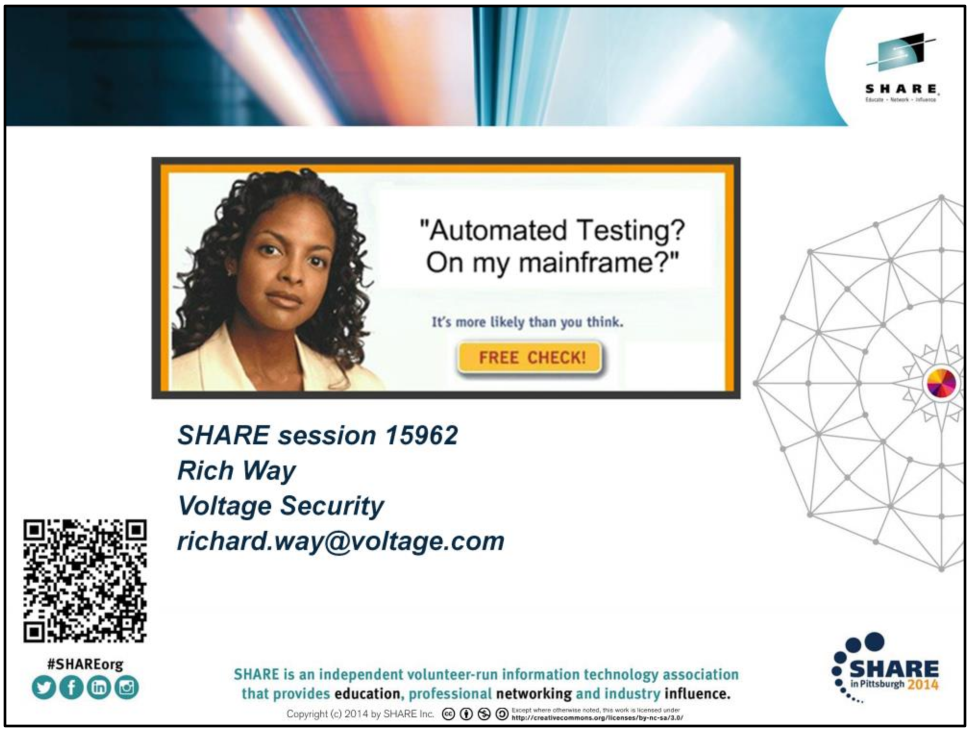


"Automated Testing? On my mainframe?"

It's more likely than you think.

FREE CHECK!

SHARE session 15962
Rich Way
Voltage Security
richard.way@voltage.com

#SHAREorg

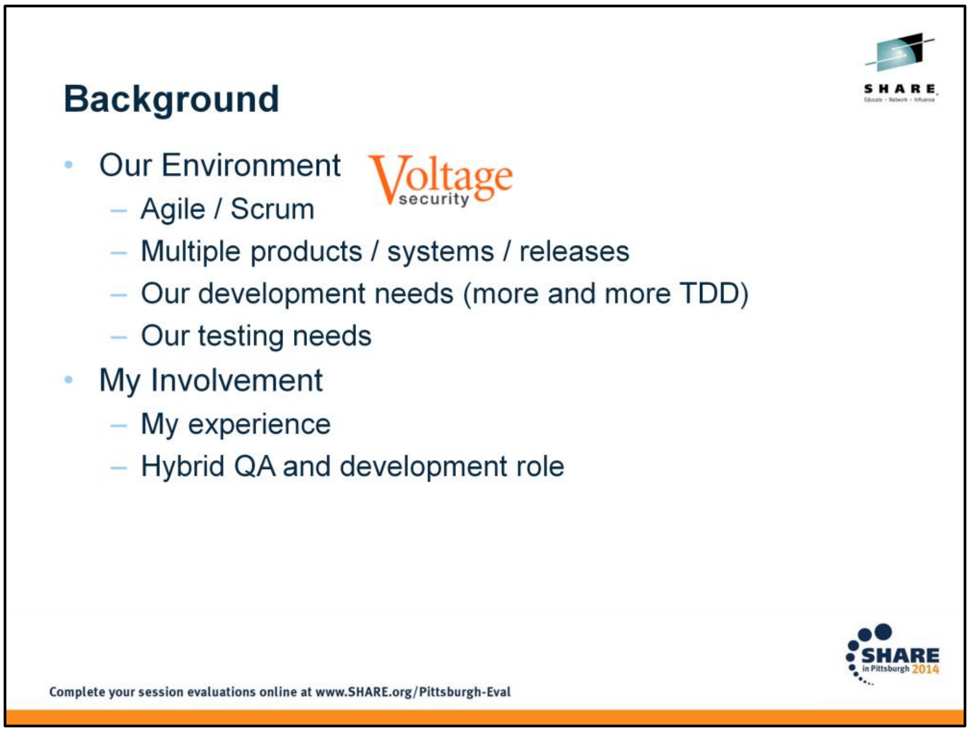
SHARE is an independent volunteer-run information technology association that provides education, professional networking and industry influence.

SHARE in Pittsburgh 2014

Anyone remember this old banner ad? (it was for Net Nanny – "Pornography? On *MY* Computer? It's more likely than you think.")

Mentioned product names may be trademarks and / or copyrighted by their respective owners.

QR code appears on last slide too.

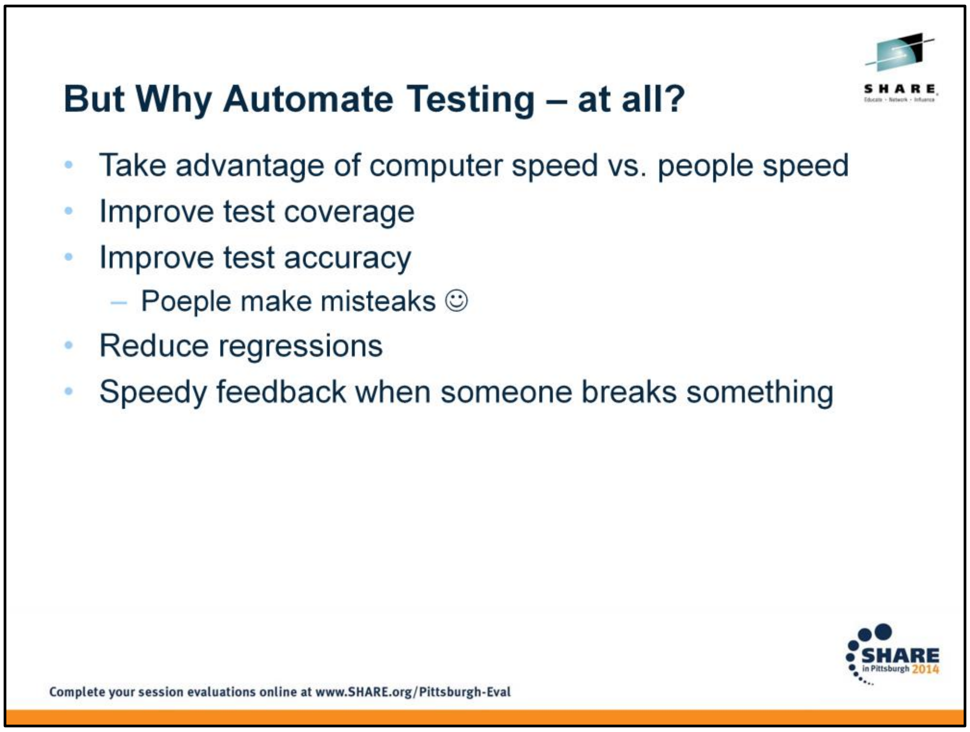


Background

- Our Environment
  - Agile / Scrum
  - Multiple products / systems / releases
  - Our development needs (more and more TDD)
  - Our testing needs
- My Involvement
  - My experience
  - Hybrid QA and development role

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

Agile/Scrum: both development speed and code quality are extremely important.
TDD = test-driven development (write the test first).
My experience – insurance / finance, multiple ISVs (ADS / Centura / Compuware (Xpediter), DSSI (PRO/JCL, JOB/SCAN)), IBM (Enterprise Content Manager), Voltage Security (SecureData, SecureMail)

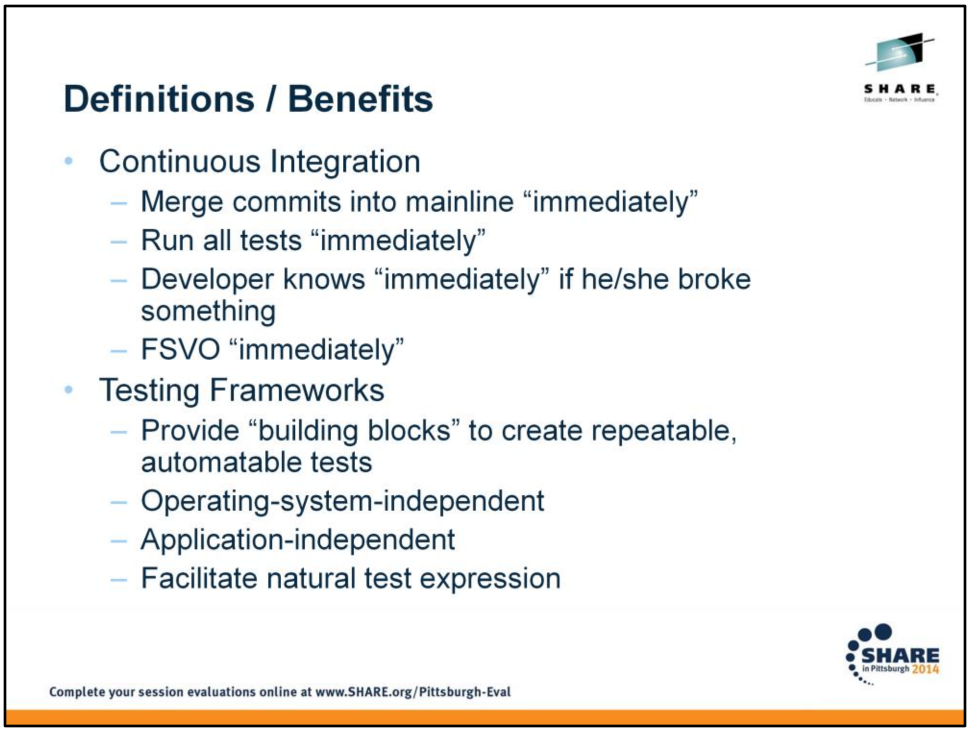


But Why Automate Testing – at all?

- Take advantage of computer speed vs. people speed
- Improve test coverage
- Improve test accuracy
  - Poeple make misteaks ☺
- Reduce regressions
- Speedy feedback when someone breaks something

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

Improve test coverage: for example, it may be impractical to execute performance tests “by hand” using large amounts of randomly-generated data.

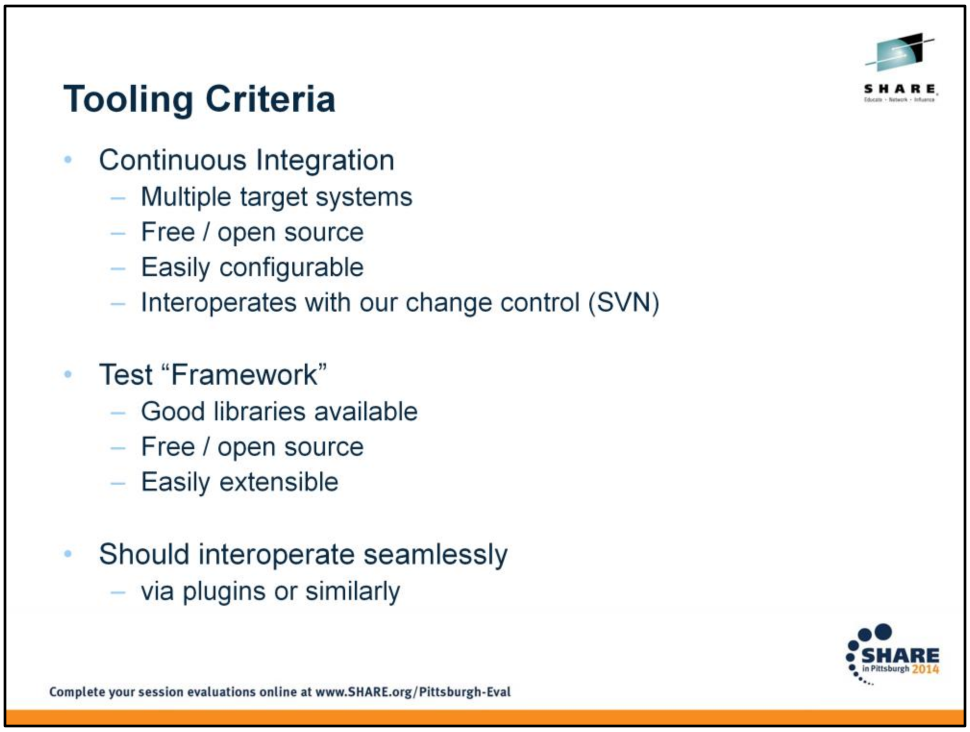


Definitions / Benefits

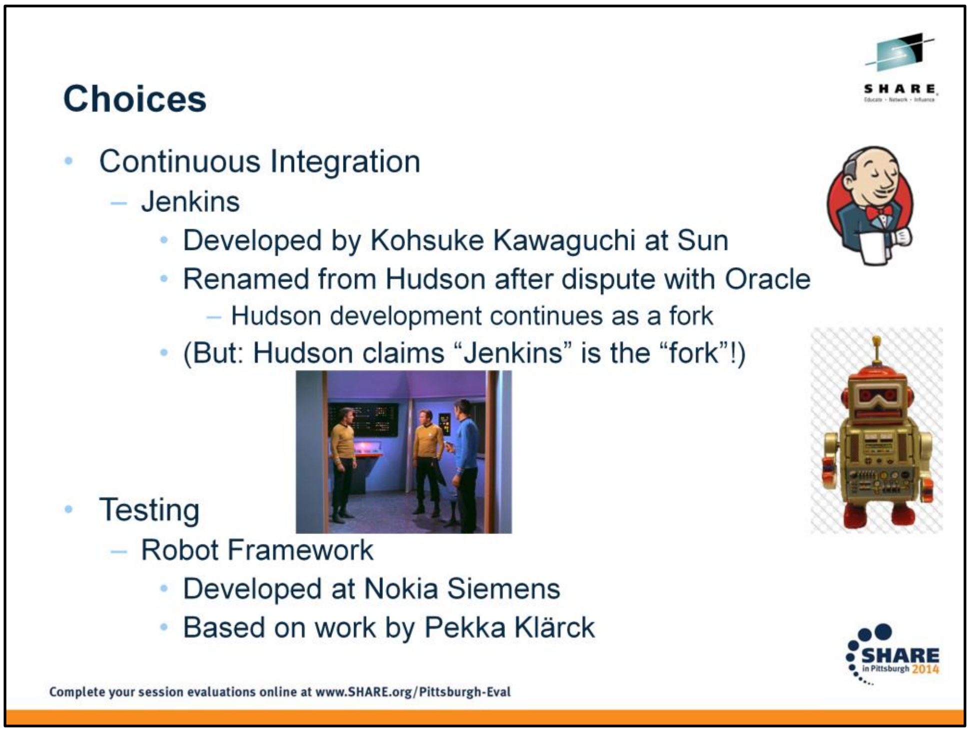
- Continuous Integration
  - Merge commits into mainline "immediately"
  - Run all tests "immediately"
  - Developer knows "immediately" if he/she broke something
  - FSVO "immediately"
- Testing Frameworks
  - Provide "building blocks" to create repeatable, automatable tests
  - Operating-system-independent
  - Application-independent
  - Facilitate natural test expression

NOTE: Continuous integration is NOT a substitute for unit testing! Developers shouldn't be committing code that breaks the builds.

Immediacy depends on how long the tests take to give feedback. Some organizations choose to have a "sanity-check" level of testing that runs more quickly (and more often), backed by a "full" build that runs less often. Failures found in the "full" build wouldn't be reported as quickly.


Tooling Criteria

- Continuous Integration
  - Multiple target systems
  - Free / open source
  - Easily configurable
  - Interoperates with our change control (SVN)
- Test “Framework”
  - Good libraries available
  - Free / open source
  - Easily extensible
- Should interoperate seamlessly
  - via plugins or similarly

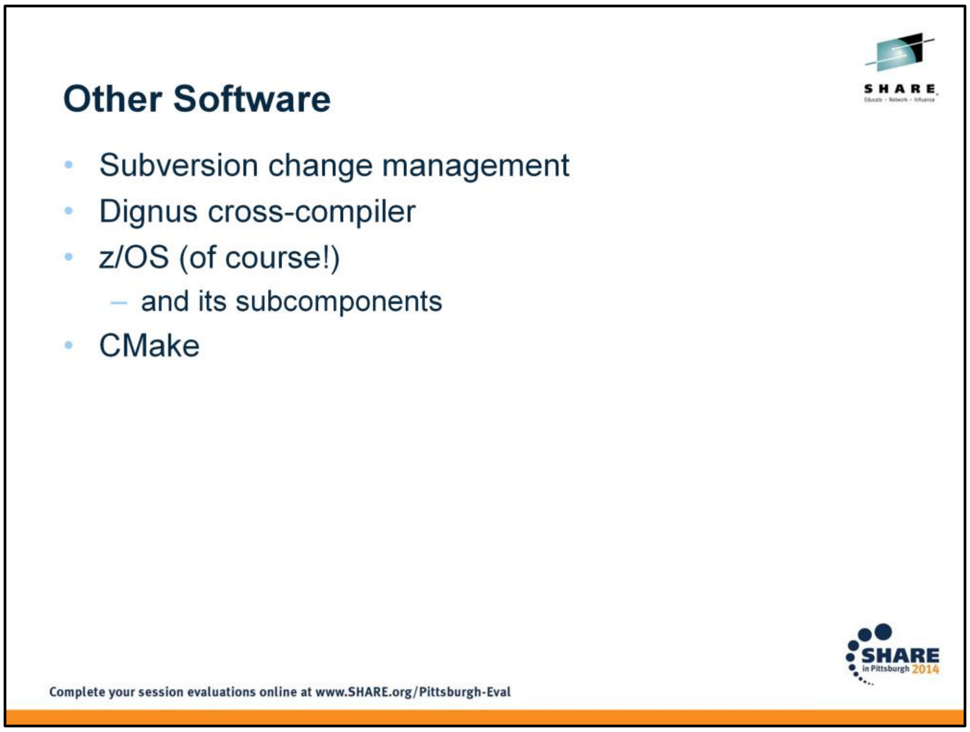


Choices

- Continuous Integration
  - Jenkins
    - Developed by Kohsuke Kawaguchi at Sun
    - Renamed from Hudson after dispute with Oracle
      - Hudson development continues as a fork
    - (But: Hudson claims “Jenkins” is the “fork”!)
- Testing
  - Robot Framework
    - Developed at Nokia Siemens
    - Based on work by Pekka Klärck

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

Can anyone identify this episode? “Whom Gods Destroy”. Spock had to know which Kirk was “evil” (good Kirk told him to kill them both).
Too many continuous integration products to go into detail; see references at end.
Pekka Klarck is still the prime contributor for Robot Framework.

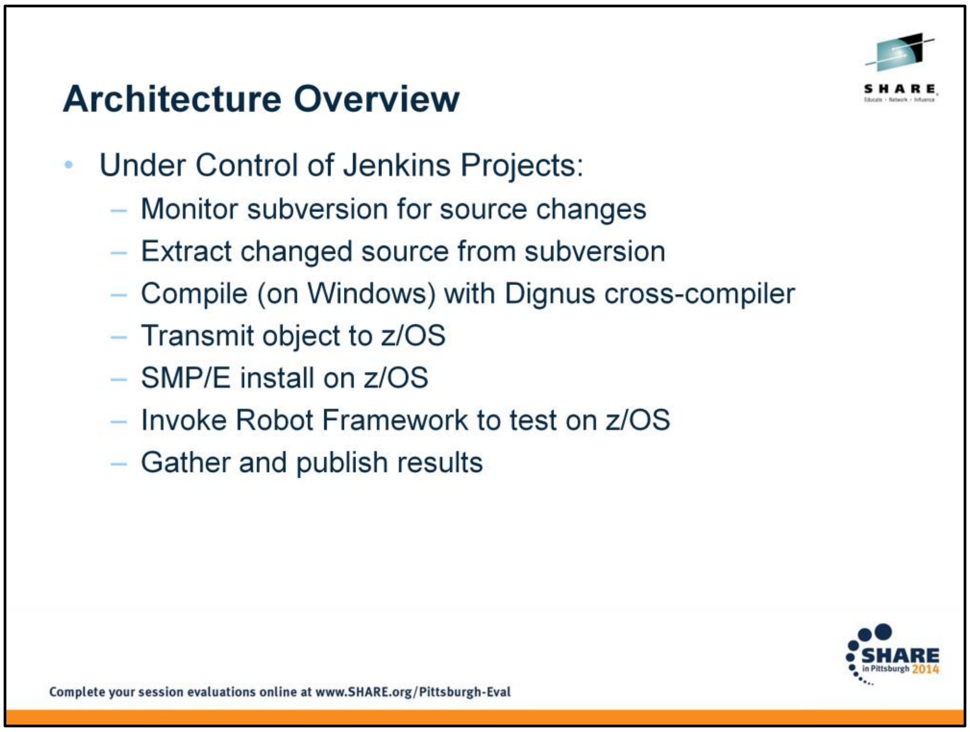


Other Software

- Subversion change management
- Dignus cross-compiler
- z/OS (of course!)
  - and its subcomponents
- CMake

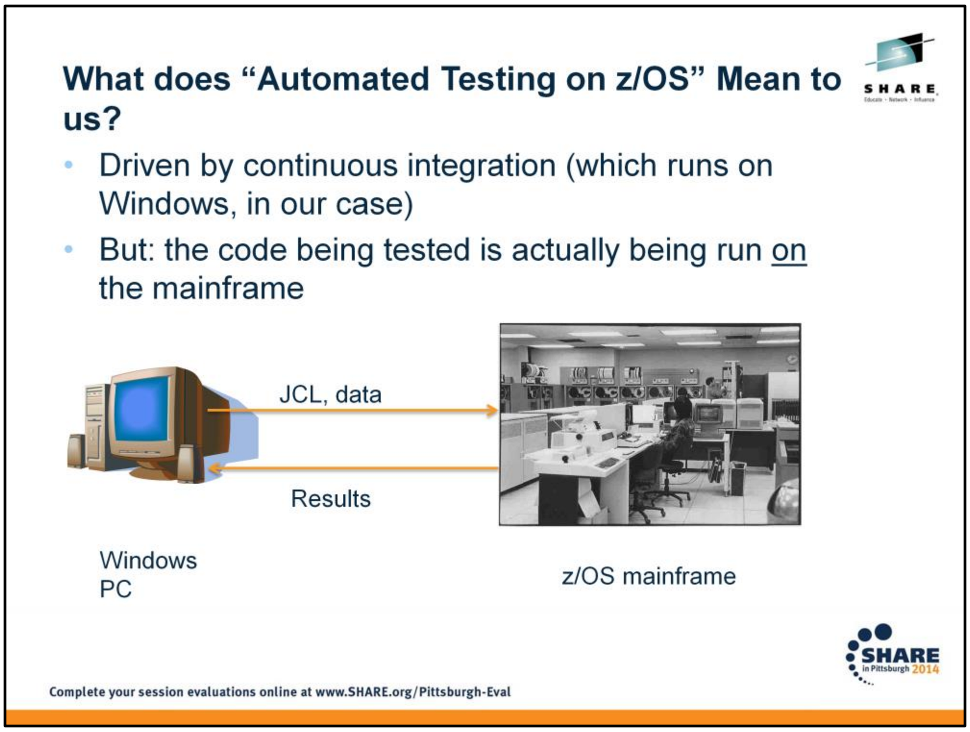
Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

(We use Dignus cross-compiler to generate z/OS object code on a Windows machine. We upload it to the mainframe and link it there. The reason is primarily that we rent time on an IBM-supplied mainframe, and we don’t want to store our proprietary source anywhere outside the company’s control, nor do we want to incur the cost of extra storage and CPU cycles to do the compiles there. If you have your own mainframe, this probably isn’t an issue for you).



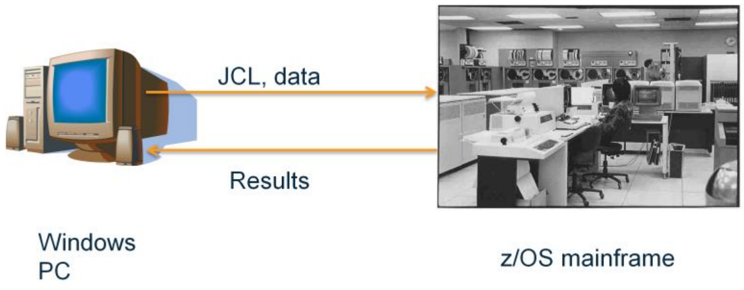
Architecture Overview

- Under Control of Jenkins Projects:
  - Monitor subversion for source changes
  - Extract changed source from subversion
  - Compile (on Windows) with Dignus cross-compiler
  - Transmit object to z/OS
  - SMP/E install on z/OS
  - Invoke Robot Framework to test on z/OS
  - Gather and publish results



What does "Automated Testing on z/OS" Mean to us?

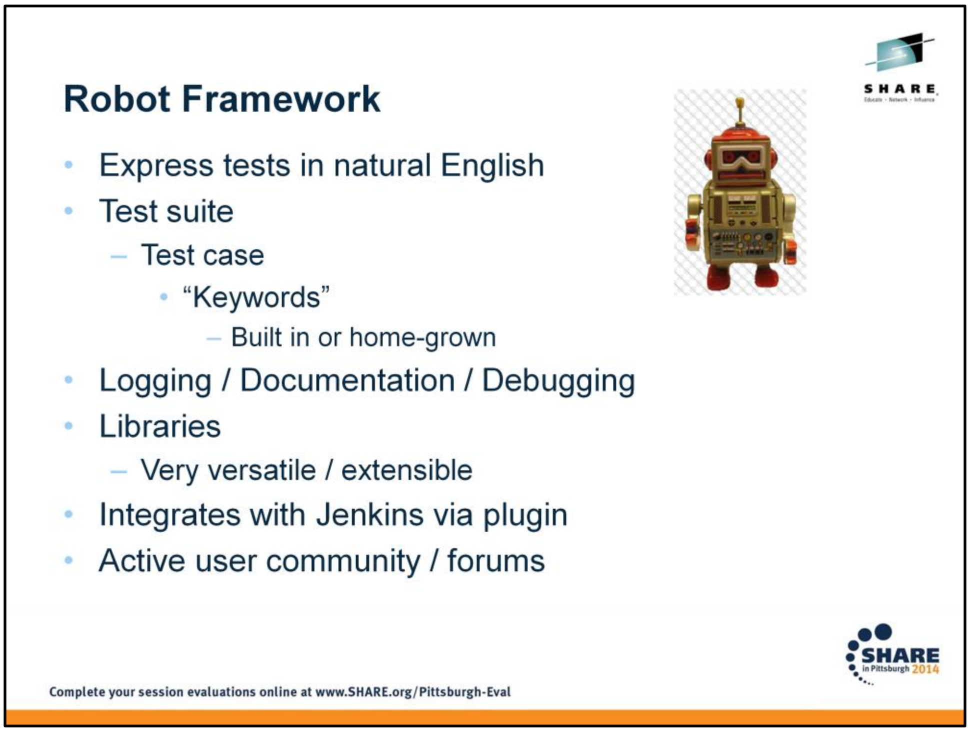
- Driven by continuous integration (which runs on Windows, in our case)
- But: the code being tested is actually being run <u>on</u> the mainframe



Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

(Since I found an image of an old clunky desktop for the Windows machine, I thought I'd keep to that theme for the mainframe image too ☺).

Note: this is *not* instrumentation of the code with #ifdefs (or equivalent) that stub things out to make it compile and (sort of) execute on Windows



Robot Framework

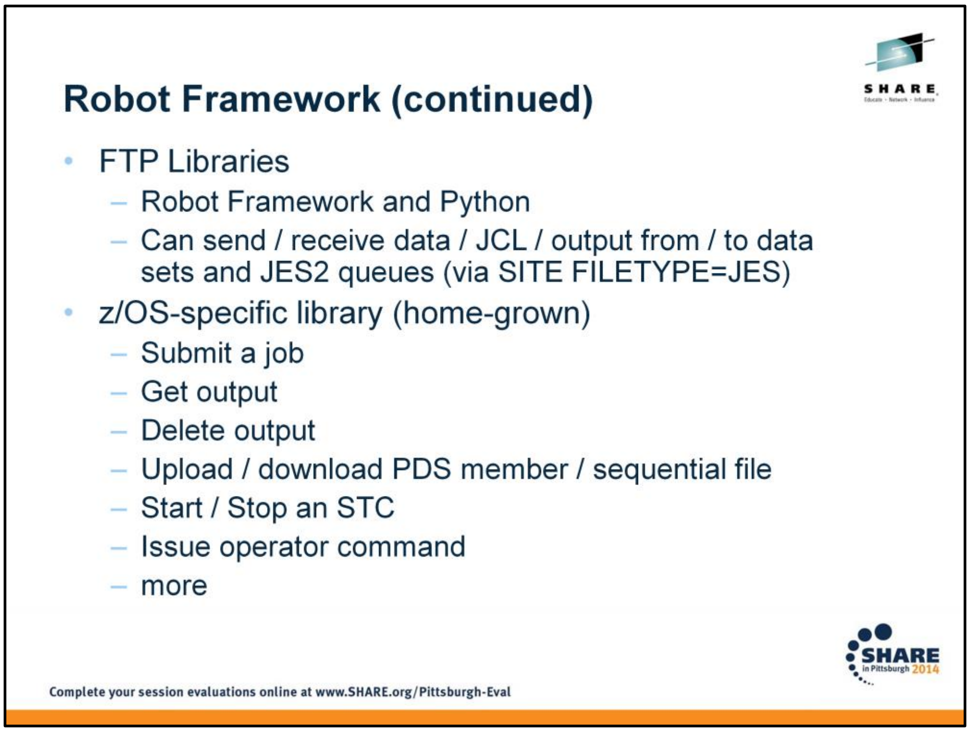
- Express tests in natural English
- Test suite
  - Test case
    - “Keywords”
      - Built in or home-grown
- Logging / Documentation / Debugging
- Libraries
  - Very versatile / extensible
- Integrates with Jenkins via plugin
- Active user community / forums

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

Tests should be understandable in non-technical terms – for example, the product owner should be able to tell by reading the names of the test cases what is being tested. Tests can theoretically be written by non-programmers.

“Keywords” in Robot Framework can be thought of as subroutines, methods, or modules. Do NOT think of them like “keyword=value”! They are really just steps to be performed.

Some of the libraries provided by Robot Framework include Builtin, OperatingSystem, Telnet, Collections, String, Dialogs, Remote, Eclipse, Selenium, even Android and iOS extensions!

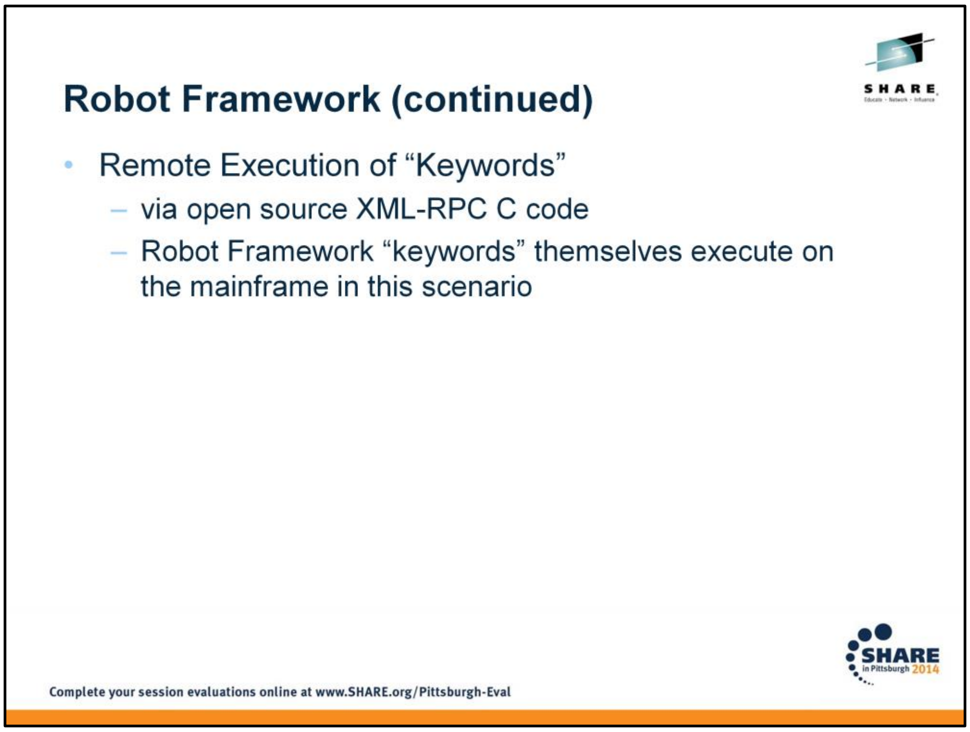


Robot Framework (continued)

- FTP Libraries
  - Robot Framework and Python
  - Can send / receive data / JCL / output from / to data sets and JES2 queues (via SITE FILETYPE=JES)
- z/OS-specific library (home-grown)
  - Submit a job
  - Get output
  - Delete output
  - Upload / download PDS member / sequential file
  - Start / Stop an STC
  - Issue operator command
  - more

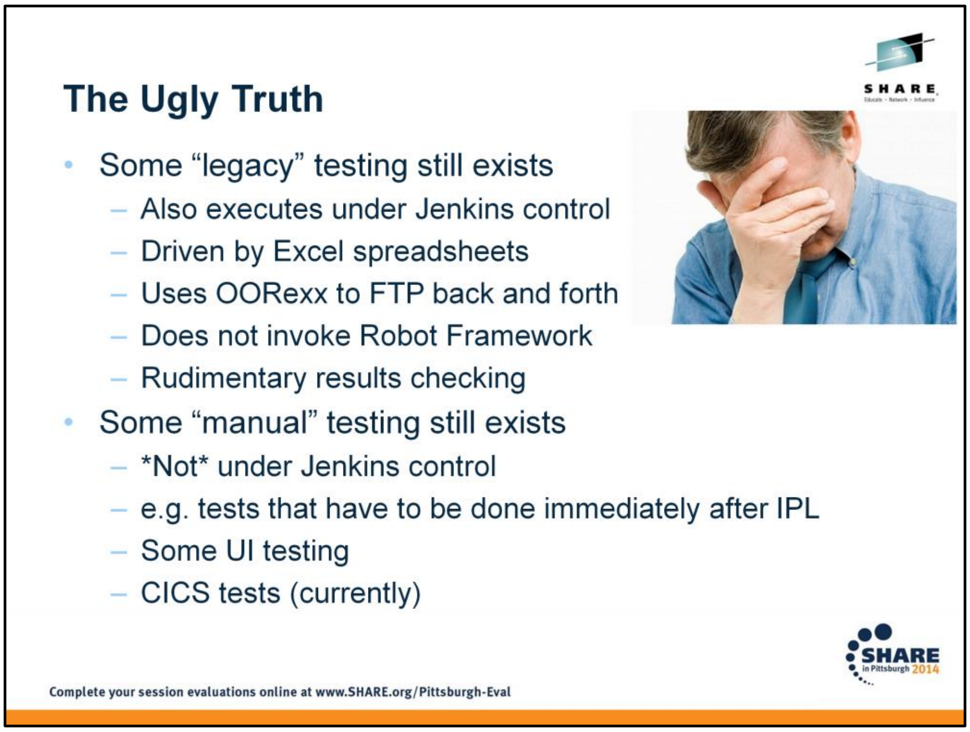
Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

The z/OS-specific library that we wrote is not publicly available. Is there interest in making it available?



Robot Framework (continued)

- Remote Execution of “Keywords”
  - via open source XML-RPC C code
  - Robot Framework “keywords” themselves execute on the mainframe in this scenario

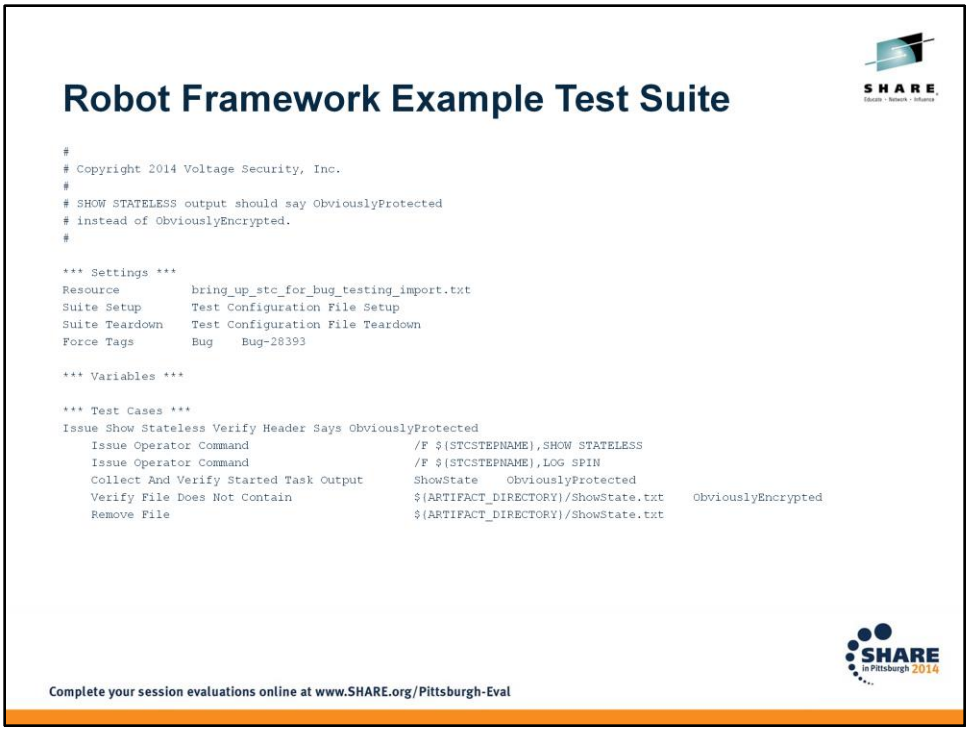


The Ugly Truth

- Some “legacy” testing still exists
  - Also executes under Jenkins control
  - Driven by Excel spreadsheets
  - Uses OORexx to FTP back and forth
  - Does not invoke Robot Framework
  - Rudimentary results checking
- Some “manual” testing still exists
  - *Not* under Jenkins control
  - e.g. tests that have to be done immediately after IPL
  - Some UI testing
  - CICS tests (currently)

Note that even if you’re not using a testing framework like Robot Framework, you can still take advantage of continuous integration.

You probably don’t want your system being IPL’ed under control of your continuous integration process – right? ☺

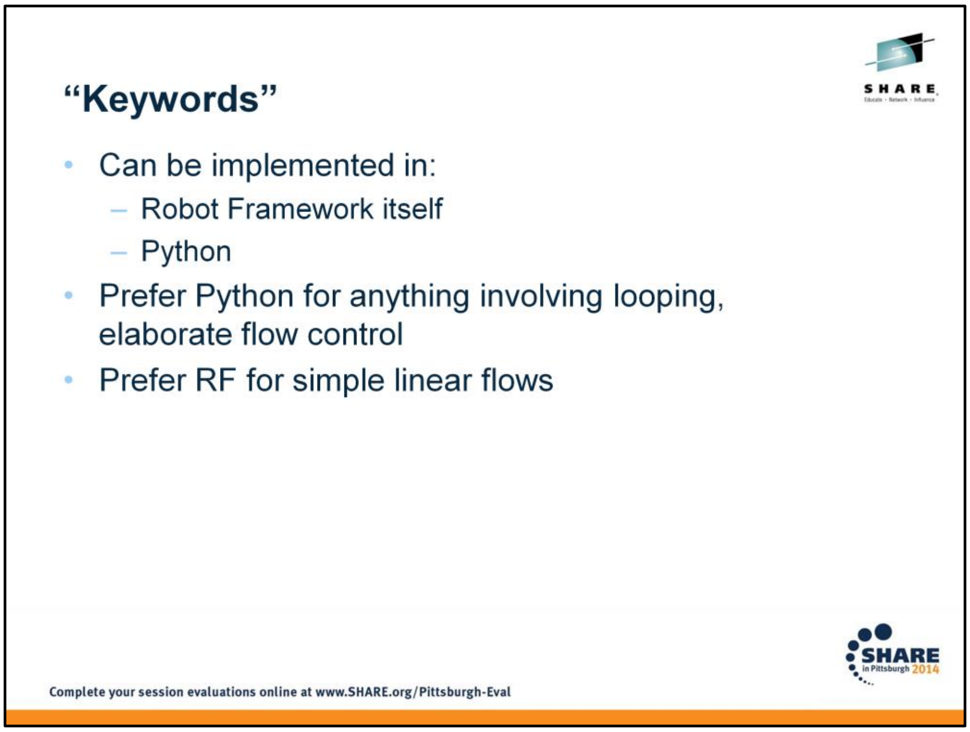


Robot Framework Example Test Suite

```
#
# Copyright 2014 Voltage Security, Inc.
#
# SHOW STATELESS output should say ObviouslyProtected
# instead of ObviouslyEncrypted.
#

*** Settings ***
Resource          bring_up_stc_for_bug_testing_import.txt
Suite Setup       Test Configuration File Setup
Suite Teardown    Test Configuration File Teardown
Force Tags        Bug    Bug-28393

*** Variables ***

*** Test Cases ***
Issue Show Stateless Verify Header Says ObviouslyProtected
    Issue Operator Command                       /F ${STCSTEPNAME},SHOW STATELESS
    Issue Operator Command                       /F ${STCSTEPNAME},LOG SPIN
    Collect And Verify Started Task Output       ShowState    ObviouslyProtected
    Verify File Does Not Contain                 ${ARTIFACT_DIRECTORY}/ShowState.txt    ObviouslyEncrypted
    Remove File                                  ${ARTIFACT_DIRECTORY}/ShowState.txt
```

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

The definition of a "test suite" is a bit odd in Robot Framework. Here we have a single file, consisting of a "suite" of <u>one</u> test case ("Issue Show Stateless Verify Header Says ObviouslyProtected"). You can (and typically do) have more than one test case in a single file (suite), but you can <u>also</u> run an entire directory of files (or a subset thereof) as a "suite", each file of which can consist of one or more test cases (and therefore you can essentially have "suites of suites"). The basic concepts apply, though, no matter how the suite is implemented.

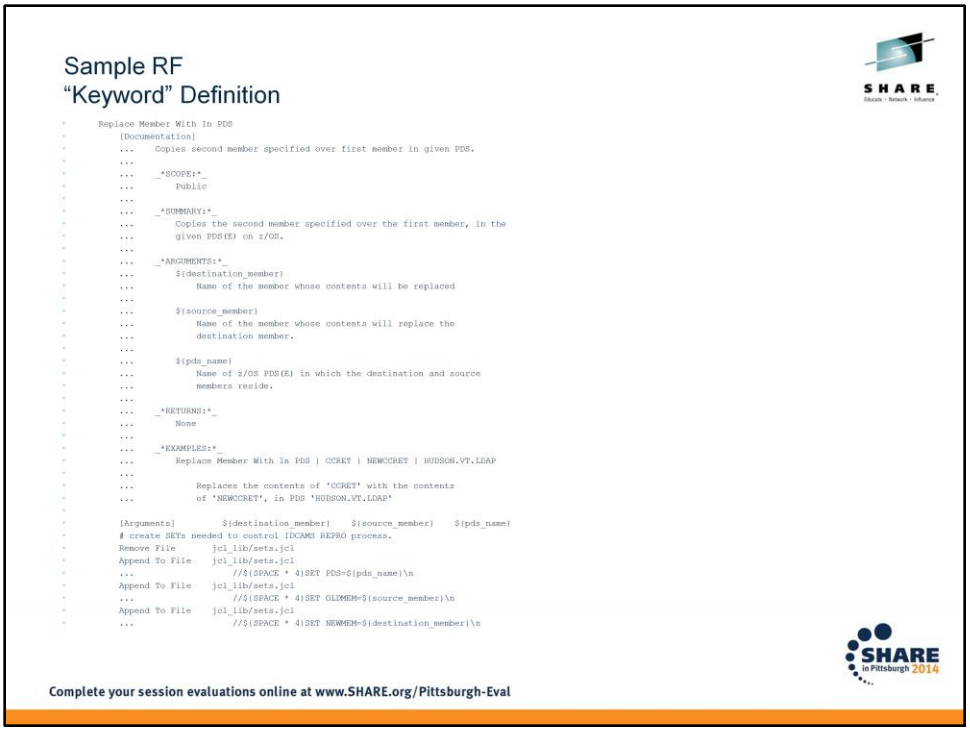


"Keywords"

- Can be implemented in:
  - Robot Framework itself
  - Python
- Prefer Python for anything involving looping, elaborate flow control
- Prefer RF for simple linear flows

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

There is some loop control provided directly in Robot Framework, but it's limited and hard to use. It can make the test implementations difficult to understand, too, which goes against one of the main points of using Robot Framework in the first place.

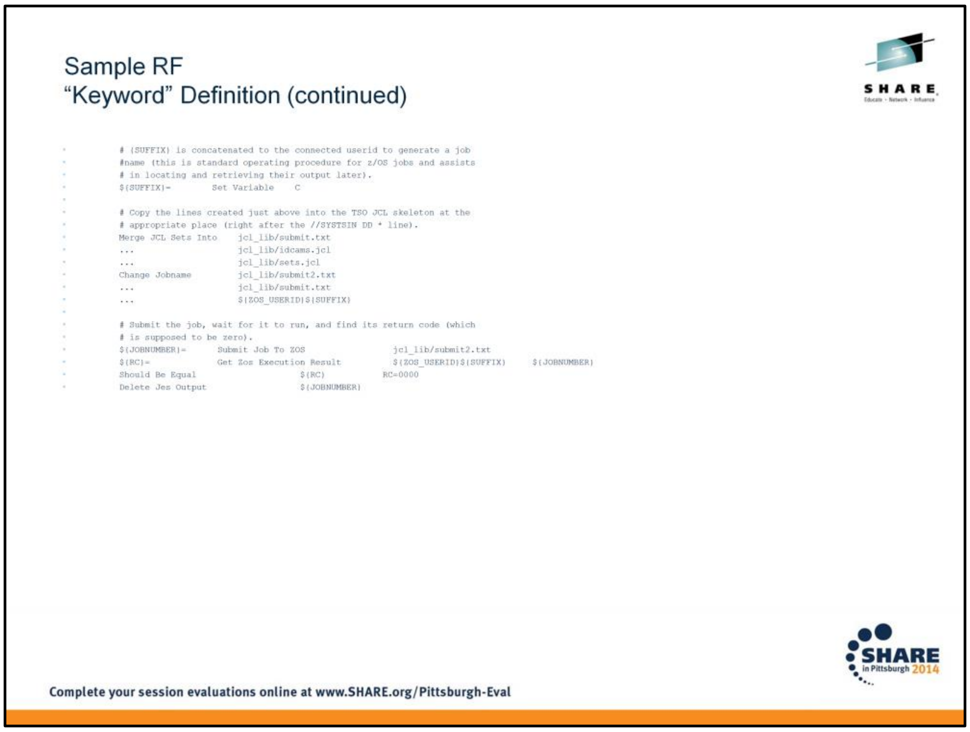


Sample RF “Keyword” Definition

```
Replace Member With In PDS
    [Documentation]
    ...    Copies second member specified over first member in given PDS.
    ...
    ...    _*SCOPE:*_
    ...        Public
    ...
    ...    _*SUMMARY:*_
    ...        Copies the second member specified over the first member, in the
    ...        given PDS(E) on z/OS.
    ...
    ...    _*ARGUMENTS:*_
    ...        ${destination_member}
    ...            Name of the member whose contents will be replaced
    ...
    ...        ${source_member}
    ...            Name of the member whose contents will replace the
    ...            destination member.
    ...
    ...        ${pds_name}
    ...            Name of z/OS PDS(E) in which the destination and source
    ...            members reside.
    ...
    ...    _*RETURNS:*_
    ...        None
    ...
    ...    _*EXAMPLES:*_
    ...        Replace Member With In PDS | CCRET | NEWCCRET | HUDSON.VT.LDAP
    ...
    ...            Replaces the contents of 'CCRET' with the contents
    ...            of 'NEWCCRET', in PDS 'HUDSON.VT.LDAP'

    [Arguments]         ${destination_member}    ${source_member}    ${pds_name}
    # create SETs needed to control IDCAMS REPRO process.
    Remove File       jcl_lib/sets.jcl
    Append To File    jcl_lib/sets.jcl
    ...                   //${SPACE * 4}SET PDS=${pds_name}\n
    Append To File    jcl_lib/sets.jcl
    ...                   //${SPACE * 4}SET OLDMEM=${source_member}\n
    Append To File    jcl_lib/sets.jcl
    ...                   //${SPACE * 4}SET NEWMEM=${destination_member}\n
```

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

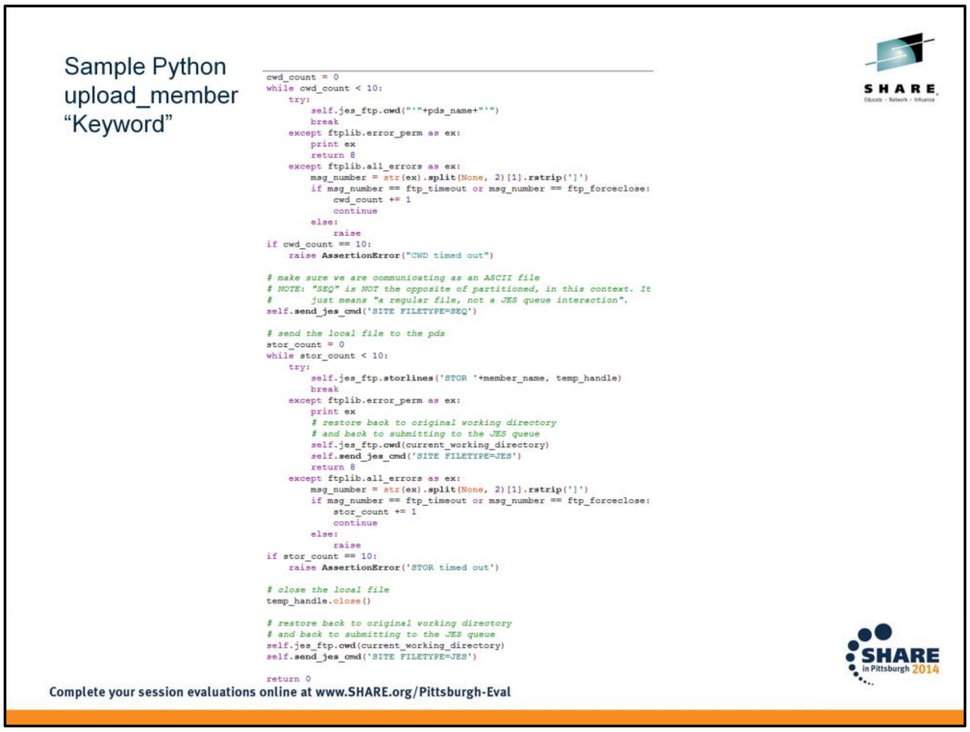
Keywords are implemented in terms of other keywords – all the way down until you get to Robot Framework-provided keywords or python-implemented home-grown keywords. This keyword starts by cleaning up any leftover file (with “Remove File”), populating it with JCL lines, including variable substitution (with “Append to File”), then… [next slide]



Sample RF "Keyword" Definition (continued)

```
# {SUFFIX} is concatenated to the connected userid to generate a job
#name (this is standard operating procedure for z/OS jobs and assists
# in locating and retrieving their output later).
${SUFFIX}=        Set Variable    C

# Copy the lines created just above into the TSO JCL skeleton at the
# appropriate place (right after the //SYSTSIN DD * line).
Merge JCL Sets Into    jcl_lib/submit.txt
...                    jcl_lib/idcams.jcl
...                    jcl_lib/sets.jcl
Change Jobname         jcl_lib/submit2.txt
...                    jcl_lib/submit.txt
...                    ${ZOS_USERID}${SUFFIX}

# Submit the job, wait for it to run, and find its return code (which
# is supposed to be zero).
${JOBNUMBER}=      Submit Job To ZOS                jcl_lib/submit2.txt
${RC}=             Get Zos Execution Result         ${ZOS_USERID}${SUFFIX}     ${JOBNUMBER}
Should Be Equal                   ${RC}            RC=0000
Delete Jes Output                 ${JOBNUMBER}
```

setting a variable (with "Set Variable" (built in)), merging the SET statements into existing base JCL, changing the JOB card, submitting the job to z/OS, acquiring the output from the JES queue, figuring out the highest return code, making sure it is "0000", and purging the output from the queue.

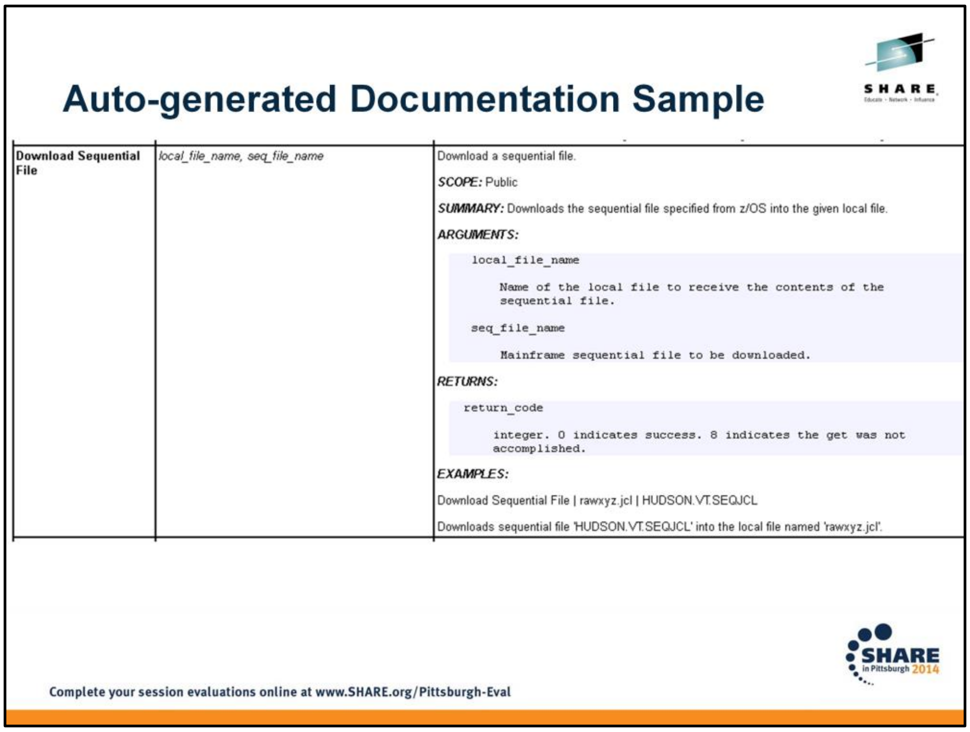


Sample Python upload_member "Keyword"

```
cwd_count = 0
while cwd_count < 10:
    try:
        self.jes_ftp.cwd("'"+pds_name+"'")
        break
    except ftplib.error_perm as ex:
        print ex
        return 8
    except ftplib.all_errors as ex:
        msg_number = str(ex).split(None, 2)[1].rstrip(']')
        if msg_number == ftp_timeout or msg_number == ftp_forceclose:
            cwd_count += 1
            continue
        else:
            raise
if cwd_count == 10:
    raise AssertionError("CWD timed out")

# make sure we are communicating as an ASCII file
# NOTE: "SEQ" is NOT the opposite of partitioned, in this context. It
#       just means "a regular file, not a JES queue interaction".
self.send_jes_cmd('SITE FILETYPE=SEQ')

# send the local file to the pds
stor_count = 0
while stor_count < 10:
    try:
        self.jes_ftp.storlines('STOR '+member_name, temp_handle)
        break
    except ftplib.error_perm as ex:
        print ex
        # restore back to original working directory
        # and back to submitting to the JES queue
        self.jes_ftp.cwd(current_working_directory)
        self.send_jes_cmd('SITE FILETYPE=JES')
        return 8
    except ftplib.all_errors as ex:
        msg_number = str(ex).split(None, 2)[1].rstrip(']')
        if msg_number == ftp_timeout or msg_number == ftp_forceclose:
            stor_count += 1
            continue
        else:
            raise
if stor_count == 10:
    raise AssertionError('STOR timed out')

# close the local file
temp_handle.close()

# restore back to original working directory
# and back to submitting to the JES queue
self.jes_ftp.cwd(current_working_directory)
self.send_jes_cmd('SITE FILETYPE=JES')

return 0
```

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

Here's an example of a home-grown keyword that we chose to implement in Python, giving us more control over exception handling and looping to re-try on connection failures / timeouts from FTP.



Auto-generated Documentation Sample

Download Sequential File	*local_file_name, seq_file_name*	Download a sequential file. ***SCOPE:*** Public ***SUMMARY:*** Downloads the sequential file specified from z/OS into the given local file. ***ARGUMENTS:***

```
  local_file_name

      Name of the local file to receive the contents of the
      sequential file.

  seq_file_name

      Mainframe sequential file to be downloaded.
```

RETURNS:

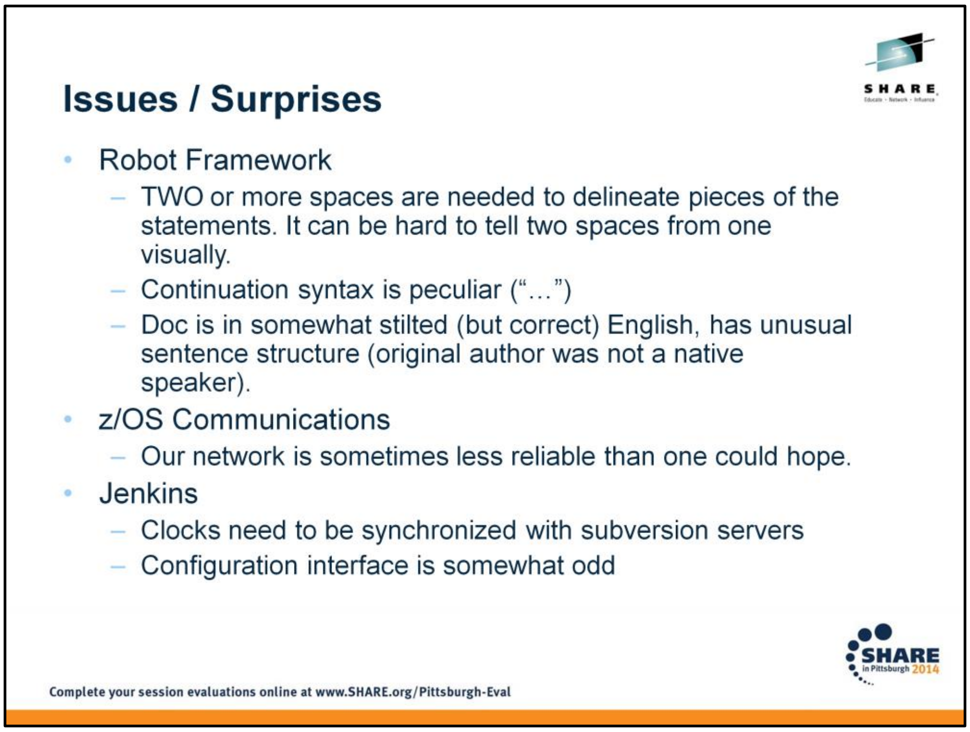
```
  return_code

      integer. 0 indicates success. 8 indicates the get was not
      accomplished.
```

EXAMPLES:

Download Sequential File | rawxyz.jcl | HUDSON.VT.SEQJCL

Downloads sequential file 'HUDSON.VT.SEQJCL' into the local file named 'rawxyz.jcl'.

This is the output from the “libdoc” command, which reads either Python or Robot Framework keyword definitions and pulls out the formatted documentation to create html.

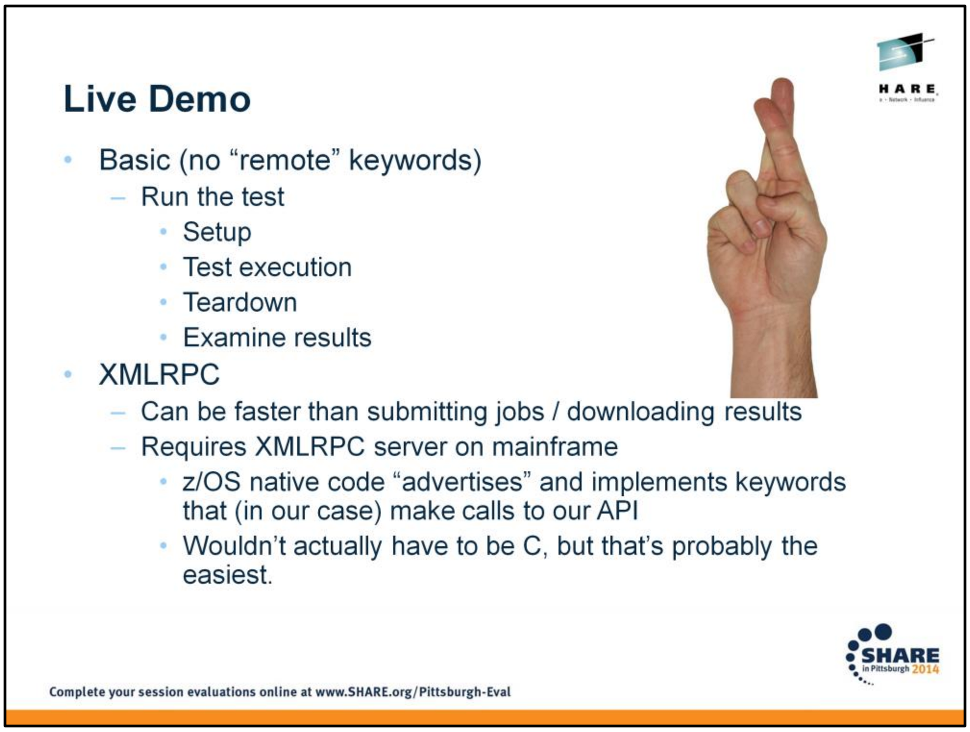


Issues / Surprises

- Robot Framework
  - TWO or more spaces are needed to delineate pieces of the statements. It can be hard to tell two spaces from one visually.
  - Continuation syntax is peculiar ("…")
  - Doc is in somewhat stilted (but correct) English, has unusual sentence structure (original author was not a native speaker).
- z/OS Communications
  - Our network is sometimes less reliable than one could hope.
- Jenkins
  - Clocks need to be synchronized with subversion servers
  - Configuration interface is somewhat odd

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

We recommend using at least 4 spaces to separate keywords from their parameters. Much easier to see the pieces.
Robot doesn't have to use multiple spaces to separate parameters from keyword names. You can use vertical bars. It's all really tables and cells when you get right down to it.
Coping with odd ways to break lines into fields and odd continuation should be old hat for JCL folks!
Important to build in toleration of timeouts and dropped connections with a limited number of retries to avoid needless restarts.
If Jenkins and subversion clocks drift apart, you can get situations where you commit something to svn but Jenkins doesn't notice it and kick off a build.



Live Demo

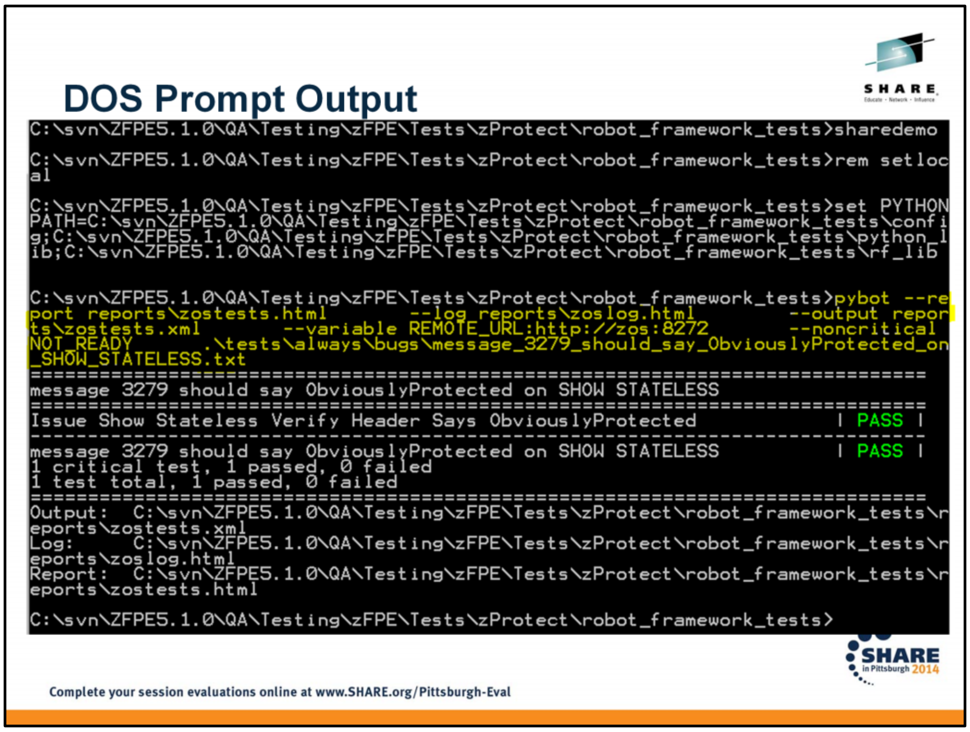
- Basic (no "remote" keywords)
  - Run the test
    - Setup
    - Test execution
    - Teardown
    - Examine results
- XMLRPC
  - Can be faster than submitting jobs / downloading results
  - Requires XMLRPC server on mainframe
    - z/OS native code "advertises" and implements keywords that (in our case) make calls to our API
    - Wouldn't actually have to be C, but that's probably the easiest.

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

SHARE in Pittsburgh 2014

Since our started tasks expose an API anyway, it's faster for us to just make that API accessible directly from the client machine than to create a job, submit it, and download the results. This is done with the XMLRPC server.
For the XMLRPC server, we started with an open source implementation in C. There was some work required due to code page issues (in some places, for example, it was checking for certain characters using their putative hexadecimal values – which are of course different between ASCII and EBCDIC).

Python will probably never be ported to z/OS native: Guido van Rossum (Python's "Benevolent Dictator for Life"): "How important is z/OS? I'm very skeptical of the viability of any OS that uses an encoding that is not a superset of ASCII." (that's from 2007!)



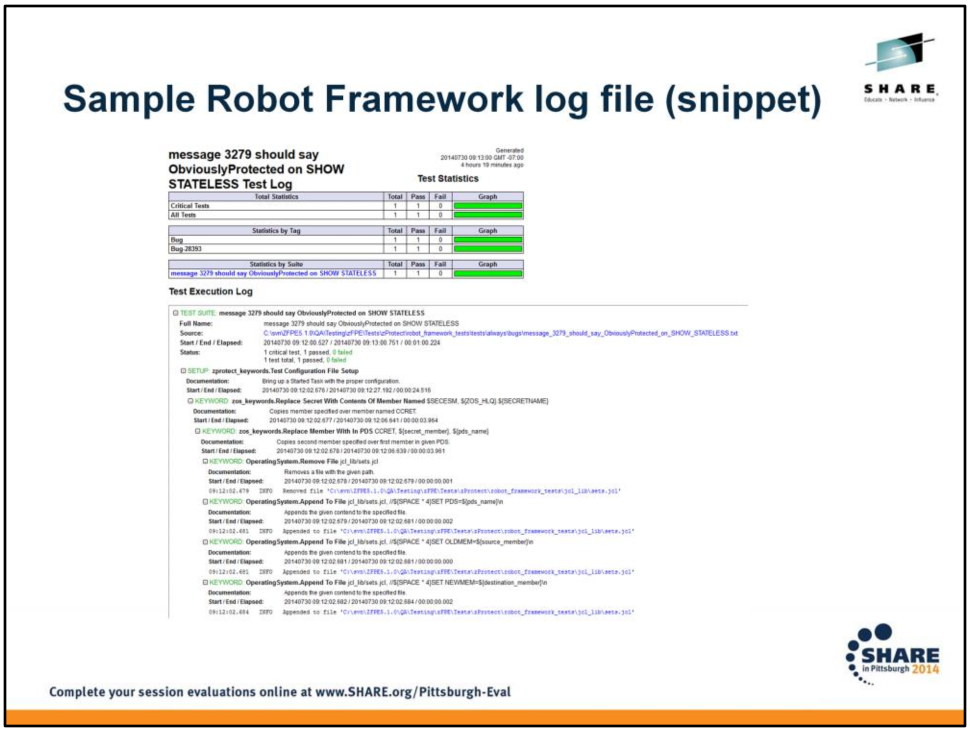
DOS Prompt Output

```
C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests>sharedemo

C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests>rem setloc
al

C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests>set PYTHON
PATH=C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\confi
g;C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\python_l
ib;C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\rf_lib

C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests>pybot --re
port reports\zostests.html       --log reports\zoslog.html        --output repor
ts\zostests.xml        --variable REMOTE_URL:http://zos:8272       --noncritical
NOT_READY       .\tests\always\bugs\message_3279_should_say_ObviouslyProtected_on
_SHOW_STATELESS.txt
==============================================================================
message 3279 should say ObviouslyProtected on SHOW STATELESS
==============================================================================
Issue Show Stateless Verify Header Says ObviouslyProtected            | PASS |
------------------------------------------------------------------------------
message 3279 should say ObviouslyProtected on SHOW STATELESS          | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
==============================================================================
Output:  C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\r
eports\zostests.xml
Log:     C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\r
eports\zoslog.html
Report:  C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\r
eports\zostests.html

C:\svn\ZFPE5.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests>
```

SHARE in Pittsburgh 2014

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

What I highlighted in yellow above is the actual command that is issued to perform the Robot Framework test. We wrap a .bat file around it.



Sample Robot Framework log file (snippet)

message 3279 should say ObviouslyProtected on SHOW STATELESS Test Log

Generated
20140730 09:13:00 GMT -07:00
4 hours 19 minutes ago

Test Statistics

Total Statistics	Total	Pass	Fail	Graph
Critical Tests	1	1	0	🟩
All Tests	1	1	0	🟩

Statistics by Tag	Total	Pass	Fail	Graph
Bug	1	1	0	🟩
Bug-28393	1	1	0	🟩

Statistics by Suite	Total	Pass	Fail	Graph
message 3279 should say ObviouslyProtected on SHOW STATELESS	1	1	0	🟩

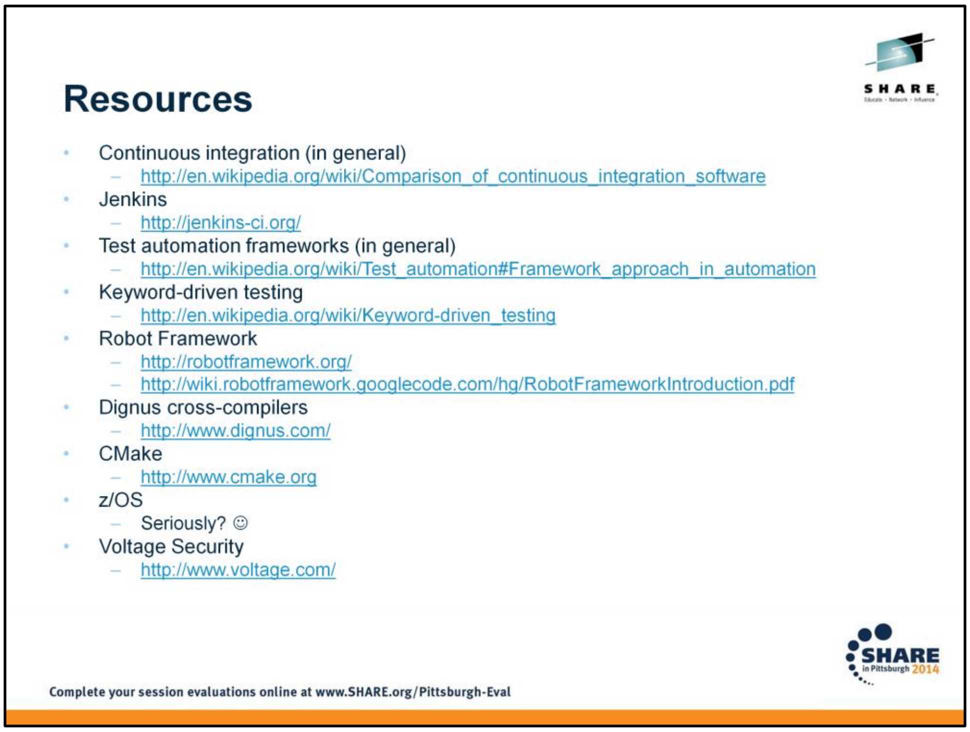
Test Execution Log

```
⊟ TEST SUITE: message 3279 should say ObviouslyProtected on SHOW STATELESS
  Full Name:             message 3279 should say ObviouslyProtected on SHOW STATELESS
  Source:                C:\svn\ZFPES 1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\tests\always\bugs\message_3279_should_say_ObviouslyProtected_on_SHOW_STATELESS.txt
  Start / End / Elapsed: 20140730 09:12:00.527 / 20140730 09:13:00.751 / 00:01:00.224
  Status:                1 critical test, 1 passed, 0 failed
                         1 test total, 1 passed, 0 failed
  ⊟ SETUP: zprotect_keywords.Test Configuration File Setup
    Documentation:         Bring up a Started Task with the proper configuration.
    Start / End / Elapsed: 20140730 09:12:02.676 / 20140730 09:12:27.192 / 00:00:24.516
    ⊟ KEYWORD: zos_keywords.Replace Secret With Contents Of Member Named $SECESM, ${ZOS_HLQ}.${SECRETNAME}
      Documentation:         Copies member specified over member named CCRET.
      Start / End / Elapsed: 20140730 09:12:02.677 / 20140730 09:12:06.641 / 00:00:03.964
      ⊟ KEYWORD: zos_keywords.Replace Member With In PDS CCRET, ${secret_member}, ${pds_name}
        Documentation:         Copies second member specified over first member in given PDS.
        Start / End / Elapsed: 20140730 09:12:02.678 / 20140730 09:12:06.639 / 00:00:03.961
        ⊟ KEYWORD: OperatingSystem.Remove File jcl_lib/sets.jcl
          Documentation:         Removes a file with the given path.
          Start / End / Elapsed: 20140730 09:12:02.678 / 20140730 09:12:02.679 / 00:00:00.001
          09:12:02.679  INFO  Removed file 'C:\svn\ZFPES.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\jcl_lib\sets.jcl'
        ⊟ KEYWORD: OperatingSystem.Append To File jcl_lib/sets.jcl, //${SPACE * 4}SET PDS=${pds_name}\n
          Documentation:         Appends the given contend to the specified file.
          Start / End / Elapsed: 20140730 09:12:02.679 / 20140730 09:12:02.681 / 00:00:00.002
          09:12:02.681  INFO  Appended to file 'C:\svn\ZFPES.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\jcl_lib\sets.jcl'
        ⊟ KEYWORD: OperatingSystem.Append To File jcl_lib/sets.jcl, //${SPACE * 4}SET OLDMEM=${source_member}\n
          Documentation:         Appends the given contend to the specified file.
          Start / End / Elapsed: 20140730 09:12:02.681 / 20140730 09:12:02.681 / 00:00:00.000
          09:12:02.681  INFO  Appended to file 'C:\svn\ZFPES.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\jcl_lib\sets.jcl'
        ⊟ KEYWORD: OperatingSystem.Append To File jcl_lib/sets.jcl, //${SPACE * 4}SET NEWMEM=${destination_member}\n
          Documentation:         Appends the given contend to the specified file.
          Start / End / Elapsed: 20140730 09:12:02.682 / 20140730 09:12:02.684 / 00:00:00.002
          09:12:02.684  INFO  Appended to file 'C:\svn\ZFPES.1.0\QA\Testing\zFPE\Tests\zProtect\robot_framework_tests\jcl_lib\sets.jcl'
```

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

Kind of an eye test, I know – but in the pdf it's a little more legible. We just saw this in the live demo; this is just a screenshot from that.

Before I put up the final slide, are there any questions?



Resources

- Continuous integration (in general)
  - http://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software
- Jenkins
  - http://jenkins-ci.org/
- Test automation frameworks (in general)
  - http://en.wikipedia.org/wiki/Test_automation#Framework_approach_in_automation
- Keyword-driven testing
  - http://en.wikipedia.org/wiki/Keyword-driven_testing
- Robot Framework
  - http://robotframework.org/
  - http://wiki.robotframework.googlecode.com/hg/RobotFrameworkIntroduction.pdf
- Dignus cross-compilers
  - http://www.dignus.com/
- CMake
  - http://www.cmake.org
- z/OS
  - Seriously? ☺
- Voltage Security
  - http://www.voltage.com/

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval