine      : server1
    Machine Type : Literal
    Service      : STAX
    Service Type : Literal
    Request      : EXECUTE FILE c:/tests/runAll.xml
    Request Type : Literal
    Prepare Script: <None>
    Minute       : <None>
    Hour         : 8
    Day of Month : <None>
    Month        : <None>
    Day of Week  : <None>
    Once         : false
    Enabled      : true
  }
]
```

## TRIGGER

TRIGGER submits a registered STAF command. It is useful for testing the STAF command at any given time (without matching the registered time triggers).

### Syntax

```
TRIGGER ID <registrationID> [SCRIPT ]...
```

ID specifies the Cron ID which is to be triggered.

SCRIPT defines Python code to be executed. You may specify any number of SCRIPT options. They will be executed in the order specified, and will be executed prior to the registration's PREPARE, PYTHONMACHINE, PYTHONSERVICE, AND PYTHONREQUEST values are evaluated.

### Security

This request requires trust level 5.

## Results

If the STAF command was successfully submitted, the return code will be zero. This does not mean that the STAF command was successful, however; it only indicates that the STAF command was successfully submitted. You will need to check the Cron service log to determine the RC and result that the STAF command returned.

Upon successful return:

- The result buffer for a TRIGGER request will contain a marshalled `<Map:STAF/Service/Cron/Trigger>` which represents the information about the submitted STAF command. The map is defined as follows:

Definition of map class STAF/Service/Cron/Trigger			
<b>Description:</b> This map class represents information about the submitted STAF command.			
Key Name	Display Name	Type	Format / Value
machine	Machine	<String>	
requestNumber	Request Number	<String>	

## Examples

- **Goal:** Submit the STAF command for registration ID 6.

**Syntax:** STAF local CRON TRIGGER ID 6

**Result:** If the request is submitted from the command line, the result, in the verbose format, could look like:

```
Machine          : dave2268.austin.ibm.com
Request Number: 1583
```

- **Goal:** Query the Cron log for the previous TRIGGER request.

**Syntax:** STAF local LOG QUERY MACHINE {STAF/Config/MachineNickname} LOGNAME cron CONTAINS "[ID=6] [dave2268.austin.ibm.com:1583]"

**Result:** If the request is submitted from the command line, the result, in the verbose format, could look like:

```
20060809-14:18:52 Info [ID=6] [dave2268.austin.ibm.com:1583] Submitted a STAF
                    command. Triggered manually Submitted STAF command: STA
                    F local var RESOLVE STRING {STAF/Config/MachineNickname}
20060809-14:18:52 Info [ID=6] [dave2268.austin.ibm.com:1583] Completed a STAF
                    command. RC=0, Result=testmachine1
```

- **Goal:** Submit the STAF command for registration ID 12 and set two Python variables to be used when evaluating the STAF command.

**Syntax:**

```
STAF local CRON TRIGGER ID 12 SCRIPT "platform = 'win32'" SCRIPT "version = '3.x'"
```

**Result:** If the request is submitted from the command line, the result, in the verbose format, could look like:

```
Machine          : dave2268.austin.ibm.com
Request Number: 2021
```

## ENABLE

ENABLE enables a STAF command. This means that the command will be submitted when the specified time triggers occur.

### Syntax

```
ENABLE ID <registrationID>
```

ID specifies the Cron ID which is to be enabled.

### Security

This request requires at least trust level 4.

### Results

Upon successful return, the result buffer will be empty. Note that an error will not be returned if the Cron ID is already enabled.

## DISABLE

DISABLE disables a STAF command. This means that the command will not be submitted when the specified time triggers occur.

### Syntax

```
DISABLE ID <registrationID>
```

ID specifies the Cron ID which is to be disabled.

### Security

This request requires at least trust level 4.

### Results

Upon successful return, the result buffer will be empty. Note that an error will not be returned if the Cron ID is already disabled.

## VERSION

VERSION displays the version level of the Cron service or the version level of Jython packaged with the Cron service.

### Syntax

```
VERSION [ JYTHON ]
```

VERSION specifies to display the version level of the Cron service or the version level of Jython packaged with the Cron service.

JYTHON specifies to display the version level of Jython packaged with the Cron service.

### Security

This request requires at least trust level 1.

### Results

The result buffer contains the version level of the Cron service if option JYTHON is not specified. If option JYTHON is specified, the result buffer contains the version level of Jython packaged with the Cron service.

## Examples

- **Goal:** Display the version level of the Cron service.

**Syntax:** VERSION

**Result:** 3.4.0

- **Goal:** Display the version level of Jython packaged with the Cron service.

**Syntax:** VERSION JYTHON

**Result:** 2.5.2-staf-v1

## HELP

HELP displays the request options and how to use them.

### Syntax

HELp

### Security

This request requires at least trust level 1.

### Results

The result buffer contains the Help messages for the request options for the Cron service.

---

## Persistent Registration Data

Note that Cron registration information is persistent data. This means that if you register with the Cron Service, if you shutdown STAF and restart it (even if you reboot the machine), the prior registration information will still be active. When STAF starts, it reads in the previous Cron registration information, and will execute the registered STAF commands at the specified time interval(s).

In Cron V3.2.0, the format of this data was modified. Persistent registration data for versions prior to Cron V3.2.0 will automatically be migrated to the new format. Note that the new format can not be used with versions of Cron prior to V3.2.0.

---

## Service Logging

The Cron service maintains a machine log where it writes information about the STAF commands that it has submitted. It is important to check the log to determine the results of STAF commands submitted by the Cron service. The Cron service will log an entry when the following occurs:

- A REGISTER request is received. This log entry will begin with

```
[ID=<xx>] [<machine>, <handleName>]
```

where <xx> is the ID that was registered, <machine> is the machine that originated the REGISTER request, and <handleName> is the name of the handle that originated the REGISTER request. The level of the log entry will be Info.

- A time trigger matches the options previously specified in a REGISTER request, and the STAF command has been submitted. This log entry will begin with

```
[ID=<xx>] [<machine>:<requestNumber>]
```

where <xx> is the ID for which the STAF command was submitted, <machine> is the machine on which the STAF command was submitted, and <requestNumber> is the request number for the STAF command. The level of the log entry will be Info.

- A time trigger matches the options previously specified in a REGISTER request, but a Python error was encountered when evaluating the registration options. This log entry will begin with

```
[ID=<xx>] Python error
```

where <xx> is the ID for which the Python interpreter encountered an error. The level of the log entry will be Error.

- A time trigger matches the options previously specified in a REGISTER request, but an error was encountered when submitting the STAF command. This log entry will begin with

```
[ID=<xx>] Error submitting a STAF command.
```

where <xx> is the registration ID of the STAF command that was submitted. The level of the log entry will be Error.

- A time trigger matches the options previously specified in a REGISTER request, but the ID is disabled. This log entry will begin with

```
[ID=<xx>] ID is disabled. STAF command not submitted.
```

where <xx> is the ID for which the time trigger matched. The level of the log entry will be Info..

- The service receives notification that a submitted STAF command has completed. This log entry will begin with

```
[ID=<xx>] [<machine>:<requestNumber>]
```

where <xx> is the ID for which the STAF command was executed, <machine> is the machine on which the STAF command was executed, and <requestNumber> is the request number for the STAF command. The log entry will include the RC and Result from the STAF command. If the RC is 0, the level of the log entry will be Pass; otherwise the level of the log entry will be Fail.

- The service receives notification that an asynchronous PROCESS START command has completed. This log entry will begin with

```
[ID=<xx>] [<machine>:<requestNumber>]
```

where <xx> is the ID for which the STAF command was executed, <machine> is the machine on which the STAF command was executed, and <requestNumber> is the request number for the STAF command. The log entry will include the RC and Result from the STAF process. If the RC is 0, the level of the log entry will be Pass; otherwise the level of the log entry will be Fail.

- An UNREGISTER request is received. This log entry will begin with

```
[ID=<xx>] [<machine>, <handleName>]
```

where <xx> is the ID that was unregistered, <machine> is the machine that originated the UNREGISTER request, and <handleName> is the name of the handle that originated the UNREGISTER request. The level of the log entry will be Info.

- An ENABLE request is received. This log entry will begin with

```
[ID=<xx>] [<machine>, <handleName>]
```

where <xx> is the ID that was enabled, <machine> is the machine that originated the ENABLE request, and <handleName> is the name of the handle that originated the ENABLE request. The level of the log entry will be Info.

- A DISABLE request is received. This log entry will begin with

```
[ID=<xx>] [<machine>, <handleName>]
```

where <xx> is the ID that was disabled, <machine> is the machine that originated the DISABLE request, and <handleName> is the name of the handle that originated the DISABLE request. The level of the log entry will be Info.

The [ID=<xx>] and [ID=<xx>] [<machine>:<requestNumber>] tags in the log entries can be useful when querying the log. They can be used with the CONTAINS option of the log service's QUERY request.

The logname for the Cron service is the name under which the service is registered.

Here is an example of what a Cron service log on the local machine could look like (shown via a request from the command line in the table format):

```
C:\>STAF local log QUERY MACHINE {STAF/Config/MachineNickname} LOGNAME cron
Response
-----
Date-Time          Level  Message
-----
20060808-16:08:29  Info  [ID=1] [local://local, STAF/Client] Registered a STAF c
ommand. Register request: REGISTER DESCRIPTION :20:Get
the STAF version MACHINE local SERVICE misc REQUEST ver
sion MINUTE 9 ONCE
20060808-16:08:29  Info  [ID=2] [local://local, STAF/Client] Registered a STAF c
ommand. Register request: REGISTER DESCRIPTION :17:Run
java -version MACHINE local SERVICE process REQUEST :61
:start command java parms -version returnstdout stderrt
ostdout MINUTE 10
20060808-16:08:29  Info  [ID=3] [local://local, STAF/Client] Registered a STAF c
ommand. Register request: REGISTER DESCRIPTION :20:Pyth
on runtime error MACHINE local SERVICE misc REQUEST ver
sion PREPARE a=b MINUTE 11
20060808-16:09:01  Info  [ID=1] [dave2268.austin.ibm.com:2664] Submitted a STAF
command. Triggers: minute[9] Submitted STAF command: ST
AF local misc version
20060808-16:09:01  Info  [ID=1] [local://local, STAF/SERVICE/cron] Unregistered
a STAF command.
20060808-16:09:01  Pass  [ID=1] [dave2268.austin.ibm.com:2664] Completed a STAF
command. RC=0, Result=3.1.4.1
20060808-16:10:01  Info  [ID=2] [dave2268.austin.ibm.com:2681] Submitted a STAF
command. Triggers: minute[10] Submitted STAF command: S
TAF local process start command java parms -version ret
urnstdout stderrtostdout
20060808-16:10:01  Pass  [ID=2] [dave2268.austin.ibm.com:2681] Completed a STAF
command. RC=0, Result=134
20060808-16:10:01  Pass  [ID=2] [dave2268.austin.ibm.com:2681] Process completed
. Process info: {key=, handle=134, rc=0, endTimestamp=2
0060808-16:10:01, fileList=[{rc=0, data=java version "1
.5.0" Java(TM) 2 Runtime Environment, Standard Edition
(build 1.5.0-b64) Java HotSpot(TM) Client VM (build 1
```

```
.5.0-b64, mixed mode)  ]}]
```

```
20060808-16:11:01 Error [ID=3] Python error in the PREPARE value.  PREPARE: a=b
      PyException: Traceback (innermost last):      File "<st
ring>", line 1, in ? NameError: b  Processing registr
ation:  ID: 3  machine: local  service: misc  reques
st: version  prepare: a=b
20060808-16:11:25 Info [ID=2] [local://local, STAF/Client] Unregistered a STAF
command.
20060808-16:11:27 Info [ID=3] [local://local, STAF/Client] Unregistered a STAF
command.
```

Here is an example of only displaying the log records for a particular registration ID:

```
C:\>STAF local LOG QUERY MACHINE {STAF/Config/MachineNickname} LOGNAME cron CONTAINS [ID=2]
Response
-----
```

Date-Time	Level	Message
20060808-16:08:29	Info	[ID=2] [local://local, STAF/Client] Registered a STAF c ommand. Register request: REGISTER DESCRIPTION :17:Run java -version MACHINE local SERVICE process REQUEST :61 :start command java parms -version returnstdout stderrt ostdout MINUTE 10
20060808-16:10:01	Info	[ID=2] [dave2268.austin.ibm.com:2681] Submitted a STAF command. Triggers: minute[10] Submitted STAF command: S TAF local process start command java parms -version ret urnstdout stderrtostdout
20060808-16:10:01	Info	[ID=2] [dave2268.austin.ibm.com:2681] Completed a STAF command. RC=0, Result=134
20060808-16:10:01	Info	[ID=2] [dave2268.austin.ibm.com:2681] Process completed . Process info: {key=, handle=134, rc=0, endTimestamp=2 0060808-16:10:01, fileList=[{rc=0, data=java version "1 .5.0" Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0-b64) Java HotSpot(TM) Client VM (build 1 .5.0-b64, mixed mode)  ]}]
20060808-16:11:25	Info	[ID=2] [local://local, STAF/Client] Unregistered a STAF command.

Here is an example of only displaying the log records for a particular registration ID, machine, and STAF request number:

```
C:\>STAF local LOG QUERY MACHINE {STAF/Config/MachineNickname} LOGNAME cron CONTAINS "[ID=1]
[dave2268.austin.ibm.com:2664]"
Response
-----
```

Date-Time	Level	Message
20060808-16:09:01	Info	[ID=1] [dave2268.austin.ibm.com:2664] Submitted a STAF command. Triggers: minute[9] Submitted STAF command: ST AF local misc version
20060808-16:09:01	Info	[ID=1] [dave2268.austin.ibm.com:2664] Completed a STAF command. RC=0, Result=3.1.4.1

Here is an example of only displaying the error log records, and using the `-verbose` option so that the error output is more readable:

```
C:\>STAF -verbose local LOG QUERY MACHINE {STAF/Config/MachineNickname} LOGNAME cron LEVELMASK
error
Response
-----
```

```
[
  {
    Date-Time: 20060808-16:11:01
    Level      : Error
    Message   : [ID=3] Python error in the PREPARE value.

PREPARE:
a=b

PyException:
Traceback (innermost last):
  File "<string>", line 1, in ?
NameError: b

Processing registration:
  ID: 3
  machine: local
  service: misc
  request: version
  prepare: a=b
  }
]
```

---

## Using Python

The Cron service lets uses the Python scripting language for variable and expression evaluation for the following REGISTER and TRIGGER request options:

- PYTHONMACHINE
- PYTHONSERVICE
- PYTHONREQUEST
- PREPARE
- SCRIPT

This allows the Cron service to take advantage of the powerful and easy-to-use features of Python. The Cron service uses Jython 2.5.2 to execute Python code. Jython is a version of Python written entirely in Java that runs under any compliant Java Virtual Machine (JVM).

Python variable names must follow the Python variable naming conventions. In Python, variable names come into existence when you assign values to them, but there are a few rules to follow when picking names for variables.

- *Syntax: (underscore or letter) + (any number of letters, digits or underscores)*  
Variable names must start with an underscore or letter, and be followed by any number of letters, digits, or underscores. `_machine`, `machine`, and `Mach_1` are legal names, but `1_Mach`, `mach$`, and `@#!` are not.
- *Case matters: MACHNAME is not the same as machname*  
Python always pays attention to case, both in names you create and in reserved words. For instance, `X` and `x` refer to two different variable names.
- *Python reserved words are off limits*  
You cannot define variable names to be the same as words that mean special things in the Python language. Following are reserved words in Python: `and`, `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `exec`, `finally`, `for`, `from`, `global`, `if`, `import`, `in`, `is`, `lambda`, `not`, `or`, `pass`, `print`, `raise`, `return`, `try`, `while`.

**Note:** Python lets you use the names of Python built-in functions as variable names. However, we recommend that you don't

use the name of a Python built-in function as a variable name because you may want to use the Python built-in function at some point when triggering your registration. Following are names of Python built-in functions: abs, basestring, bool, callable, chr, classmethod, cmp, compile, complex, delattr, dict, dir, divmod, enumerate, eval, execfile, file, filter, float, getattr, globals, hasattr, hash, help, hex, id, input, int, isinstance, issubclass, iter, len, locals, long, map, max, min, object, oct, open, ord, pow, property, range, raw\_input, reduce, reload, repr, round, setattr, slice, staticmethod, str, sum, super, tuple, type, unichr, unicode, vars, xrange, zip.

Python string constants can be enclosed in single or double quotes, which allows embedded quotes of the opposite flavor.

For example, the following two lines of Python code do exactly the same thing. They assign a string constant (literal) "CoolTest" to the value of a variable named testName.

```
testName = "CoolTest1"
testName = 'CoolTest1'
```

However, the following line is not the same. It assigns the value of a variable named CoolTest1 to the value of a variable named testName. If this was not what you intended and a variable named CoolTest1 does not exist, a PythonException will be raised and logged in the service log.

```
testName = CoolTest1
```

Say, you wanted to use a Python variable named machName in a STAF service request string. Your Python code could look like:

```
request = 'RELEASE POOL MachinePool ENTRY %s' % (machName)
```

where the %s indicates a String format (and can also be used for decimal format, etc.), and where the value of the machName variable would replace the %s marker.

So, if the value of variable machName is 'client1.company.com', after being evaluated by Python, the request variable's value would be:

```
'RELEASE POOL MachinePool ENTRY client1.company.com'
```

Here are a couple of examples of how you might use Python code in the PREPARE and PYTHONREQUEST options when submitting a REGISTER request to the Cron service:

- This example registers a STAF command that submits a RELEASE request to the ResPool service to release entry client1.company.com from resource pool MachinePool every night at 10:00pm. It sets Python variable **machName** to 'client1.company.com' using Python code in the PREPARE option. In the PYTHONREQUEST option, it uses Python code to construct the request to release the entry specified by Python variable **machName** from the MachinePool.

```
STAF local CRON REGISTER MACHINE server1 SERVICE ResPool PYTHONREQUEST "'RELEASE POOL
MachinePool ENTRY %s' % (machName)" PREPARE "machName = 'client1.company.com'" HOUR 22
```

- This example registers a STAF command that submits a START request to the PROCESS service at midnight every Saturday to run command C:/MyApp/myApp.exe passing it the registration ID of this cron job as a parameter. In the PREPARE option, it imports the com.ibm.staf.STAFUtil Java class so it can use its wrapData() method to "wrap" the value for the COMMAND option since it contains a space. It also constructs the command to run (e.g. 'C:/MyApp/myApp.exe 1' if its registration ID is 1) passing Python variable STAF\_CronID as a parameter to myApp.exe. Note that STAF\_CronID is set by the Cron service to the registration ID of the request being triggered.

```
STAF local CRON REGISTER MACHINE local SERVICE PROCESS PYTHONREQUEST "'START SHELL
COMMAND %s RETURNSTDOUT STDERRTOSTDOUT' % (STAFUtil.wrapData(command))" PREPARE "from com.
ibm.staf import STAFUtil; command = 'C:/MyApp/myApp.exe %s' % (STAF_CronID)" WEEKDAY
Saturday HOUR 0
```

Actually, it's usually easier to register more complex requests like these that use Python code via the Cron User Interface (CronUI) instead of

the command line because then you don't have to bother so much with escaping quotes and you can have multiple lines of Python code. See the [Registration Example using Python and Prepare Script](#) section for an example of how to register a STAF command via the CronUI.

Refer to the "[References](#)" section for where to get more information about Jython and Python.

If you are already a CPython programmer, or are hoping to use CPython code under Jython, refer to the "[Jython and CPython Differences](#)" section for information about differences in the two implementations of Python.

---

## Cron Python Variables

The following variables are set in Python by the Cron service when a registration is triggered. These Python variables can be referenced by Python code in the PREPARE, PYTHONREQUEST, PYTHONMACHINE, and PYTHONSERVICE options for the REGISTER request and in the SCRIPT option for the TRIGGER request.

- **STAFCronID**
  - Description: The ID of the Cron registration being triggered
  - Assigned: When a registration is triggered
  - Type: numeric (PyInteger)
- **STAFCronSubmit**
  - Description: Indicates whether the Cron service should submit the STAF service request when its registration is triggered.
  - Assigned: When a registration is triggered, the Cron service sets it to 'true' by default, but you can override its value if desired by setting it to any other string value in the PREPARE option on a REGISTER request
  - Type: string

---

## Cron User Interface (CronUI)

The Cron service provides a User Interface (CronUI) to simplify interaction with the Cron service. The CronUI is especially useful when registering complex STAF commands with Python code because it helps avoid some issues that you have when registering via the command line by reducing the need for escaping quotes and allows you to easily enter multiple lines of Python code via a text box for the PREPARE option. The CronUI also allows you to modify existing registrations (without having to unregister and the re-register the updated STAF command), and view the Cron service's log.

Note that when viewing the Cron service's log, the Cron User Interface utilizes the STAFLogViewer class to display the logs. For more information on how to use the STAFLogViewer, refer to section "3.11 Class STAFLogViewer" of the STAF Java User's Guide (<http://staf.sourceforge.net/current/STAFJava.htm>).

### Using the CronUI

To use the Cron User Interface, after installing and configuring the Cron Service, from a command prompt enter one of the following commands:

- `java -jar STAFCron.jar`

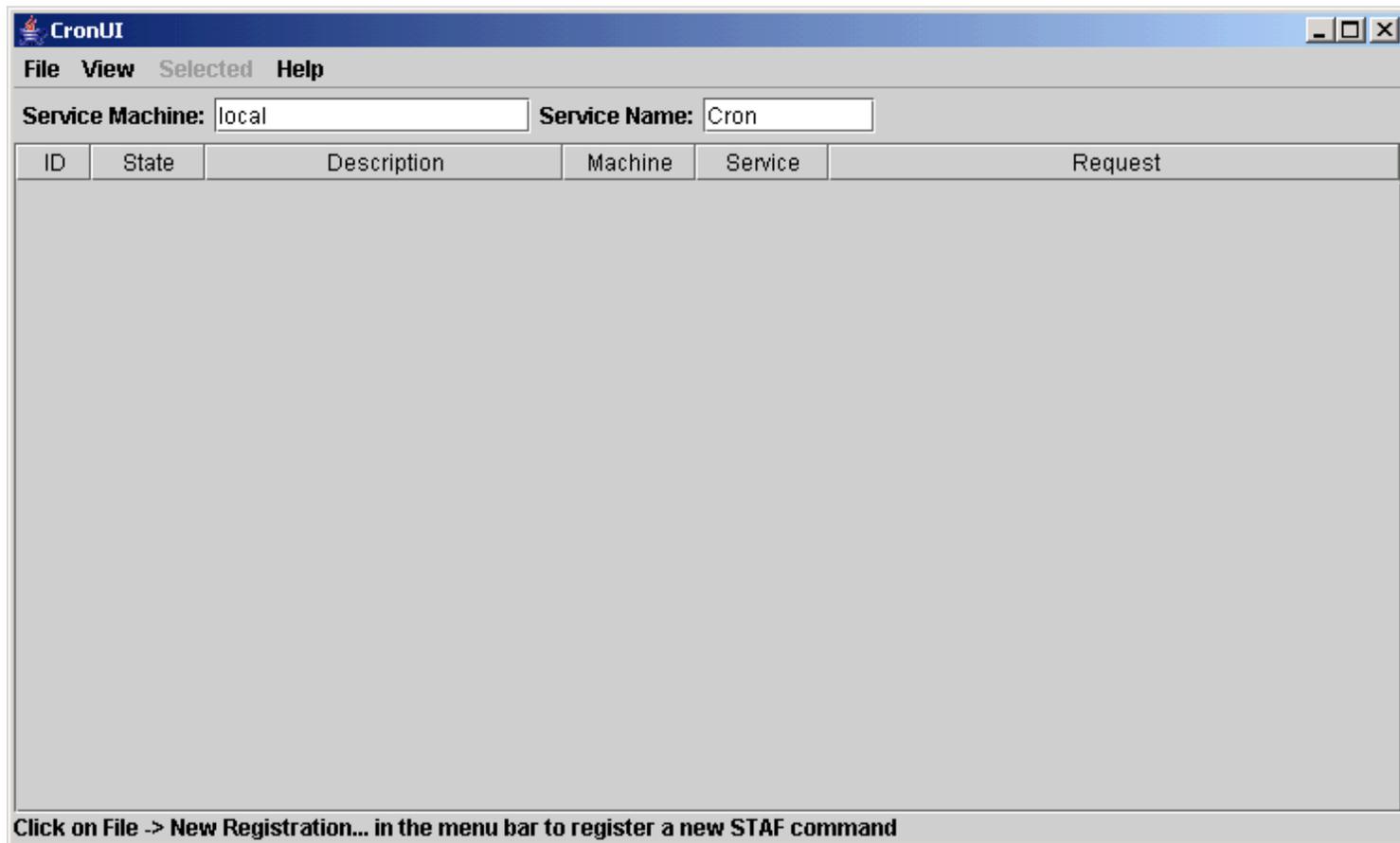
If the STAFCron.jar file is not in the current directory where you are executing this command, then you will need to specify the full path (or the path relative to the current directory) to the STAFCron.jar file.

Note that to run this command, the STAFcron.jar file does not need to be included in your CLASSPATH.

- `java com.ibm.staf.service.cron.CronUI`

Note that to run this command, the STAFcron.jar file must be included in your CLASSPATH.

A User Interface will be displayed which allows you to interact with the Cron service.



The main CronUI window displays a table of all of the currently registered STAF commands. The screen capture above shows what the table would look like if there are no registrations. The window is comprised of the following components (from top to bottom):

1. Menu bar
2. Service configuration options
3. Registration table
4. Status bar

## Menu bar

The File menu bar contains the following menu items:

- **New Registration...** - This opens a new registration dialog where you can input information about a new STAF command you wish to register.
- **Exit** - This exits the CronUI application.

The View menu bar contains the following menu items:

- **Refresh** - This refreshes the registration table by querying the specified Cron service name on the specified machine name.
- **Service Log for ID...** - This allows you to query the service log records for a particular ID. A dialog will be displayed where you can enter the ID. It will generate a QUERY request with the option `CONTAINS [ ID=<xx> ]` where `<xx>` is the ID you specified.

- **Service Log last 100 records** - This allows you to query the last 100 records of the service log.
- **Service Log for submitted STAF commands...** - This allows you to query the service log records for a particular submitted STAF command. A new dialog will open that shows you a list of the submitted STAF commands in the service log. The most recent entries in the service log will be at the top of the list, and the oldest entries in the service log will be at the bottom of the list. The list entries will be in the format [ ID=<xx> ] [ <machine>:<requestNumber> ] <Submitted STAF command> where <xx> is the ID for which the STAF command was submitted, <machine> is the machine where the STAF command was submitted, <requestNumber> is the request number for the submitted STAF command, and <Submitted STAF command> is the actual STAF command that was submitted (this is the STAF command after all Python code has been evaluated). Select the entry in the list for which you wish to view the log records, and a QUERY request will be generated with the option CONTAINS [ ID=<xx> ] [ <machine>:<requestNumber> ] where <xx> is the ID for which the STAF command was submitted, <machine> is the machine where the STAF command was submitted, and <requestNumber> is the request number for the submitted STAF command.
- **Entire Service Log** - This allows you to query the entire service log.
- **Delete Service Log** - This allows you to delete the service log. A confirmation popup will be displayed.

The Selected menu bar will only be enabled if there is a row currently selected in the registration table, and contains the following menu items:

- **Edit** - This allows you edit the data for the selected ID. A new dialog will be displayed, similar to the dialog displayed when you click on "New Registration..." in the File menu bar, except the fields will be filled in with the data for the selected ID.
- **Enable** - This allows you enable the selected ID. This option will only be available if the selected ID is currently disabled.
- **Disable** - This allows you disable the selected ID. This option will only be available if the selected ID is currently enabled.
- **Trigger** - This allows you trigger the STAF command for the selected ID. This is useful when testing your registered STAF commands (where you don't want to wait for any of the time triggers to match). Any Python code in the registration will be evaluated, and the STAF command will be submitted. After the STAF command has been submitted, the service log will be queried with the option CONTAINS [ ID=<xx> ] [ <machine>:<requestNumber> ] where <xx> is the ID for which the STAF command was submitted, <machine> is the machine where the STAF command was submitted, and <requestNumber> is the request number for the submitted STAF command. In the log viewer, you can click on "Refresh" in the View menu bar to refresh the log entries and determine if the STAF command has completed. Note that if the submitted STAF command was registered with the ONCE option, triggering the STAF command will not cause the registration to be unregistered.
- **Trigger with script...** - This is similar to the **Trigger** menu item, except prior to submitting the command, a dialog will be displayed where you can enter any Python code that you want to have executed by the Python interpreter, prior to the evaluation of any PREPARE, PYTHONMACHINE, PYTHONSERVICE, or PYTHONREQUEST options. This is useful if you need to simulate the setting of any variables that might be needed for these options.
- **Copy to new registration** - This allows to you copy the registration data for an existing registration to a new registration. A new dialog will be presented with all of the data from the selected registration. You can change any of the data as needed, and then click on the "Register" button and a new ID will be registered.
- **Unregister** - This allows you to unregister the selected registration. A confirmation popup will be displayed.

Note that all of the "Selected" menu items are also available in a pop-up menu when you right click on a row in the registration table, and they perform the exact same functions as the "Selected" menu bar items.

The Help menu bar contains the following menu items:

- **About** - This displays the version of the Cron service.

## Service configuration options

The "Service Machine" text field allows you to specify the hostname of the machine where the Cron service is running. The default is "local".

The "Service Name" text field allows you to specify the service name for which the Cron service was configured. The default is "Cron".

After changing either of these values, you can press the "Enter" key to refresh the table, or click on "View" in the menu bar, and then select "Refresh".

## Registration table

The registration table displays the current registrations for the Cron service you specified in the service configuration fields (for Service Machine and Service Name). It is refreshed every time you click on "View" in the menu bar and select "Refresh", as well as when you change the Service Machine or Service Name fields and press "Enter".

It is also refreshed whenever you add a new registration, unregister a registration, edit a registration, or copy an existing registration to a new registration.

The registration table has the following columns:

- **ID** - This is the registration ID.
- **Description** - This is the registration's description.
- **Machine** - This is the registration's machine. Note that it may contain Python code.
- **Service** - This is the registration's service. Note that it may contain Python code.
- **Request** - This is the registration's request. Note that it may contain Python code.

## Status bar

The status bar shows informational messages. Here are the possible messages:

- **Double click on an existing registration to edit** - Displayed when there is at least one registration in the table.
- **Click on File -> New Registration... in the menu bar to register a new STAF command** - Displayed when there are currently no registrations in the table.
- **Enter a valid Service Machine and Service Name** - Displayed if the machine entered in the "Service Machine" text field cannot be reached, or if the service name entered in the "Service Name" text field is not a valid service name on the machine.

## Submit a New Registration

To submit a new registration, click on "File" in the menu bar and then select "New Registration...". You will see the following window:

**Cron Registration**

**Cron Register options**

Registration ID: N/A

Description:

Target Machine:  Python

Target Service:  Python

Target Request:  Python

Prepare Script:

Minute:

Hour:

Day:

Month:

Weekday:

Once:

Enabled:

**Register** **Cancel**

This window allows you to specify the data for the registration:

- **Registration ID** - The registration ID. Since this is a new registration, the value is "N/A".
- **Description** - The description of the registration. This field is optional.
- **Target Machine** - The machine where you wish to have the STAF command executed. This field is required. If the text contains Python code, then select "Python" next to the "Target Machine" label.
- **Target Service** - The service for the STAF command you wish to have executed. This field is required. If the text contains Python code, then select "Python" next to the "Target Service" label.
- **Target Request** - The STAF request you wish to have executed. This field is required. If the text contains Python code, then select "Python" next to the "Target Request" label.
- **Prepare Script** - Any Python code that you wish to have evaluated prior to the evaluation of any PYTHONMACHINE, PYTHONSERVICE, or PYTHONREQUEST options. This field is optional.
- **Minute** - Specifies the exact minute(s) that the STAF command executes. The valid values are 0 - 59. You can specify either \* (an asterisk) or ANY, meaning all valid values, or a list of elements separated by commas. An element is either a number or an inclusive range, indicated by two numbers separated by a minus sign (such as 1-15).
- **Hour** - Specifies the exact hour(s) that the STAF command executes. The valid values are 0 - 23. You can specify either \* (an asterisk) or ANY, meaning all valid values, or a list of elements separated by commas. An element is either a number or an inclusive range, indicated by two numbers separated by a minus sign (such as 8-17).
- **Day** - Specifies the exact day(s) that the STAF command executes. The valid values are 1 - 31. You can specify either \* (an asterisk) or ANY, meaning all valid values, or a list of elements separated by commas. An element is either a number or an inclusive range, indicated by two numbers separated by a minus sign (such as 1-2).
- **Month** - Specifies the exact month(s) that the STAF command executes. The valid values are 1 - 12. You can specify either \* (an asterisk) or ANY, meaning all valid values, or a list of elements separated by commas. An element is either a number or an inclusive range, indicated by two numbers separated by a minus sign (such as 5-6).
- **Weekday** - Specifies the exact weekday(s) that the STAF command executes. The valid values are 0 - 6 (Sunday = 0, ....). You can also specify the textual weekday ("Sunday", etc.). You can specify either \* (an asterisk) or ANY, meaning all valid values, or a list of elements separated by commas. An element is either a number or an inclusive range, indicated by two numbers separated by a minus sign (such as 1-5).
- **Once** - Indicates that the STAF command should only be executed one time. After the specified time interval has occurred and the

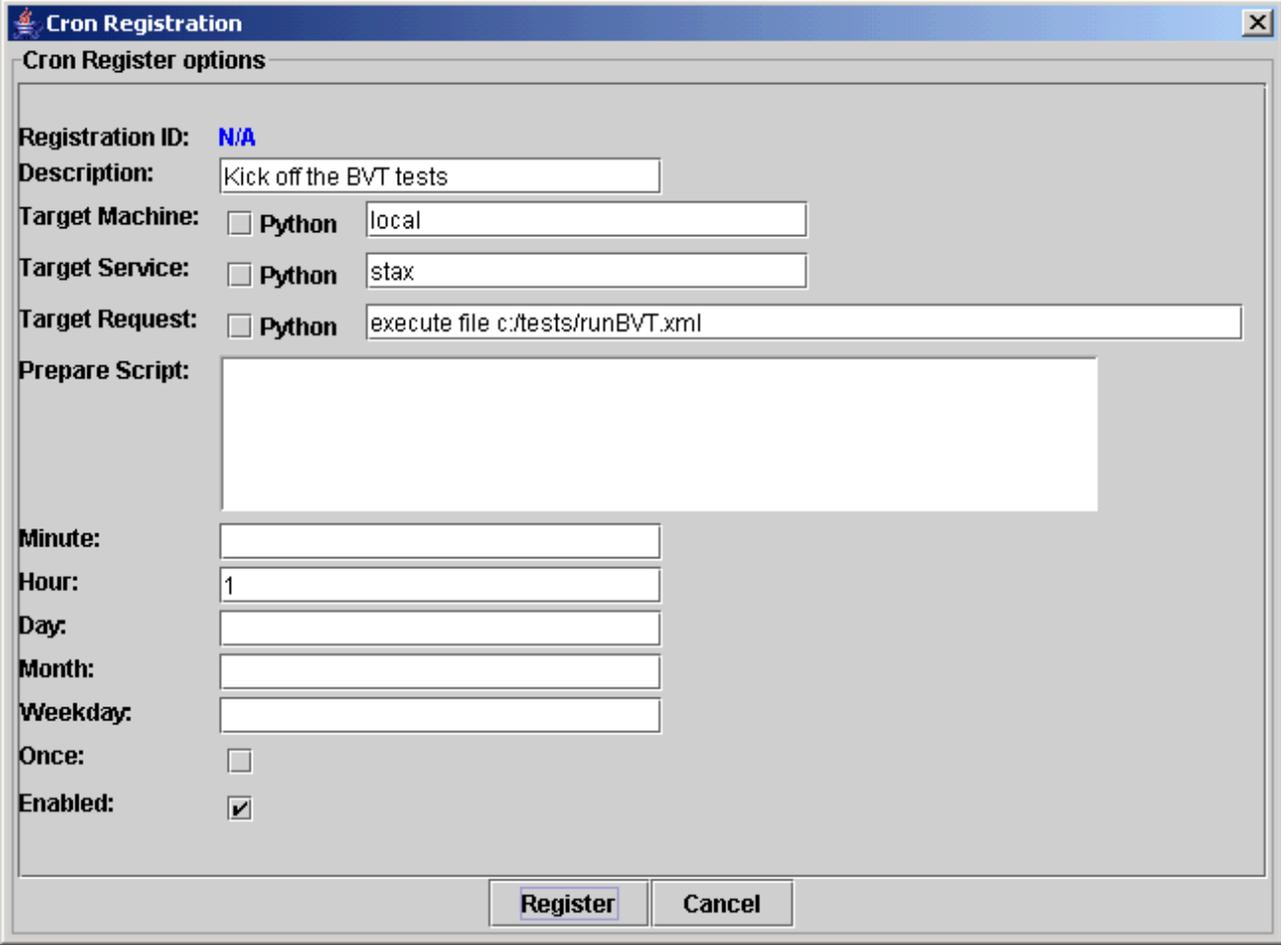
STAF command has been submitted, the ID for this request will be unregistered.

- **Enabled** - Indicates that the STAF command will be enabled when it is registered. If the STAF command is disabled, it will not be submitted when its time triggers occur.

Note that at least one of the time interval options (Minute, Hour, Day, Month, Weekday) must be specified.

To get help on any of these options while using CronUI, mover the cursor over the text field or checkbox and a tool tip will be displayed.

Fill in the appropriate information for the STAF command you wish to have executed. For example, to have a STAX job executed at 1:00AM every day:

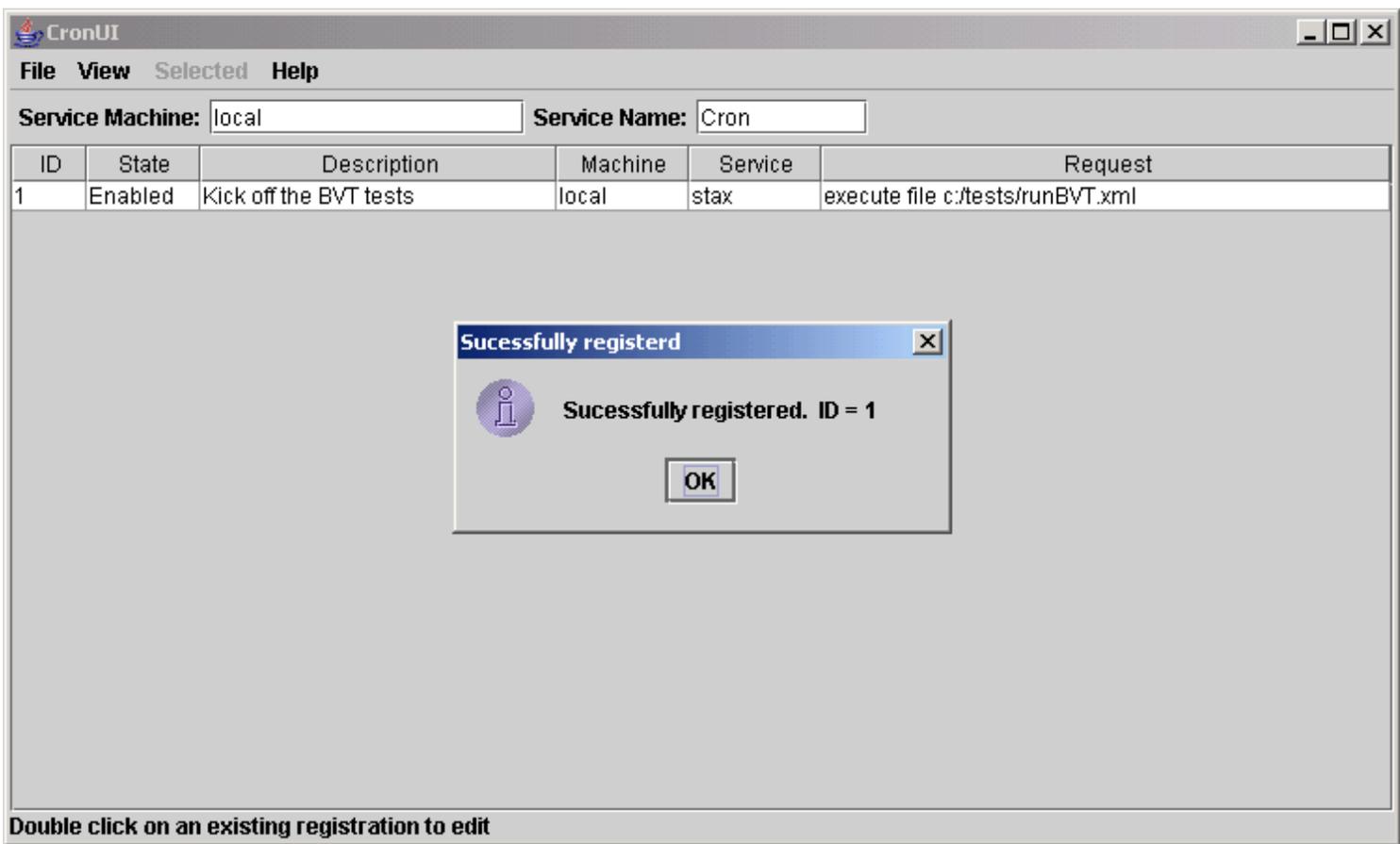


The screenshot shows a dialog box titled "Cron Registration" with a close button (X) in the top right corner. The dialog is titled "Cron Register options" and contains the following fields and controls:

- Registration ID:** N/A
- Description:** Kick off the BVT tests
- Target Machine:**  Python local
- Target Service:**  Python stax
- Target Request:**  Python execute file c:/tests/runBVT.xml
- Prepare Script:** (Empty text area)
- Minute:** (Empty text field)
- Hour:** 1
- Day:** (Empty text field)
- Month:** (Empty text field)
- Weekday:** (Empty text field)
- Once:**
- Enabled:**

At the bottom of the dialog are two buttons: "Register" and "Cancel".

Click on the "Register" button and you will see that the STAF command has been assigned a unique registration ID, and that it is listed in the registration table.



## [Registration Example using Python and Prepare Script](#)

To register a more complex STAF command, you may find it easier to use the CronUI instead of submitting a REGISTER request to the Cron service via the command line. This is especially true for STAF commands that use Python. Also, if you select to specify the request using Python and use the "Prepare Script" field, you usually don't have to escape as many quotes.

This section shows an example of registering a more complex STAF command using the CronUI. It will use the same example that was discussed in the [REGISTER](#) section that registered a STAF command via the command line to execute STAX job /tests/TestA.xml via the STAX service on machine server1 every day at midnight and passed the following argument map to the STAX job: { 'machine' : 'client1', 'testDir': '/tests' }

Remember that a STAF variable reference is denoted by surrounding the variable in curly braces, e.g. {MyVar}. So, since the request contains a "{" which is not being used to denote a STAF variable reference, this example selects "Python" for the target request so that STAF variable resolution will not be performed. If "Python" was not selected for the target request, then the "{" would need to be escaped with a caret, "^", so that no STAF variable resolution would be performed. This example also uses the "Prepare Script" field to construct the STAX EXECUTE request using Python.

The following is an example of how you could register the following STAX EXECUTE command to run every day at midnight:

```
STAF server1 STAX EXECUTE FILE "/tests/TestA.xml" ARGS "{ 'machine': 'client1', 'testDir': '/tests' }"
```

**Cron Registration**

**Cron Register options**

**Registration ID:** 1

**Description:** Run STAX Job at Midnight Every Day

**Target Machine:**  Python server1

**Target Service:**  Python STAX

**Target Request:**  Python EXECUTE FILE "%s" ARGS "%s" ' % (xmlFile, args)

**Prepare Script:**

```
xmlFile = 'tests/TestA.xml'
args = { 'machine': 'client1', 'testDir': 'tests' }
```

**Minute:**

**Hour:** 0

**Day:**

**Month:**

**Weekday:**

**Once:**

**Enabled:**

**Register** **Cancel**

## [Edit a Registration](#)

To edit a registration, double click on its row in the registration table, click on "Selected" in the menu bar and select "Edit", or right-click on its row in the registration table, and select "Edit" in the popup menu.

For example, if we wanted to change the registration ID 1 to only execute at 1:00AM Monday-Friday:

**Cron Registration** [X]

**Cron Register options**

Registration ID: **1**

Description:

Target Machine:  Python

Target Service:  Python

Target Request:  Python

Prepare Script:

Minute:

Hour:

Day:

Month:

Weekday:

Once:

Enabled:

**Register** **Cancel**

Click on the "Register" button to unregister the existing ID, and register the updated registration data:

**Cron Registration** [X]

**Cron Register options**

Registration ID: **1**

Description:

Target Machine:  Python

Target Service:  Python

Target Request:  Python

Prepare Script:

Minute:

Hour:

Day:

Month:

Weekday:

Once:

Enabled:

**Register** **Cancel**

**Select an Option** [X]

 Do you want to unregister ID 1 and re-register the STAF request?

**Yes** **No** **Cancel**

Register

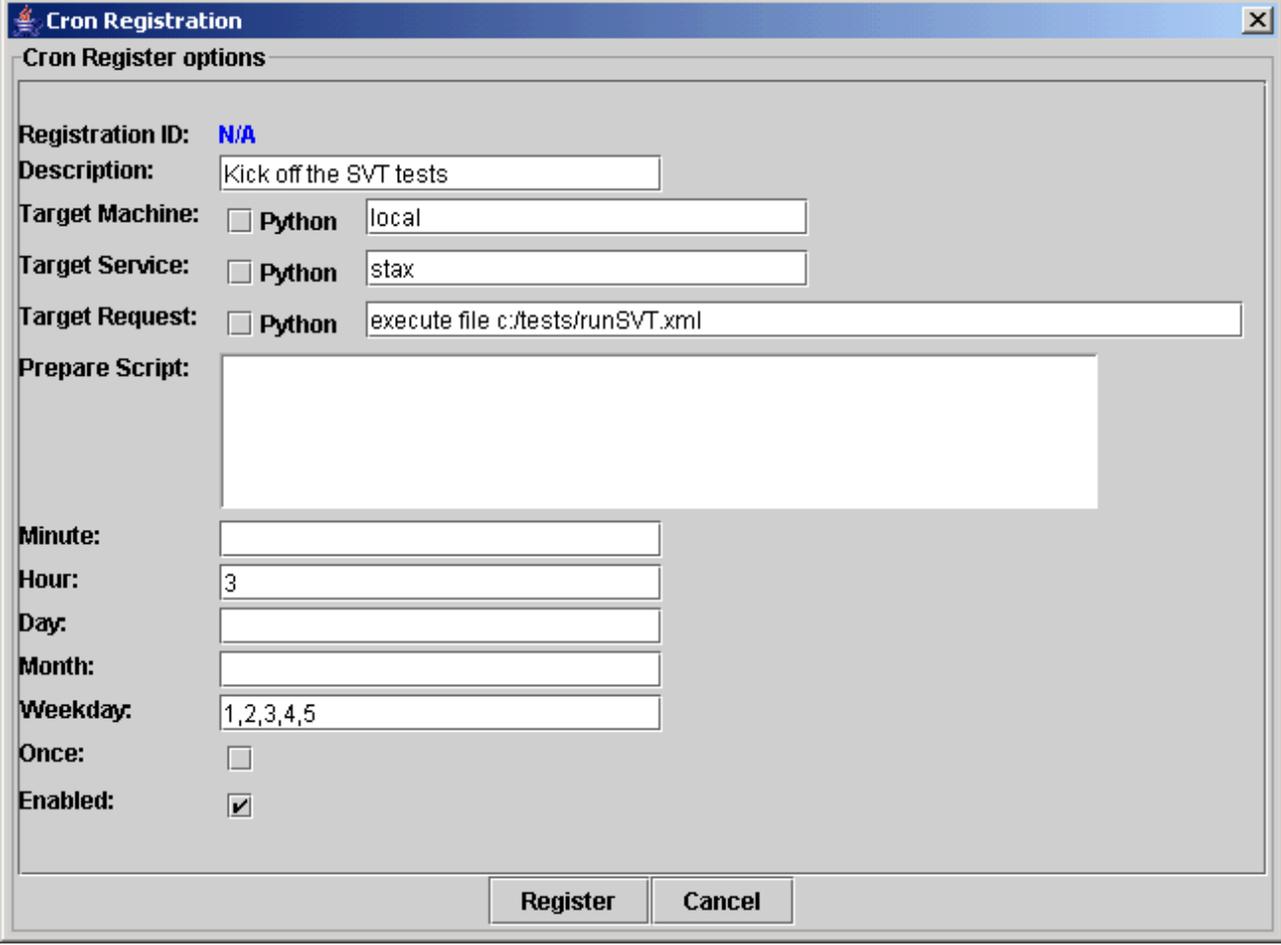
Cancel

Click on yes, and you will see that the updated registration data has been saved. In this scenario, since we only had one existing registration, the ID for the updated registration is still 1. If there are already existing registration IDs, then the assigned registration id would be the next sequential ID.

## Copy a Registration

To copy the information from an existing registration to a new registration, click on "Selected" in the menu bar and select "Copy to new registration", or right-click on its row in the registration table, and select "Copy to new registration" in the popup menu.

Update the information for the new registration:

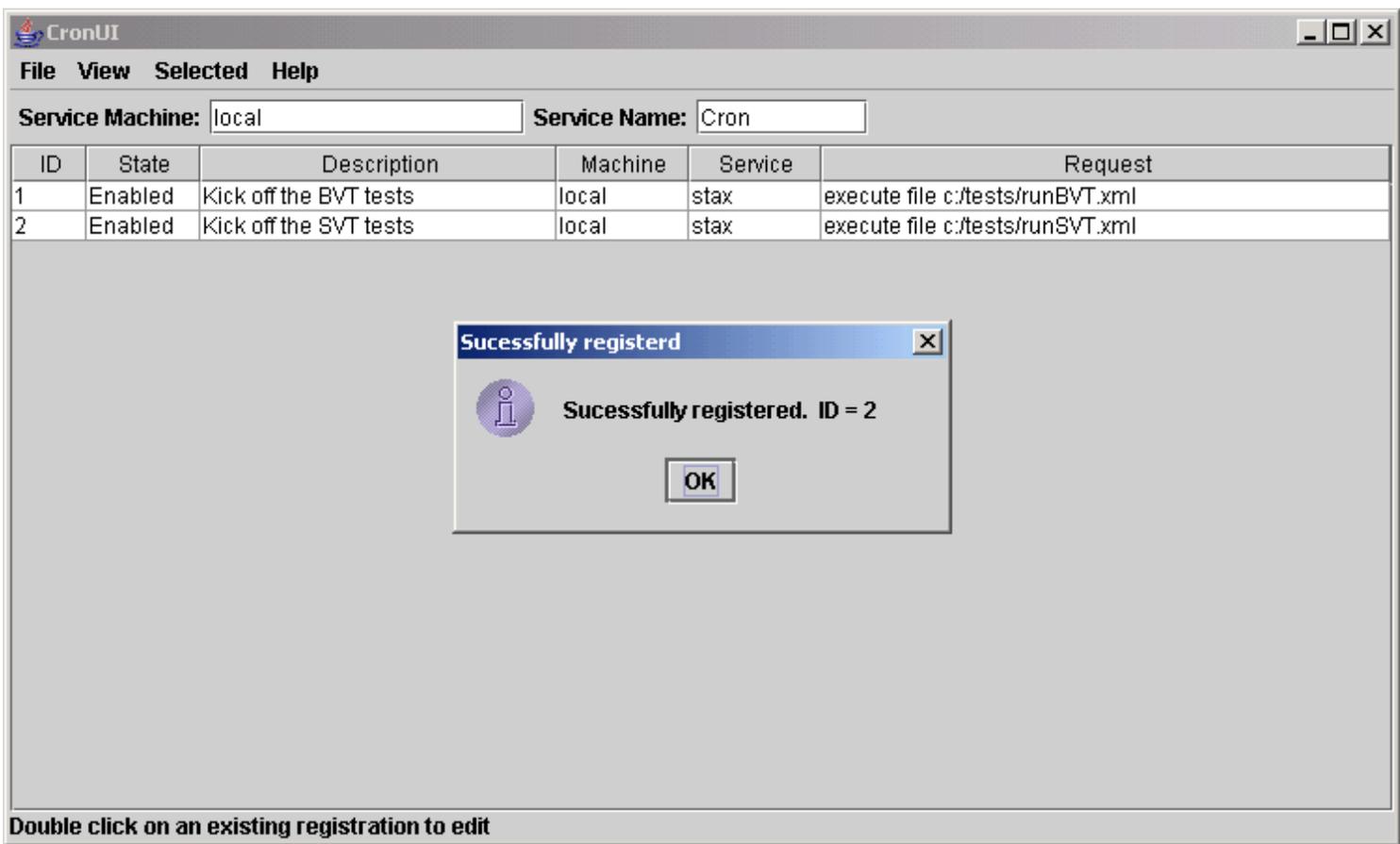


The screenshot shows a dialog box titled "Cron Registration" with a close button (X) in the top right corner. The dialog is titled "Cron Register options" and contains the following fields and controls:

- Registration ID:** N/A
- Description:** Kick off the SVT tests
- Target Machine:**  Python local
- Target Service:**  Python stax
- Target Request:**  Python execute file c:/tests/runSVT.xml
- Prepare Script:** (Empty text area)
- Minute:** (Empty text field)
- Hour:** 3
- Day:** (Empty text field)
- Month:** (Empty text field)
- Weekday:** 1,2,3,4,5
- Once:**
- Enabled:**

At the bottom of the dialog are two buttons: "Register" and "Cancel".

Click on the "Register" button to register.



The screenshot shows the CronUI application window. At the top, there is a menu bar with "File", "View", "Selected", and "Help". Below the menu bar, there are two input fields: "Service Machine:" with the value "local" and "Service Name:" with the value "Cron".

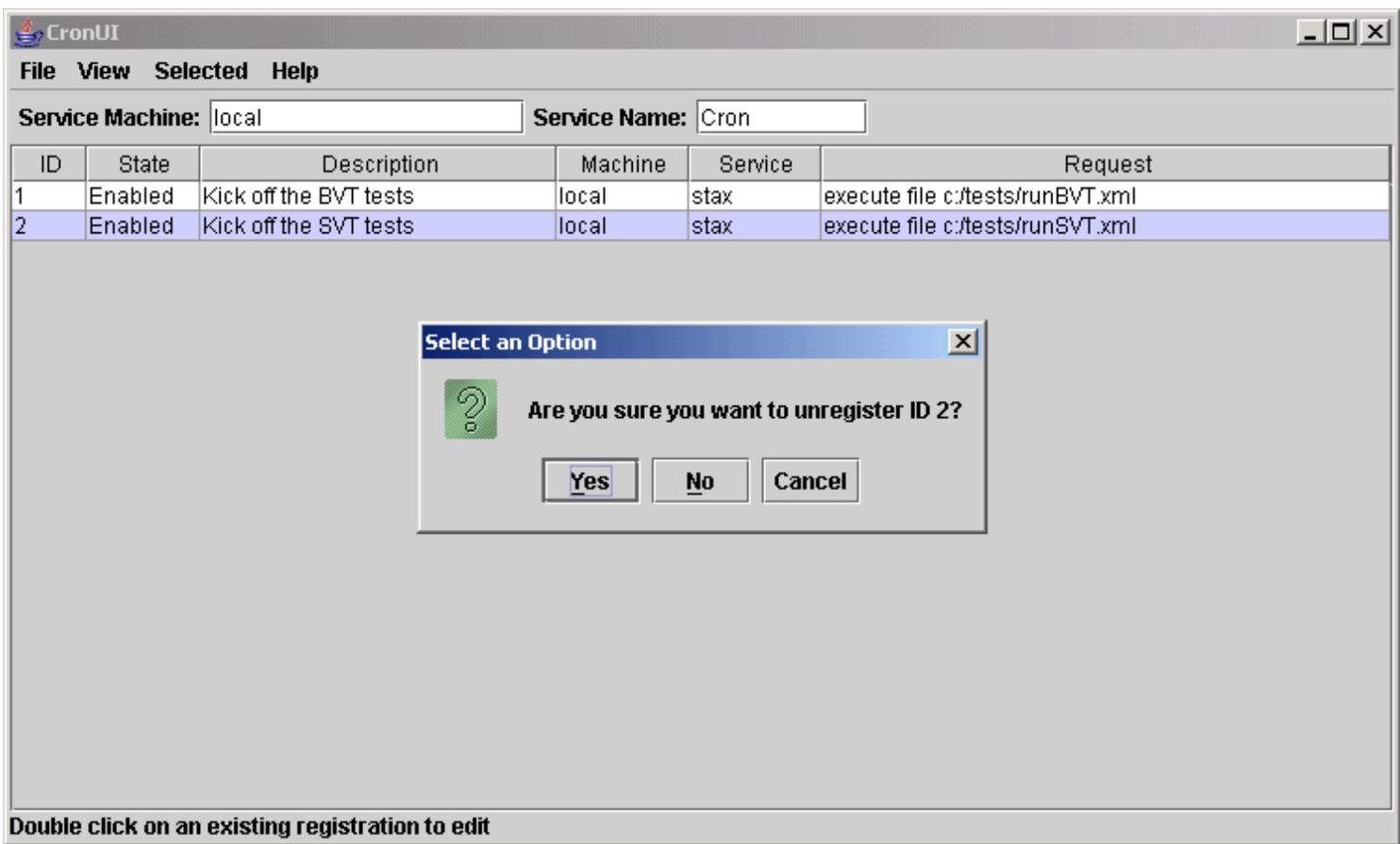
ID	State	Description	Machine	Service	Request
1	Enabled	Kick off the BVT tests	local	stax	execute file c:/tests/runBVT.xml
2	Enabled	Kick off the SVT tests	local	stax	execute file c:/tests/runSVT.xml

In the center of the window, a dialog box titled "Sucessfully registerd" (note the typo) is displayed. It contains an information icon, the text "Sucessfully registered. ID = 2", and an "OK" button.

At the bottom of the window, there is a footer text: "Double click on an existing registration to edit".

## [Unregister a Registration](#)

To unregister a registration, click on "Selected" in the menu bar and select "Unregister", or right-click on its row in the registration table, and select "Unregister" in the popup menu. You will get a confirmation popup:



Click on "Yes" and the ID will be unregistered.

Next, let's register to have the command "java -version" executed at the beginning of every month. In the menu bar click on "File" and then "New Registration..." and fill in the data:

**Cron Registration**

**Cron Register options**

Registration ID: **N/A**

Description:

Target Machine:  Python

Target Service:  Python

Target Request:  Python

Prepare Script:

Minute:

Hour:

Day:

Month:

Weekday:

Once:

Enabled:

**Register** **Cancel**

Click on the "Register" button and we will see that it has been assigned a unique registration ID:

**CronUI**

File View Selected Help

Service Machine:  Service Name:

ID	State	Description	Machine	Service	Request
1	Enabled	Kick off the BVT tests	local	stax	execute file c:/tests/runBVT.xml
3	Enabled	Get the Java version	local	process	start command java parms -version returnstdout stde...

**Successfully registered**

 **Successfully registered. ID = 3**

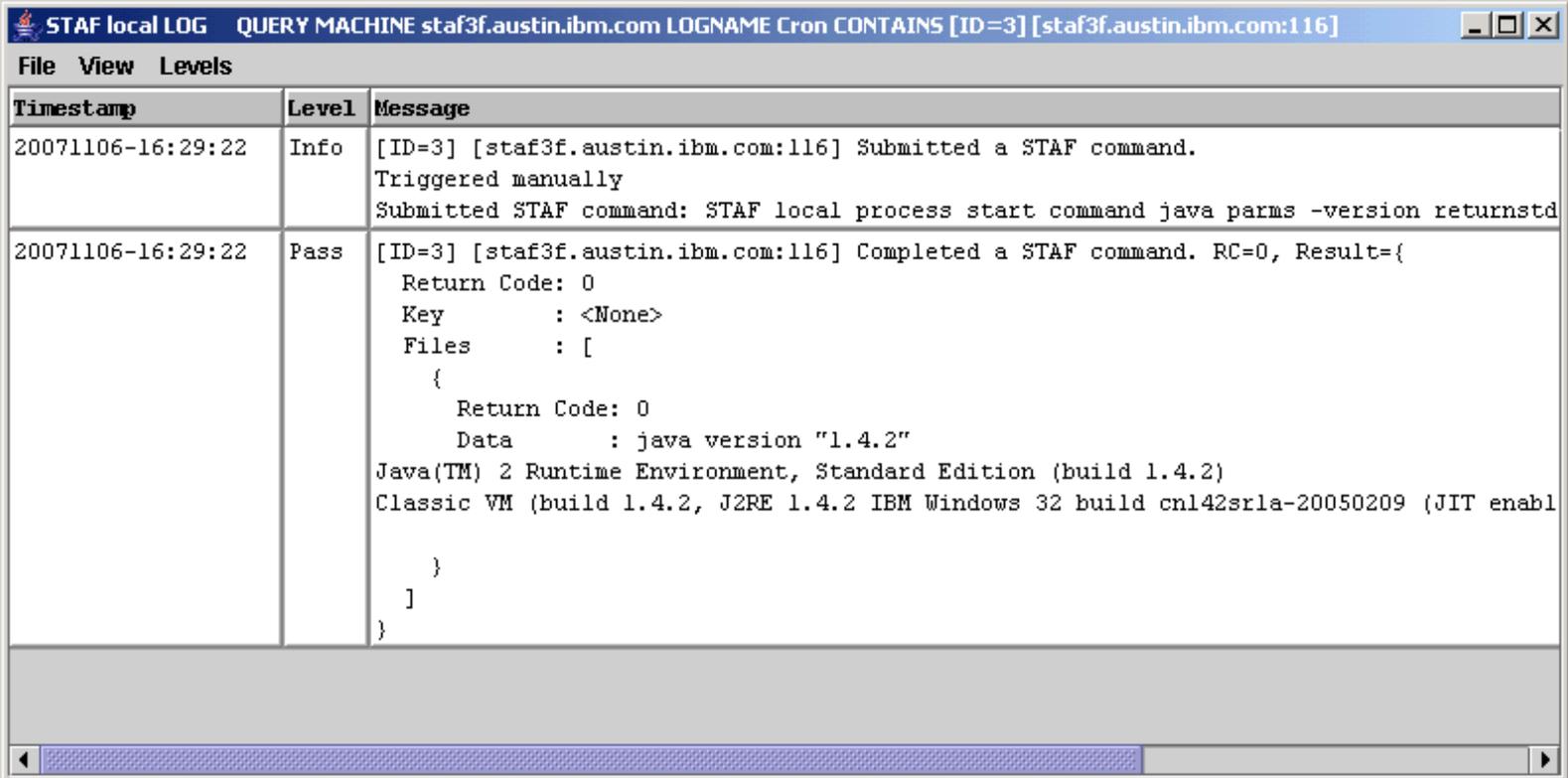
**OK**

Double click on an existing registration to edit

## [Trigger a Registration](#)

Suppose we want to test the STAF command that we just registered, and we don't want to wait until the next time the month changes. To do this, click on "Selected" in the menu bar and select "Trigger", or right-click on its row in the registration table, and select "Trigger" in the popup menu.

The STAF command will be submitted and a STAFLogViewer window will open to display the service log entries for that STAF command:



The screenshot shows a window titled "STAF local LOG QUERY MACHINE staf3f.austin.ibm.com LOGNAME Cron CONTAINS [ID=3] [staf3f.austin.ibm.com:116]". The window has a menu bar with "File", "View", and "Levels". Below the menu bar is a table with three columns: "Timestamp", "Level", and "Message".

Timestamp	Level	Message
20071106-16:29:22	Info	[ID=3] [staf3f.austin.ibm.com:116] Submitted a STAF command. Triggered manually Submitted STAF command: STAF local process start command java parms -version returnstd
20071106-16:29:22	Pass	[ID=3] [staf3f.austin.ibm.com:116] Completed a STAF command. RC=0, Result={ Return Code: 0 Key : <None> Files : [ { Return Code: 0 Data : java version "1.4.2" Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2) Classic VM (build 1.4.2, J2RE 1.4.2 IBM Windows 32 build cnl42sr1a-20050209 (JIT enabl } ] }

In this case, since the process returned the version immediately, the log query shows that the process has completed, and includes the information it returned.

## [View the Service Log](#)

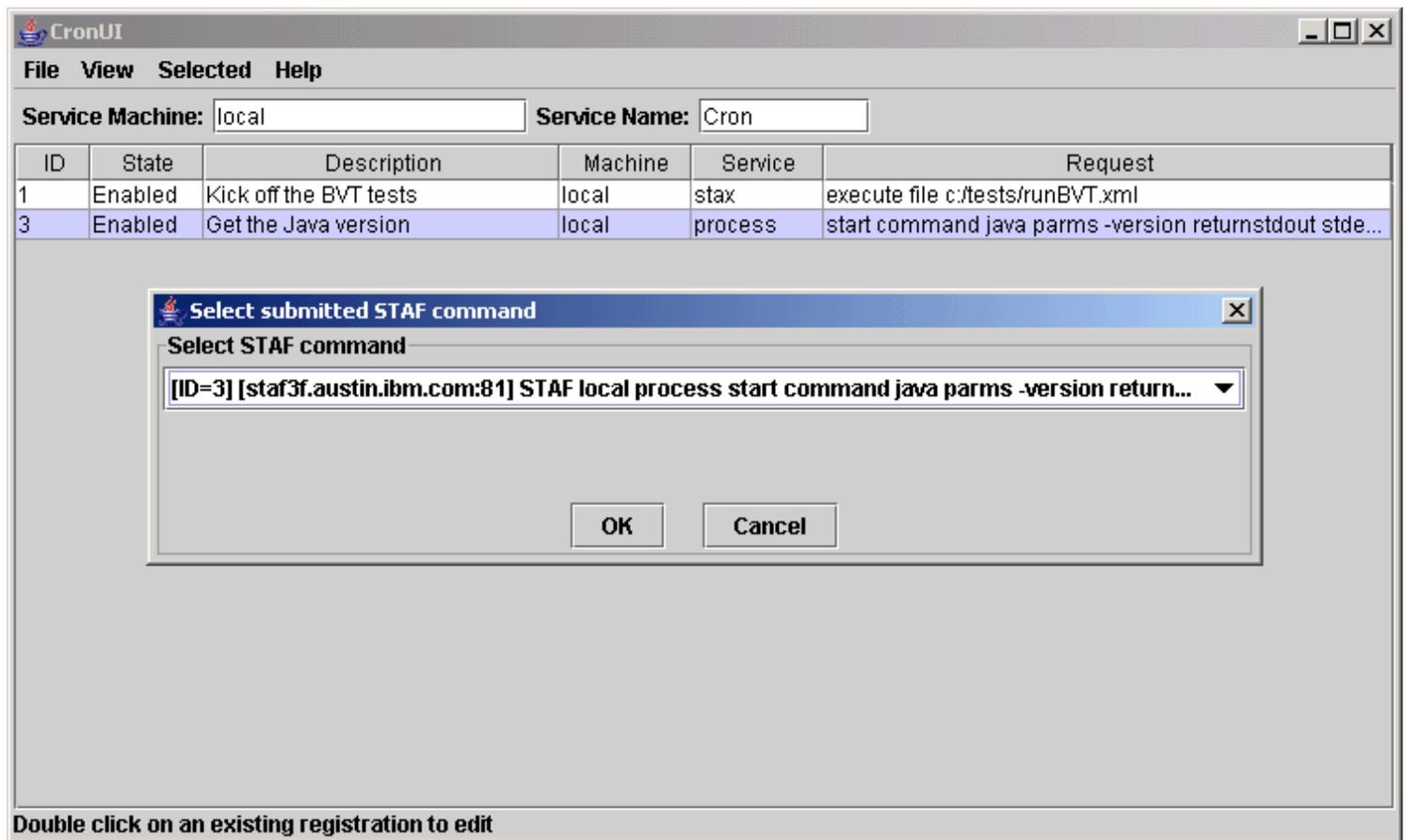
To view the service log's entries for a particular registration ID, click on "View" in the menu bar and then select "Service Log for ID...". In the popup dialog, enter ID "3" and click on OK. A STAFLogViewer window will open with all of the log entries for ID 3:

STAF local LOG QUERY MACHINE staf3f.austin.ibm.com LOGNAME Cron CONTAINS :6:[ID=3]		
Timestamp	Level	Message
20071106-16:25:49	Info	[ID=3] [local://local, STAF/Cron/UI] Registered a STAF command. Register request: REGISTER MACHINE :5:local SERVICE :7:process REQUEST :66:start com
20071106-16:26:44	Info	[ID=3] [local://local, STAF/Cron/UI] Triggering a STAF command. TRIGGER ID 3
20071106-16:26:44	Info	[ID=3] [staf3f.austin.ibm.com:81] Submitted a STAF command. Triggered manually Submitted STAF command: STAF local process start command java parms -version returns
20071106-16:26:45	Pass	[ID=3] [staf3f.austin.ibm.com:81] Completed a STAF command. RC=0, Result={ Return Code: 0 Key : <None> Files : [ { Return Code: 0 Data : java version "1.4.2" Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2) Classic VM (build 1.4.2, J2RE 1.4.2 IBM Windows 32 build cnl42sr1a-20050209 (JIT ena } ] }

To view the last 100 entries in the service log, click on "View" in the menu bar and then select "Service Log last 100 records". A STAFLogViewer window will open with the last 100 entries:

STAF local LOG QUERY MACHINE staf3f.austin.ibm.com LOGNAME Cron LAST 100		
Timestamp	Level	Message
20071106-16:23:09	Info	[ID=2] [local://local, STAF/Cron/UI] Registered a STAF command. Register request: REGISTER MACHINE :5:local SERVICE :4:stax REQUEST :32:execute file
20071106-16:25:04	Info	[ID=2] [local://local, STAF/Cron/UI] Unregistered a STAF command.
20071106-16:25:49	Info	[ID=3] [local://local, STAF/Cron/UI] Registered a STAF command. Register request: REGISTER MACHINE :5:local SERVICE :7:process REQUEST :66:start com
20071106-16:26:44	Info	[ID=3] [local://local, STAF/Cron/UI] Triggering a STAF command. TRIGGER ID 3
20071106-16:26:44	Info	[ID=3] [staf3f.austin.ibm.com:81] Submitted a STAF command. Triggered manually Submitted STAF command: STAF local process start command java parms -version returns
20071106-16:26:45	Pass	[ID=3] [staf3f.austin.ibm.com:81] Completed a STAF command. RC=0, Result={ Return Code: 0 Key : <None> Files : [ { Return Code: 0 Data : java version "1.4.2" Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2) Classic VM (build 1.4.2, J2RE 1.4.2 IBM Windows 32 build cnl42sr1a-20050209 (JIT ena } ] }

To view the log entries for a particular submitted STAF command, click on "View" in the menu bar and then select "Service log for submitted STAF command...". A new dialog will be displayed showing a list of all of the submitted STAF commands:



Select the submitted STAF command, and click on "OK". A STAFLogViewer window will open with all of the log entries for that submitted STAF command:

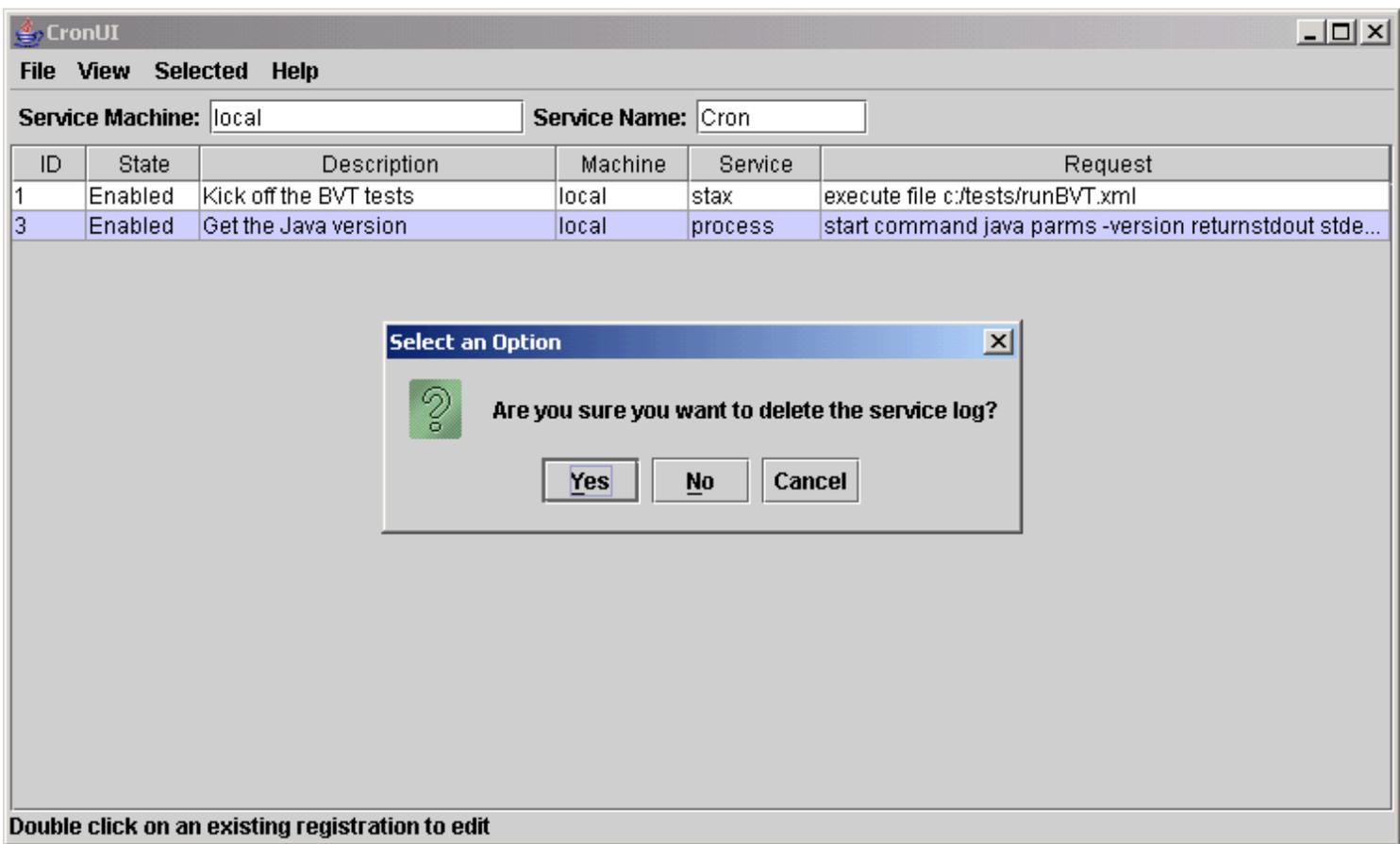
STAF local LOG QUERY MACHINE staf3f.austin.ibm.com LOGNAME Cron CONTAINS :33:[ID=3] [staf3f.austin.ibm.com:81] ALL		
Timestamp	Level	Message
20071106-16:26:44	Info	[ID=3] [staf3f.austin.ibm.com:81] Submitted a STAF command. Triggered manually Submitted STAF command: STAF local process start command java parms -version returnstd
20071106-16:26:45	Pass	[ID=3] [staf3f.austin.ibm.com:81] Completed a STAF command. RC=0, Result={ Return Code: 0 Key : <None> Files : [ { Return Code: 0 Data : java version "1.4.2" Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2) Classic VM (build 1.4.2, J2RE 1.4.2 IBM Windows 32 build cnl42srla-20050209 (JIT enabl } ] }

To view the entire service log, click on "View" in the menu bar and then select "Entire Service Log". A STAFLogViewer window will open with all of the log entries:

STAF local LOG QUERY MACHINE staf3f.austin.ibm.com LOGNAME Cron ALL		
Timestamp	Level	Message
20071106-16:25:49	Info	[ID=3] [local://local, STAF/Cron/UI] Registered a STAF command. Register request: REGISTER MACHINE :5:local SERVICE :7:process REQUEST :66:start com
20071106-16:26:44	Info	[ID=3] [local://local, STAF/Cron/UI] Triggering a STAF command. TRIGGER ID 3
20071106-16:26:44	Info	[ID=3] [staf3f.austin.ibm.com:81] Submitted a STAF command. Triggered manually Submitted STAF command: STAF local process start command java parms -version returns
20071106-16:26:45	Pass	[ID=3] [staf3f.austin.ibm.com:81] Completed a STAF command. RC=0, Result={ Return Code: 0 Key : <None> Files : [ { Return Code: 0 Data : java version "1.4.2" Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2) Classic VM (build 1.4.2, J2RE 1.4.2 IBM Windows 32 build cnl42srla-20050209 (JIT ena } ] }

## [Delete the Service Log](#)

To delete the service log, click on "View" in the menu bar and then select "Delete Service Log". You will receive a confirmation popup:



## Cron Error Codes

In addition to the common STAF return codes, the following Cron return codes are defined:

**Table 1. Cron Service Return Codes**

Error Code	Meaning	Comment
4001	Python error	A Python error during a TRIGGER command occurred while evaluating the request to be submitted. The result buffer will include details about the Python error.
4002	Request not submitted	The request for a TRIGGER command was not submitted because the value of STAF_Cron_Submit was not set to true. The result buffer will include the value of STAF_Cron_Submit.

## Appendix A: References

- See the <http://www.jython.org> website for more information about Jython.
- See the <http://www.python.org> website for more information about Python.

## Appendix B: Jython and CPython Differences

Although in most cases Jython behavior is identical to the C-language implementation of Python (CPython), there are still cases where the two implementations differ. If you are already a CPython programmer, or are hoping to use CPython code under Jython, you need to be aware of these differences. Also, there is a time lag between a new CPython release and the corresponding Jython release. The Cron service uses Jython 2.5.2 which implements the same set of language features as CPython 2.5. Jython 2.5.2 cannot execute Python code that uses functions that were provided in later versions of Python, such as Python 2.6.

Most Python modules that are written in Python work fine in Jython. A few types of modules will not run under Jython such as:

- Modules that contain functionality not included in a JVM

Some standard CPython modules depend on operating system calls that are not available under Java. The most notable of these is *os*, which actually does run in Jython, but is missing much of its functionality.

- Modules that are implemented in C

A number of common CPython modules are implemented in C rather than Python, either for a speed boost or because the module is a C wrapper around an external C library. The C modules, or any modules that depend on them, will not run in Jython.

See the "Jython Essentials" book, written by Samuele Pedroni and Noel Rappin, for more information about the differences between Jython and CPython.

## Appendix C: Licenses and Acknowledgements

### Jython

Jython is an implementation of the high-level, dynamic, object-oriented language Python written in 100% Pure Java, and seamlessly integrated with the Java platform. It thus allows you to run Python on any Java platform.

### Acknowledgement

This product includes software developed by the Jython Developers (<http://www.jython.org/>).

### Licence

PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2

-----

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using this software ("Jython") in source or binary form and

its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Jython alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright (c) 2007 Python Software Foundation; All Rights Reserved" are retained in Jython alone or in any derivative version prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on or incorporates Jython or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Jython.

4. PSF is making Jython available to Licensee on an "AS IS" basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF JYTHON WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF JYTHON FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING JYTHON, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Jython, Licensee agrees to be bound by the terms and conditions of this License Agreement.

Jython 2.0, 2.1 License

=====

Copyright (c) 2000-2009 Jython Developers.

All rights reserved

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of the Jython Developers nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.