#SHAREorg

SHARE

Managing Server Component Output from WebSphere Application Server on z/OS

A WebSphere on z/OS exclusive!

Mike Loos
IBM
mikeloos@us.ibm.com

Session number 13600
Monday, August 12, 2013
4:30 PM

SHARE in Boston

WebSphere Application Server on z/OS Sessions in Boston

Day	Time	Room	#	Title	Speaker
Monday	9:30	203	13597	Getting Started with WebSphere Liberty Profile on z/OS	David Follis
Monday	4:30	203	13600	Managing Server Output from WAS on z/OS	Mike Loos
Tuesday	9:30	203	13644	Using WAS Optimized Local Adapters (WOLA) to migrate your COBOL to zAAP-able Java	Jim Mulvey
Tuesday	11:00	203	13640	Need A Support Assistant? Check Out IBM's! (ISA)	Mike Stephen
Tuesday	3:00	203	13641	zWAS: In Real Life	Rod Feak
Wednesday	1:30	202	13601	Lab: WebSphere Liberty Profile on z/OS	everybody
Thursday	11:00	203	13598	Getting Started with Compute Grid (Batch)	John Hutchinson
Thursday	3:00	203	13645	Configuring Security for Liberty	Mike Loos

What is this all about?

- Why another form of output?
- What does this one do that's different?
- How do you set it up?
  - In WebSphere...
  - Security
  - The HTTP Server (V7 – V8.5)...
- Demo.
- Q and A?

3

This presentation, which is based on the techdoc titled “Implementing the Output APAR (PM74923) enhancements in WebSphere Application Server on z/OS”, will describe a new method of providing output to both z/OS and non-z/OS based folks.
The historical output solutions will be described briefly, then this new methodology will be described in more detail.
How to set this up in WebSphere on z/OS, how to set up security covering the output, and how to set up the http server to allow browsing of the output and limit it to the appropriate viewers will be covered.
We'll do a short demonstration of a configuration that implements this method, and hopefully still allow for some Q and A.

What is available now?

- SYSOUT.
  - Great for z/OS familiar folks!
  - For others, not so much...
- SYSOUT redirected to the file system.
  - Naming and sizing limitations.
- HPEL.
  - Platform agnostic.
  - Some nice operational features.
    - *File switching and archiving on size, date/time, number of files.*
    - *End user may find working with it a little clumsy at times...*
- This new thing...

Since early versions of WebSphere, the ability to allow non-z/OS folks access to the WebSphere on z/OS output (stdout and stderr), has presented a bit of a challenge. The stdout stream is available in the z/OS SYSPRINT DD and the stderr stream is available in the z/OS SYSOUT DD, both of which default to spool output.
The first iteration of a solution, involved changing the DD statements to point to the file system, while allowing the access desired, has some limitations in that the names are set in the DD statement, and the file, since basically you can only switch it by restarting the server, will continue to grow in size as long as the server is up. Also, there was the possibility of file naming conflicts if two servants were started at the same time.
The next major solution was the introduction of the High Performance Extensible Logging (HPEL) option in V8. It has the advantage of being platform agnostic, and has some nice operational features, such as file switching and archiving based on the size of the file, date and/or time, and limits on the number of files.
Since it is a binary file type, it does require that you use a provided interface to view the logs, either the viewer provided within the adminconsole, or the logViewer.sh script. Some folks might view this interface as a little less than elegent.
So finally we come to this new thing...

The New(est) Method

- File based.
- WebSphere ensures that there will be no naming conflicts.
- Works for all components (Daemon, Dmgr, Nodeagents, Servers (both controllers, adjuncts, and servants)).
- File switching is simple using a z/OS MODIFY command.
- No need for users to access the filesystem.
- Access to the output can be uncontrolled or controlled...
  - At the cell level.
  - At the node.
  - Server by server.
  - Via any security system that the HTTP server supports (SAF, LDAP, etc.).

5

This “new” implementation is file based, much the same as the redirection via the SYSOUT DD cards. But in this case, WebSphere ensures that there will be no naming conflicts.
This works with, and is granular enough to allow seperate security over, the output of all WebSphere on z/OS components (Daemons, the Deployment Manager (controller and servant), nodeagents, and servers (controller, adjunct, and servants).
Switching to a new file is accomplished by either a component restart, or a simple z/OS Modify command. The use of the Modify command interface allows control by normal z/OS automation procedures.
There is no need for viewers of the output to access the filesystem directly, so the viewers of the output can be restricted from reading, writing, and/or other modification of the files in the filesystem.
Access to the ouput may be either completely uncontrolled, or it may be controllled at the cell level, the node level, or the server level.
The controlling security may be any security provider supported by the http server, SAF (RACF, other vendor solutions) or LDAP.

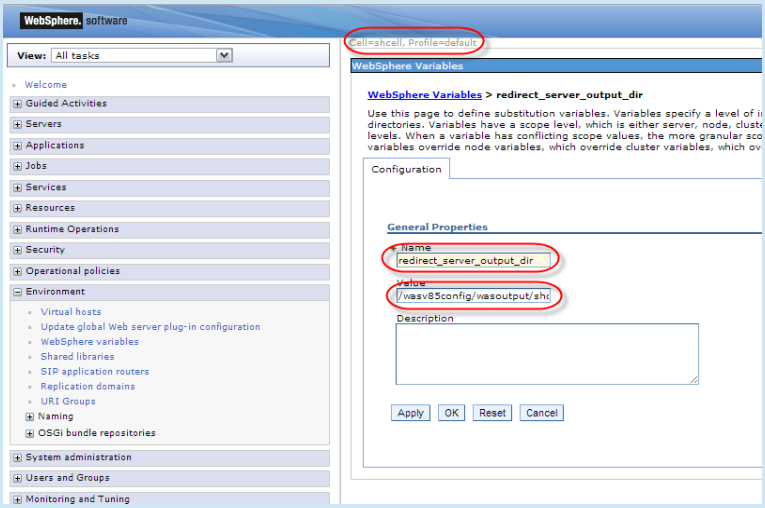
How does it work? What do I do to set it up?

- In WebSphere:
  - Add variables at appropriate scopes.
    - *They will be inherited by lower levels...*
      - *Simplest setup is to just add them at the cell level and let all components write to the same path.*
      - *The same variable at a lower level will take precedence.*
    - *Variable names:*
      - ***DAEMON_redirect_server_output_dir*** *(for the Daemon).*
      - ***redirect_server_output_dir*** *(for everything else).*
      - *Value is simply the path name where you wish the output to be written.*
      - *Ex.* ***/wasv85config/wasoutput/shcell/shcell***

6

This is all setup rather simply. In the WebSphere on z/OS cell, you simply add variables at appropriate scopes (cell, node, server). The variables will be inherited by lower (more granular) scope levels, unless overridden at the lower scope level.
The most simple setup is to add the variables at the cell level and allow all components to write to the same path. That also effectively limits to security to a single level, which may be adequate for many cells.
There are two variable names, DAEMON_redirect_server_output_dir and redirect_server_output_dir for the daemon and other components respectively. The value associated with the variable is simply the path name to which you wish the output to be written.

Setup in WebSphere.



That's all there is! ...in WebSphere.

This is what it looks like when you are adding the variable in the adminconsole.

How does it work? What do I do to set it up?

- In WebSphere:
  - All that is left is to restart the components you believe that you have modified, and the output will be redirected to the path you've specified.

```
:/shared/wasoutput/shcell/shcell
-> ls
SHCELL.SHDMNODE.SHDMGR.SHDMGR.STC00160.CTL.130528.101731.SYSOUT.txt
SHCELL.SHDMNODE.SHDMGR.SHDMGR.STC00160.CTL.130528.101731.SYSPRINT.txt
SHCELL.SHDMNODE.SHDMGR.SHDMGRS.STC00165.SR.130528.101752.SYSOUT.txt
SHCELL.SHDMNODE.SHDMGR.SHDMGRS.STC00165.SR.130528.101752.SYSPRINT.txt
SHCELL.SHDMNODE.SYSC.SHDEMN.STC00163.DAEMON.130528.141731.SYSOUT.txt
SHCELL.SHDMNODE.SYSC.SHDEMN.STC00163.DAEMON.130528.141731.SYSPRINT.txt
SHCELL.SHNODEC.SHAGNTC.SHAGNTC.STC00167.CTL.130528.101853.SYSOUT.txt
SHCELL.SHNODEC.SHAGNTC.SHAGNTC.STC00167.CTL.130528.101853.SYSPRINT.txt
SHCELL.SHNODED.SHAGNTD.SHAGNTD.STC00168.CTL.130528.101927.SYSOUT.txt
SHCELL.SHNODED.SHAGNTD.SHAGNTD.STC00168.CTL.130528.101927.SYSPRINT.txt
SHCELL.SHNODED.SYSD.SHDEMN.STC00169.DAEMON.130528.141928.SYSOUT.txt
SHCELL.SHNODED.SYSD.SHDEMN.STC00169.DAEMON.130528.141928.SYSPRINT.txt
```


After the variable(s) have been added, all you have to do to make it take effect is to restart the component(s) you believe that you have covered with the variable(s) and the output will be redirected to the location to which the path name specified points.
The files are named, for all components except the daemon(s) by concatenating the cell short name, node short name, server short name, job name, started task id, creation date, creation time, output type (SYSOUT for stderr and SYSPRINT for stdout), and the constant “txt” indicating the output type. The daemon output is named slightly differently, cell short name, node short name, system id, job name, started tast id, the constant “DAEMON”, creation date, creation time, output type, and the constant “txt”.

How does it work? What do I do to set it up?

- If you wish to switch to a new file, from any z/OS Console:

```
F SHDEMN,ROLL_LOGS

BBOO0211I MODIFY COMMAND ROLL_LOGS COMPLETED SUCCESSFULLY
```

```
 :/shared/wasoutput/shcell/shcell
 -> ls
 SHCELL.SHDMNODE.SHDMGR.SHDMGR.STC00160.CTL.130528.101731.SYSOUT.txt
 SHCELL.SHDMNODE.SHDMGR.SHDMGR.STC00160.CTL.130528.101731.SYSPRINT.txt
 SHCELL.SHDMNODE.SHDMGR.SHDMGRS.STC00165.SR.130528.101752.SYSOUT.txt
 SHCELL.SHDMNODE.SHDMGR.SHDMGRS.STC00165.SR.130528.101752.SYSPRINT.txt
 SHCELL.SHDMNODE.SYSC.SHDEMN.STC00163.DAEMON.130528.141731.SYSOUT.txt
 SHCELL.SHDMNODE.SYSC.SHDEMN.STC00163.DAEMON.130528.141731.SYSPRINT.txt
 SHCELL.SHDMNODE.SYSC.SHDEMN.STC00163.DAEMON.130529.192332.SYSOUT.txt
 SHCELL.SHDMNODE.SYSC.SHDEMN.STC00163.DAEMON.130529.192332.SYSPRINT.txt
 SHCELL.SHNODEC.SHAGNTC.SHAGNTC.STC00167.CTL.130528.101853.SYSOUT.txt
 SHCELL.SHNODEC.SHAGNTC.SHAGNTC.STC00167.CTL.130528.101853.SYSPRINT.txt
 SHCELL.SHNODED.SHAGNTD.SHAGNTD.STC00168.CTL.130528.101927.SYSOUT.txt
 SHCELL.SHNODED.SHAGNTD.SHAGNTD.STC00168.CTL.130528.101927.SYSPRINT.txt
 SHCELL.SHNODED.SYSD.SHDEMN.STC00169.DAEMON.130528.141928.SYSOUT.txt
 SHCELL.SHNODED.SYSD.SHDEMN.STC00169.DAEMON.130528.141928.SYSPRINT.txt
```

- At this point, if all you want is ISPF browse or telnet access, you are done...But if you want to make it really cool...

9

Switching a component to use a new file, without a component restart, is as simple as issuing a z/OS Modify command for the component, with the operand roll_logs. For demonstration purposes, the command was issued for the daemon on SYSC. The system responds with the BBOO0211I message, indicating successful completion. Examination of the output path location, shows that the daemon in question now has another set of files with a newer timestamp.
If you were to examine the specific output files, you would find that there would be a message (the last message in the file) in the “old” file with a forward pointer to the new file name, and a message in the new file (first message) that has a backward pointer to the previous file. So you could concievably write a script to chain all of the files together in proper order if that were your desire.

0

How does it work? What do I do to set it up?

- **Security setup:**
  - For the HTTP Server (we're going to build later).
    - *RDEF STARTED SHHTTP01.* STDATA(USER(SHADMIN) GROUP(SHCFG)TRACE(YES))*
    - *SETR RACLIST(STARTED) REFRESH*
    - PERMIT BPX.SERVER CLASS(FACILITY) ID(SHCFG) AC(READ)
    - SETR RACLIST(FACILITY) REFRESH
  - For the access controls we'll be implementing.
    - **AG SHCELL OMVS(AUTOGID) /* Entire cell access */**
    - AG SHDMNODE OMVS(AUTOGID) /* Entire dmgr node access */
    - AG SHNODEC OMVS(AUTOGID) /* Entire shnodec node access */
    - AG SHNODED OMVS(AUTOGID) /* Entire shnoded node access */
    - AG SHSR01C OMVS(AUTOGID) /* Access to shsr01c only */
    - AG SHSR01D OMVS(AUTOGID) /* Access to shsr01d only */
    - AG SHSR02C OMVS(AUTOGID) /* Access to shsr02c only */
    - AG SHSR02D OMVS(AUTOGID) /* Access to shsr02d only */

There are two parts to setting up security for this process, both of which are optional. Everything will work fine if you want it to be unsecured.
The first thing necessary if security is desired, is to set up the security for the http server which we'll be building a little later in this presentation. You'll need a STARTED class definition which allows the new http server to run with the cell's admin id and configuration group. We are considering the http server to be basically just another server in the cell, so this is appropriate. It also allows the http server access to all of the files which the cell's components create. The http server also needs read access to the FACILITY class profile BPX.SERVER.
The second set of security definitions are used to allow access to the output from the various components. The various groups defined are subsequently named, either singly or in lists of groups, in access limiting statements in the http server's configuration file. The groups **MUST** have an OMVS group id, whether it is specifically specified or automatically generated.
The http server which will be built as part of this presentation will only use the first, cell level, group, but there will be info on how to use the other groups to limit access on a more granular basis.

How does it work? What do I do to set it up?

- **Building the HTTP server.**
  - From either OMVS, telnet, or SSH... (Can't do this from ISHELL or 3.17):

```
cd /usr/lpp/IHSA/V8R5BASE/bin

./install_ihs /wasv85config/shcell/shhttp01 9898
```

The install command will respond with the following command, which you then enter, unchanged (cut and paste).

```
./shared/IHSA/V8R5BASE/bin/postinst -i
/wasv85config/shcell/shhttp01

-t install -v PORT=9898 -v
SERVERNAME=wsc3.washington.ibm.com

chown -Rh shadmin:shcfg /wasv85config/shcell/shhttp01
```

  - The HTTP server is now built and ready for appropriate modifications.

11

It is time to build the http server. This is the http server for z/OS based upon Apache, not the older Domino Go Webserver (DGW). It is delivered as part of WebSphere on z/OS, and can also be obtained as part of the ported tools package, although the one delivered with WebSphere on z/OS is the most current version.
To start, you have to run the install from an interactive shell (not ISHELL or ISPF 3.17), so from telnet, ssh, or OMVS, you change directory to the bin directory of the delivered server code.
Issue the `install_hfs` command with target install location, `/wasv85config/shcell/shhttp01` and the port number upon which the http server will listen, `9898` as arguments.
The command will respond with another command as output text, which you simply cut and paste into the command prompt and issue.
Next, issue a chown command to change the ownership of the files and directories in the target location to the cell's admin id and config group.
That's all there is to building the server. It would start as is, (but not do much of anything we'd call useful).

How does it work? What do I do to set it up?

- **Modifications to the httpd.conf file to make all of this work.**
  - httpd.conf is found in `<server root>/conf/` or in our case, `/wasv85config/shcell/shhttp01/conf`.
  - Add or modify the following statements using your favorite editor...

The following LoadModule statements must be added or uncommented as appropriate.

```
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule authnz_saf_module modules/mod_authnz_saf.so
LoadModule authz_default_module modules/mod_authz_default.so
```

If SSL support is required, then the following LoadModule statement must be uncommented.

```
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
```


Some modifications now must be done to the configuration file for the http server, found in the target location `/conf directory`, the `httpd.conf` file.
The `LoadModule` statements that provide support for basic authentication, SAF support, and z/OS authorization, and if required SSL support, must be either uncommented or if necessary added.

How does it work? What do I do to set it up?

- **Continuing...**

  The DocumentRoot directive should be changed to point at the base output location.

  ```
  DocumentRoot "/wasv85config/wasoutput/shcell"
  ```

  The Directory statement should be changed from where it points by default, `/wasv85config/shcell/shhttp01/htdocs` to be the same as the DocumentRoot directive.

  So change:

  From:

  ```
  <Directory "/wasv85config/shcell/shhttp01/htdocs">
  ```

  To:

  ```
  <Directory "/wasv85config/wasoutput/shcell">
  ```


13

The `DocumentRoot` statement mus tbe updated to point to the location which we have specified as the output location for the cell. In our case this location is a symbolic link which points to the actual output location.
Also, the `Directory` statement must be changed to the same location.

How does it work? What do I do to set it up?

- **Continuing...**

  Add the following statements (immediately after the previous Directory statement is fine...).

  ```
  SetOutputFilter DEFLATE
  CharsetSourceEnc IBM-1047
  CharsetDefault ISO8859-1
  ```

  Change the options directive:

  From:

  ```
  Options FollowSymLinks
  ```

  To:

  ```
  Options FollowSymLinks Indexes
  IndexOptions NameWidth=80
  ```


14

Immediately after the `Directory` statement which was just modified, the following statements should be added to support the fine encoding of the output files (EBCDIC).
The `Options` directive must also be changed to support the lengthy file names.

How does it work? What do I do to set it up?

- **Continuing...**

  The following two changes are to support the suppression of an error message that Internet Explorer will throw when it sees a txt file with parts of verbose garbage collection data. These two changes suppress that behavior.

```
# Uncomment the next line to allow MSIE verbose GC error
suppression.

LoadModule headers_module modules/mod_headers.so
```

  And:

```
# Added the following to suppress MSIE from throwing error on
verbose GC data.

<Files *.txt>

Header set X-Content-Type-Options nosniff

</Files>
```


15

The output in the stderr stream (SYSOUT) commonly contains the output from the verbose garbage collection process of the Java virtual machine. This is, in fact, statements which belong in an xml file. Since Internet Explorer may be the browser used to view these files, you should include the following statements to prevent Internet Explorer from deciding that the files you wish to view are malformed xml files, which it will then refuse to allow the user to access.
The two statements necessary are the `LoadModule` statement that supports file headers and the `Files` stanza that tell's Internet Explorer to not “look ahead”.

How does it work? What do I do to set it up?

- **Continuing...**

  The following changes are added to secure the server.

  This first Location directive controls access to the server, requiring anyone who accesses it to provide authentication.

```
<Location /*>
  AuthName wasv85output
  AuthType Basic
  AuthBasicProvider saf
  Require valid-user
  AuthSAFExpiration "EXPIRED! oldpw/newpw/newpw"
  AuthSAFReEnter "Enter new password one more time"
</Location>
```


The `Location` directive shown will require anyone who attempts to browse the url that specifies the location of the output to be authenticated with a userid and password. It also allows for a prompt to be specified for dealing with expired passwords.

How does it work? What do I do to set it up?

- **Continuing...**

  This next Location directive controls access to the cell directory and all files within it.

```
<Location /shcell>
  AuthName shcelloutput
  AuthType Basic
  AuthBasicProvider saf
  Require saf-group SHCFG SHCELL
  AuthSAFExpiration "EXPIRED! oldpw/newpw/newpw"
  AuthSAFReEnter "Enter new password one more time"
</Location>
```

  Save the httpd.conf file.

This `Location` directive specifies the /shcell subdirectory of the url. It will require that any authenticated user that attempts to access the specified directory be a member of the SHCFG (cell configuration) or SHCELL group. The SHCELL group was set up previously for the express purpose of allowing this check.
If an user fails this check, they'll see an authorization failure at the browser.
There will be no messages on the console or the joblog (ICH408I) indicating a failure, since all the server is doing is checking to see if the authenticated user has the required GID in their list groups of which they are a member.

How does it work? What do I do to set it up?

- **Continuing...**

  Create a JCL PROC to be used to start the HTTP server and save it in an appropriate PROCLIB.

```
//SHHTTP01 PROC ACTION='start',
// DIR='/wasv85config/shcell/shhttp01/',
// CONF='conf/httpd.conf'
//*
//IHS EXEC PGM=BPXBATCH,
// PARM='SH &DIR/bin/apachectl -k &ACTION -f &CONF -DNO_DETACH',
// MEMLIMIT=512M
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
```


Next we need a PROC (JCL Procedure) in a PROCLIB which allows started tasks to be started from it. The same PROC will be used to start and to stop the http server. The name of the PROC (member name in the PROCLIB PDS) must match the name specified earlier in the STARTED profile. The PROC simply executes the BPXBATCH program with a parm field specifying what we want to do. The default action is start.

How does it work? What do I do to set it up?

- **Continuing...**

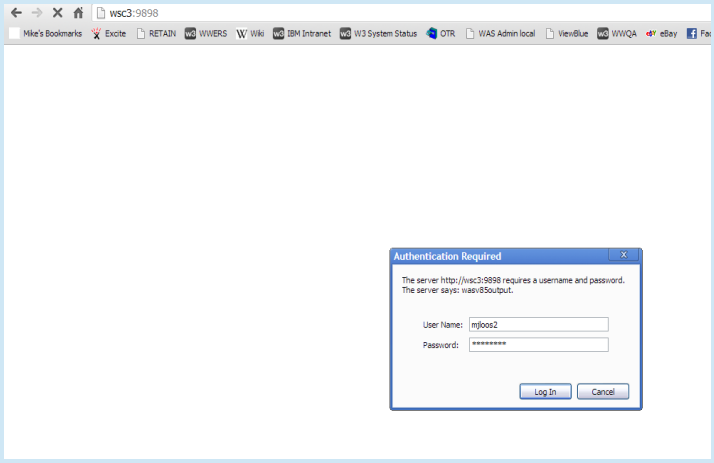
  To start the HTTP Server, issue the following command from a console. Since the default action is 'start', no parameter is necessary.

  ```
  S SHHTTP01
  ```

To stop the server, issue the same command with the action='stop' parameter (case is important, so be careful about where this command is entered).

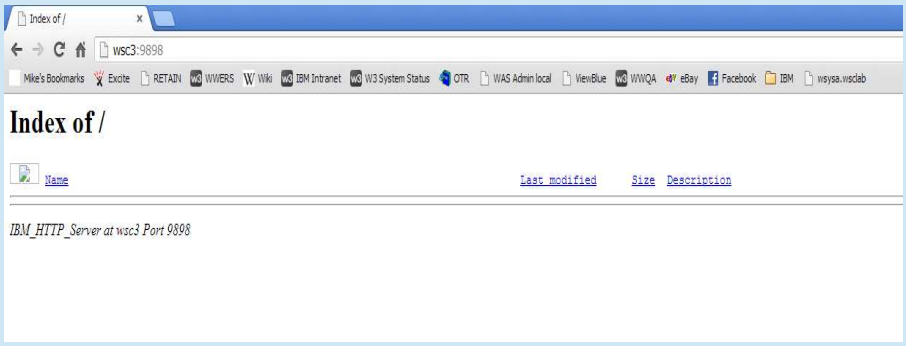
```
S SHHTTP01,ACTION='stop'
```

And finally...a Result!



Finally we can open a browser (your choice which brand), and direct it to the appropriate url for the server.
The user will get a prompt for a userid and password.
The one we have used is a userid which is NOT connected to either of the two groups on the `Location` directive, so...

And finally...a Result!



Whoops! No output...mjloos2 must not be a member of an appropriate group...

```
READY
connect mjloos2 group(shcell)
READY
```

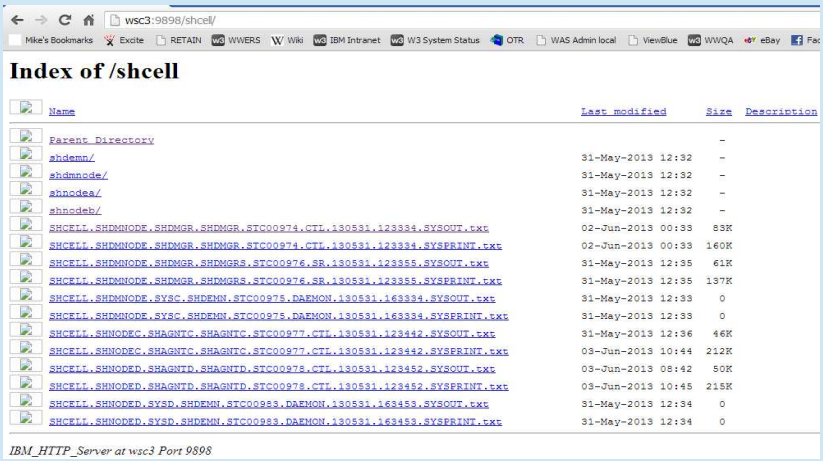

As you can see, the server has authenticated the user, but the user isn't allowed to see any output. This is simple to remedy...
Connect the user to an appropriate group with a command like the one shown.
Then the user can simply reload (or refresh) the browser and they'll see the files from the directory as appropriate, in this case, the cell level directory. To continue...

And finally...a Result!

Then click on the cell name “shcell”...

Click on the cell name link “shcell”.

And finally...a Result!

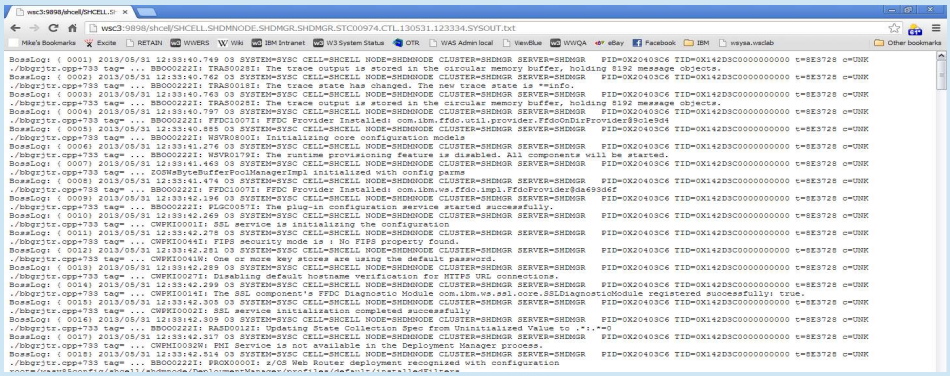


Then click on any file name you choose.

23

The user is now presented with a list of files, which are all “clickable” links. Clicking on any one of them will open and display the file.

And finally...a Result!



The ouput is scrollable, and dynamic. If you happen to be at the bottom of the screen and want to see output which has been added since you opened the file, simply click on the browser's refresh icon.

And if you prefer a more granular security setup...

- **Add something similar to the following...**

  This Location directive controls access to the files under the deployment manager node.

```
<Location /shcell/shdmnode>
  AuthName shcellshdmnodeoutput
  AuthType Basic
  AuthBasicProvider saf
  Require saf-group SHCFG SHCELL SHDMNODE
  AuthSAFExpiration "EXPIRED! oldpw/newpw/newpw"
  AuthSAFReEnter "Enter new password one more
  time"
</Location>
```

  Members of any of the three groups will be allowed access.

25

If you wished to implement more granular security, for instance to allow one group access to the deployment manager output only, you would first have to direct the output to a separate path (different directory in the same structure, keeping it hierarchical makes it easiest), with a WebSphere variable (same variable name, different scope (deployment manager node (shdmnode), and a different pathname for the value. Then you'd add a Location directive for the more specific pathname, in addition to the more generic one previously specified, that requires membership in a different group (SHDMNODE). In the shown directive, the two groups from the previous directive are also used. Although not actually necessary, this allows access in an hierarchical fashion. If you have access to the cell, you also have access to the node in the cell, the servers in the node, etc. If you only are a member of the node level group, you won't be able to see anything above the node level directory. You would also have to specify a more specific url in the browser, as you wouldn't have the access necessary to traverse through the hierarchy. Not a bad thing, necessarily, but worth knowing.

And if you prefer a more granular security setup...

- **`Continuing...`**
  This Location directive controls access to the files under a managed (AppServer) node.

```
<Location /shcell/shnodec>
  AuthName shcellshnodecoutput
  AuthType Basic
  AuthBasicProvider saf
  Require saf-group SHCFG SHCELL SHNODEC
  AuthSAFExpiration "EXPIRED! oldpw/newpw/newpw"
  AuthSAFReEnter "Enter new password one more
  time"
</Location>
```

  Members of any of the three groups will be allowed access.

The shown `Location` directive is very similar to the previous slide, but is for an appserver node. Setup is the same with altered names.

And if you prefer a more granular security setup...

- **Continuing...**

  This Location directive controls access to the files under an AppServer node for a particular server..

```
<Location /shcell/shnodec/shsr01c>
  AuthName shcellshnodecshsr01coutput
  AuthType Basic
  AuthBasicProvider saf
  Require saf-group SHCFG SHCELL SHNODEC SHSR01C
  AuthSAFExpiration "EXPIRED! oldpw/newpw/newpw"
  AuthSAFReEnter "Enter new password one more time"
</Location>
```

  Members of any of the four groups will be allowed access.

  This Location directive is same as above for a different server.

```
<Location /shcell/shnodec/shsr02c>
  AuthName shcellshnodecshsr02coutput
  AuthType Basic
  AuthBasicProvider saf
  Require saf-group SHCFG SHCELL SHNODEC SHSR02C
  AuthSAFExpiration "EXPIRED! oldpw/newpw/newpw"
  AuthSAFReEnter "Enter new password one more time"
</Location>
```

  Members of any of the four groups will be allowed access.

The two directives shown take the concept from the previous two slides down a level from the node to the server(s). Same things must be specified. Note the additional group specified. Remember that membership in any one of the four groups will allow you access to the directory specified.

And if you prefer a more granular security setup...

- **`Continuing...`**
  - Similar Location directives may (should?) be added for:
    - /shcell/shnoded
    - /shcell/shnoded/shsr01d
    - /shcell/shnoded/shsr01d
  - And for any other directories which you wish to protect.

You may add `Location` directives (and corresponding variable/value specifications at appropriate scopes) to get as much or as little granularity as you wish.
It is worth noting that the same (or a different if you prefer) http server, with the specification of appropriate `Location` directives, could be used to view seperate files for verbose garbage collection (if you direct that output to a file instead of the stderr stream), ffdc files, or anything else you wish to view from a browser.

References...

- **Directing SYSPRINT Output to an HFS File in WebSphere for z/OS**
  - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD101087
- Routing Operator Messages in WebSphere Application Servers for z/OS V6 & V7
  - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD103695
- Implementing the Output APAR (PM74923) enhancements in WebSphere Application Server on z/OS
  - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102267
  - This paper is the basis for this presentation and goes into much greater detail.
- IBM HTTP Server for WebSphere Application Server Version 8.5
  - SA32-1087-00

These are a few references which you may find helpful as you implement this function.

0

And now...

Q and A...