Software Testing Automation Framework (STAF)

Open Source Testing System from IBM
What is STAF?

- An open source automation framework
- A remote test agent to control tests on multiple machines
- Designed around the idea of reusable components, called services
- Developed by IBM, made open source because of their use in Linux testing
Supported Environments

- Runs on most operating systems: Windows, Unix, AS/400, MVS
- The same services are available from a wide array of languages, including C/C++, Java, Perl, Tcl, Python and Rexx
- There is also a command line interface
Basic Concepts

- Runs as a daemon process, called **STAFProc**, on each system.
- Operates in a peer-to-peer environment; in other words, there is no client-server hierarchy.
Services

- Services are reusable components that provide all the capabilities in STAF.
- Provides a specific set of functionality (such as Logging) and defines a set of requests that it will accept.
- STAFProc sends the requests to services as strings which describe the operation to perform.
- Requests can be sent to services on the local machine or to another remote machine.
Handles and Queues

- Every process that accesses STAF does so through a **handle**

- A handle, combined with the machine name, uniquely identifies a particular process
  - The combination of machine name and handle allows services to track requests from multiple processes on different machines

- Each handle has a priority-based message **queue** associated with it

- Applications receive messages sent from other processes/machines on their queue
Variables

- STAF provides facilities to store and retrieve **variables** per handle, such as:
  - Configuration information
  - Runtime/state information
  - System environment information

- Live within the STAFProc daemon, which allows them to be dynamically updated.
  - After the update, applications referencing the variable will get the new value.

- A global variable pool is common to all the processes on a machine.
Security

- Security defined at the machine level, as opposed to using individual userids
- Numeric trust level can be assigned to specific machines
- Each service defines what trust level is required in order to use the various functions provided
- A simple numerical comparison is used to see if the request is authorized
Types of Services

☐ Internal STAF Services

- The executable code for internal STAF services resides within STAFProc, which means they are always available and have a fixed name

☐ External STAF Services

- The executable code for external STAF services resides outside of STAFProc, for example in a Java jar file, a C++ DLL file, or a Rexx script file

☐ Custom STAF Services

- Note that you can also write your own custom services that can be plugged into STAF.
- All custom services are external services
<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>List services available and examine requests</td>
</tr>
<tr>
<td>FS</td>
<td>Get and copy files across a network</td>
</tr>
<tr>
<td>PROCESS</td>
<td>Start, stop and query processes</td>
</tr>
<tr>
<td>LOG</td>
<td>Full featured logging facility, can be replaced</td>
</tr>
<tr>
<td>MONITOR</td>
<td>Publish test case running execution status</td>
</tr>
<tr>
<td>SXE</td>
<td>Sequentially execute a number of commands</td>
</tr>
<tr>
<td>SEM</td>
<td>Networkable event and mutex semaphores</td>
</tr>
<tr>
<td>CRON</td>
<td>Run a command at specific time interval</td>
</tr>
<tr>
<td>EVENT</td>
<td>Publish/Subscribe notification system</td>
</tr>
<tr>
<td>TIMER</td>
<td>Periodically receive a notification message</td>
</tr>
</tbody>
</table>
Configuration File

- STAF is configured through a text file
  - This file may have any name you desire, but the default is STAF.cfg

- You can alter many aspects of STAF's behavior
  - Specify the network interfaces
  - Define operational parameters
  - Define global variables
  - Specify security access
  - Define Startup/Shutdown notifications
  - Enable and configure tracing
  - Register and configure external services
Configuration File

- Grant access with trust levels
  
  TRUST LEVEL 5 MACHINE office
  TRUST LEVEL 0 MACHINE badguy
  TRUST DEFAULT LEVEL 1

- External service registration

  SERVICE MONITOR LIBRARY STAFMon
  SERVICE SAMPLEJ LIBRARY JSTAF \ 
    EXECUTE C:/STAF/services/Sample.jar \ 
    OPTION "J2=-cp C:/MyJava/Extra.zip" \ 
    PARMS {STAF/Config/STAFRoot}/bin/sample.dft
  SERVICE EVENT LIBRARY JSTAF \ 
    EXECUTE C:/STAF/services/STAFEvent.jar
  SERVICE NOTIFY LIBRARY Notify PARMS "HOURS 24 DAYS 7"
Registering Services Dynamically

- You may also register and unregister services dynamically, without needing to shutdown and restart STAF

  \- ADD SERVICE Log LIBRARY STAFLog
  
  \- ADD SERVICE MyDevice LIBRARY PLSTAF EXECUTE mydev.pl
  
  \- ADD SERVICE STAX LIBRARY JSTAF EXECUTE STAX.jar
  
  \- REMOVE SERVICE STAX
Service Loaders

- External services whose purpose is to load other services on-demand
- They dynamically register the service when a request is made
- A default service loader service (STAFDSLS) is shipped with STAF. It knows how to dynamically load the Log, Monitor, and ResPool services. This service will automatically be configured

`serviceloader Library STAFDSLS`
Simple Test Cases

- Can run Test Cases which are completely unaware of STAF
- You can start using your existing test cases with STAF without making any changes to the test cases
  - You aren't required to use any STAF services
  - You aren't required to call any STAF APIs
- You can choose when/if you enable STAF in your test cases

```
staf local process start shell command
"perl SimpleTestcase.pl"
```
STAF Enabled Test Cases

- You can leverage STAF in your test cases by making calls into services
- For all of the supported STAF languages, you can do the following
  - Register with STAF
  - Submit any number of calls into services
  - Optionally unregister with STAF
STAF Support for Perl Test Cases

- **STAF::Register**
  - Allows you to register with STAF (procedural)

- **STAF::Submit**
  - Allows you to submit requests to STAF (procedural)

- **STAF::UnRegister**
  - Allows you to unregister with STAF (procedural)

- **STAF::STAFHandle**
  - Object to call into STAF (object-oriented)
    - `new` - Creates a handle that registers with STAF
    - `submit` - Same as STAF::Submit but different calling convention
    - `unRegister` - Same as STAF::UnRegister but different calling convention

- **STAF::WrapData**
  - This function takes a string and produces the colon-length-colon delimited version of that string. This function is widely used to pass the values of options in STAF requests.
Writing Custom Services

- Written in Perl, Java, C++ or REXX
  - Perl services since version 3.0 beta 4

- Life cycle of service:
  - Construction: The service exists in the STAF memory space; not yet ready to accept requests
  - Initialization: The service has been initialized (with appropriate parameters), and is ready to accept requests
  - Accepting requests: The service is active
  - Termination: The service has terminated
  - Deconstruction: The service is removed from the STAF memory space
STAX Service

- A test harness and test execution language with an XML based grammar
- Provides a powerful GUI monitoring application
- The language has a number of unique logic primitives
  - Parallel iterate command: each iteration is run concurrently in a separate thread
- Three types of STAX machines
  - STAX/Event Service Machine
  - Monitoring/Requesting Machine
  - Execution Machines
Perl STAF Service Basics

- Most services are comprised of a single script file

```perl
package MyDevice;

use PLSTAFService;
use 5.008;
use threads;
use threads::shared;
```

- Each service requires a unique package name
- Only works with Perl 5.8 or later
- Threads and shared threads allow requests to modify data without corruption
Perl STAF Service Initialization

- Must implement the `init` sub
  - Register the service
  - Process service parameters
  - Create a directory for persistent service data
  - Create command parsers

```perl
our $fhandle;

sub init {
    my $name = $_{serviceName};
    $fhandle = STAFHandle->new("STAF/Service/$name");
    ...
    ...
    ...
}
```
Perl STAF Service Parsers

- During initialization, a command parser is created for each request the service accepts
  - such as LIST, ADD, QUERY
    - `->new()` creates the parser
    - `->addOption($name, $max, $valueRequirement)` to add options to parser
    - `->addOptionGroup($optionNames, $min, $max)` to specify mutually exclusive option groups
    - `->addOptionNeed($needers, $needees)` to specify option dependency relationships
Perl STAF Service Accept Requests

- Requests are handled by `acceptRequest` sub
- A hash is passed with the request values
- The service then defines `handleCommand` methods for each request type
- Each handler should check the trust level to validate that the requesting machine has access to the method
- The handler method can then parse the request and return the result
STAF 3.0 In Beta

- Send Variables Across the Network
- Inform on Garbage Collected Handles
- Communication Interface Enhancements
- User-level Security
- Multiple copies of STAF on same machine
- Secure connections with OpenSSL