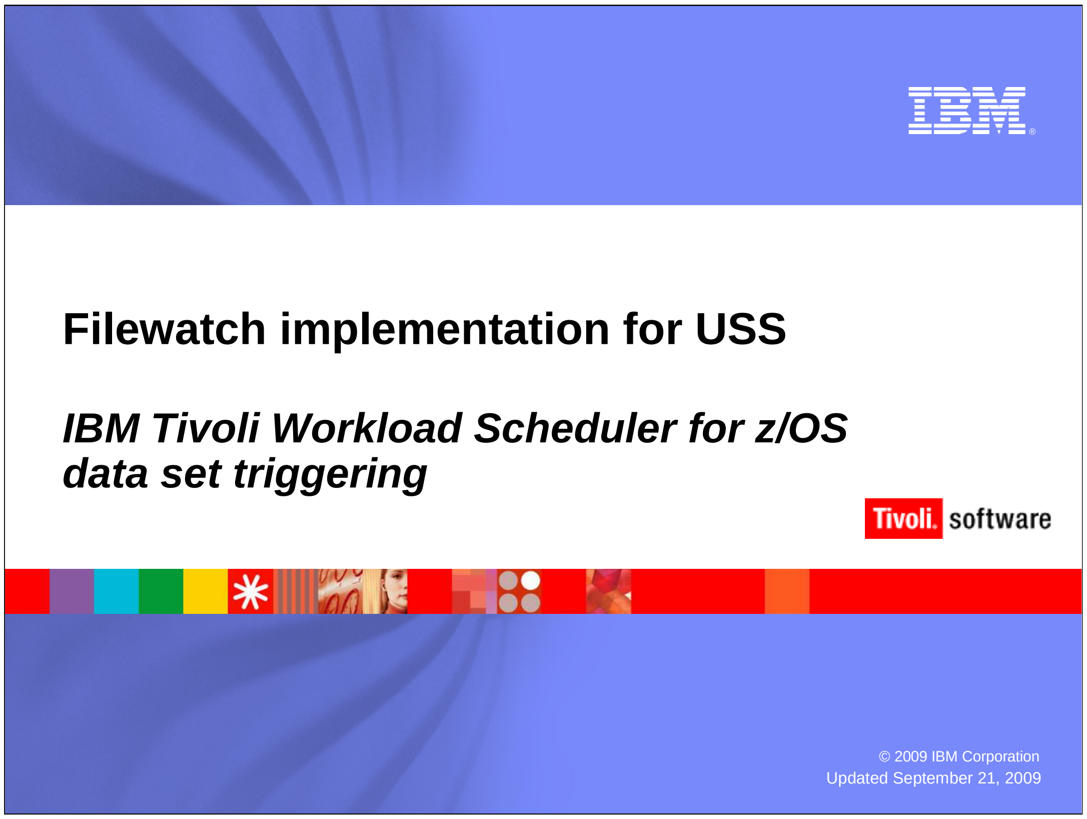
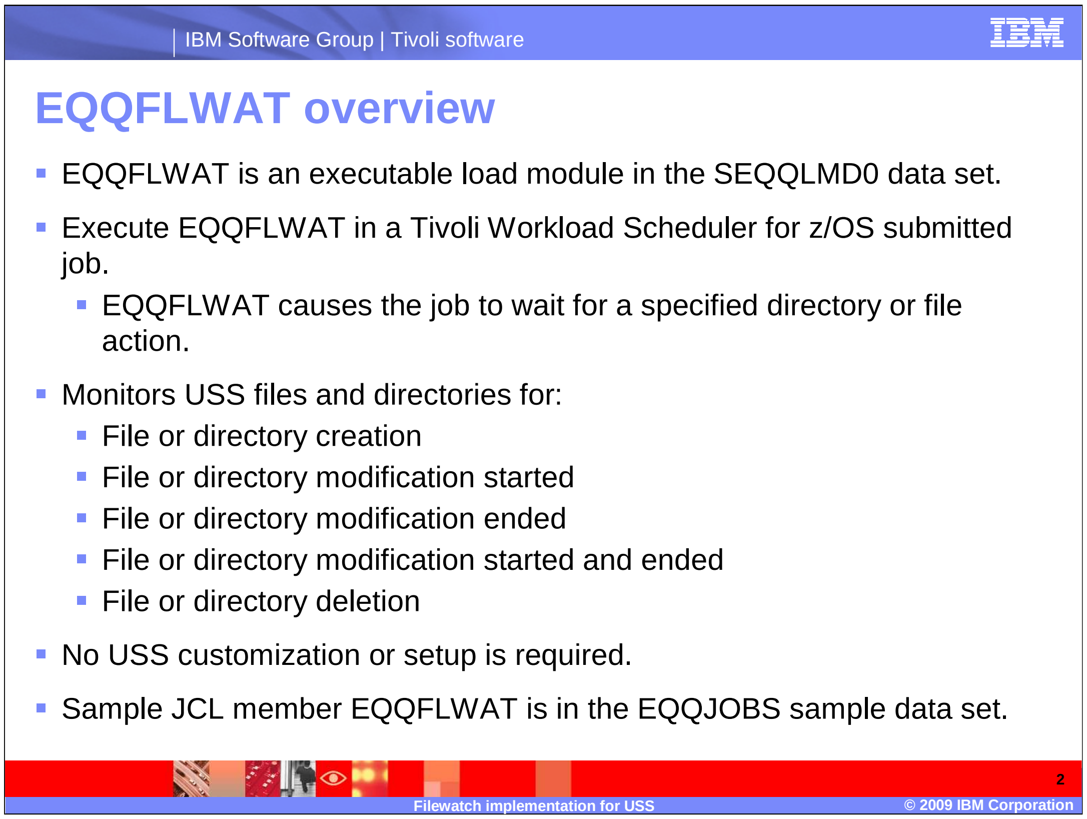


Filewatch implementation for USS

IBM Tivoli Workload Scheduler for z/OS data set triggering

Filewatch implementation for USS.

This training module provides you with the knowledge required to implement the filewatch utility EQQFLWAT provided with IBM Tivoli Workload Scheduler for z/OS® 8.5. The EQQFLWAT load module monitors changes in USS-based directories and files.



EQQFLWAT overview

- EQQFLWAT is an executable load module in the SEQQLMD0 data set.
- Execute EQQFLWAT in a Tivoli Workload Scheduler for z/OS submitted job.
  - EQQFLWAT causes the job to wait for a specified directory or file action.
- Monitors USS files and directories for:
  - File or directory creation
  - File or directory modification started
  - File or directory modification ended
  - File or directory modification started and ended
  - File or directory deletion
- No USS customization or setup is required.
- Sample JCL member EQQFLWAT is in the EQQJOBS sample data set.

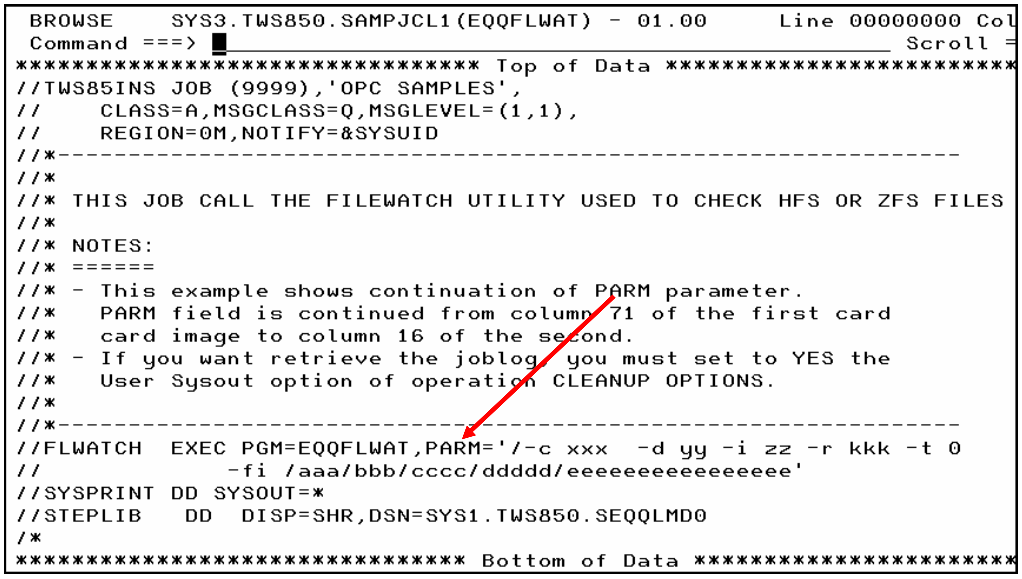
EQQFLWAT overview.

EQQFLWAT is a Tivoli Workload Scheduler for z/OS 8.5 provided load module. You can execute EQQFLWAT as part of a Tivoli Workload Scheduler for z/OS scheduled operation. By defining a job that runs this utility, you can implement a file dependency that is a relationship between a UNIX® System Services (USS) based file or directory and the operation. EQQFLWAT will monitor the file or directory for a specific activity that you define. EQQFLWAT will suspend job processing until that activity occurs or a deadline that you specify is reached. The return code that EQQFLWAT produces can then be used to determine if successor job steps or operations should be run.

The five possible file and directory actions that can be defined are: creation, modification started, modification completed, modification started and completed, and deletion. You do not have to customize USS to implement EQQFLWAT. A sample job member called EQQFLWAT is generated when you run the EQQJOBS installation utility.

IBM Software Group | Tivoli software

EQQFLWAT sample JCL



```
 BROWSE    SYS3.TWS850.SAMPJCL1(EQQFLWAT) - 01.00      Line 00000000 Col
 Command ===> █_____________________________________________________ Scroll =
********************************* Top of Data *************************
//TWS85INS JOB (9999),'OPC SAMPLES',
//    CLASS=A,MSGCLASS=Q,MSGLEVEL=(1,1),
//    REGION=0M,NOTIFY=&SYSUID
//*------------------------------------------------------------------
//*
//* THIS JOB CALL THE FILEWATCH UTILITY USED TO CHECK HFS OR ZFS FILES
//*
//* NOTES:
//* ======
//* - This example shows continuation of PARM parameter.
//*   PARM field is continued from column 71 of the first card
//*   card image to column 16 of the second.
//* - If you want retrieve the joblog, you must set to YES the
//*   User Sysout option of operation CLEANUP OPTIONS.
//*
//*------------------------------------------------------------------
//FLWATCH  EXEC PGM=EQQFLWAT,PARM='/-c xxx  -d yy -i zz -r kkk -t 0
//              -fi /aaa/bbb/cccc/ddddd/eeeeeeeeeeeeeee'
//SYSPRINT DD SYSOUT=*
//STEPLIB   DD  DISP=SHR,DSN=SYS1.TWS850.SEQQLMD0
/*
******************************** Bottom of Data ***********************
```

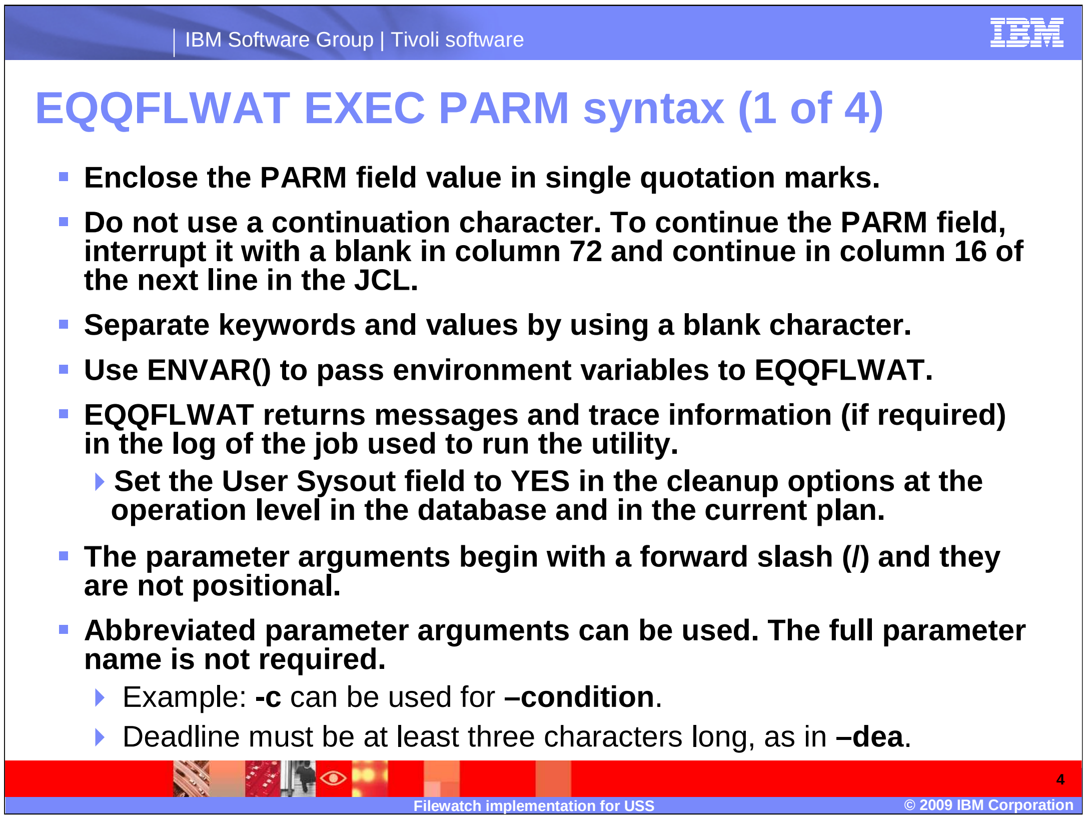
3

Filewatch implementation for USS © 2009 IBM Corporation

EQQFLWAT sample JCL.

The member EQQFLWAT in the EQQJOBS sample JCL data set provides a starting point for using the filewatch utility. The user running EQQFLWAT must have USS execute access to the directory path that contains files to be monitored. Parameters are passed to the EQQFLWAT utility using the PARM parameter of the EXEC JCL statement. The EQQFLWAT parameters provided in this sample job contain placeholders that must be replaced with valid values. The parameters are enclosed in single quotation marks and can be continued across multiple lines in the JCL.

This example shows the continuation of the PARM EXEC statement parameter. The PARM field is continued from column 71 of the EXEC statement line to column 16 of the next line.



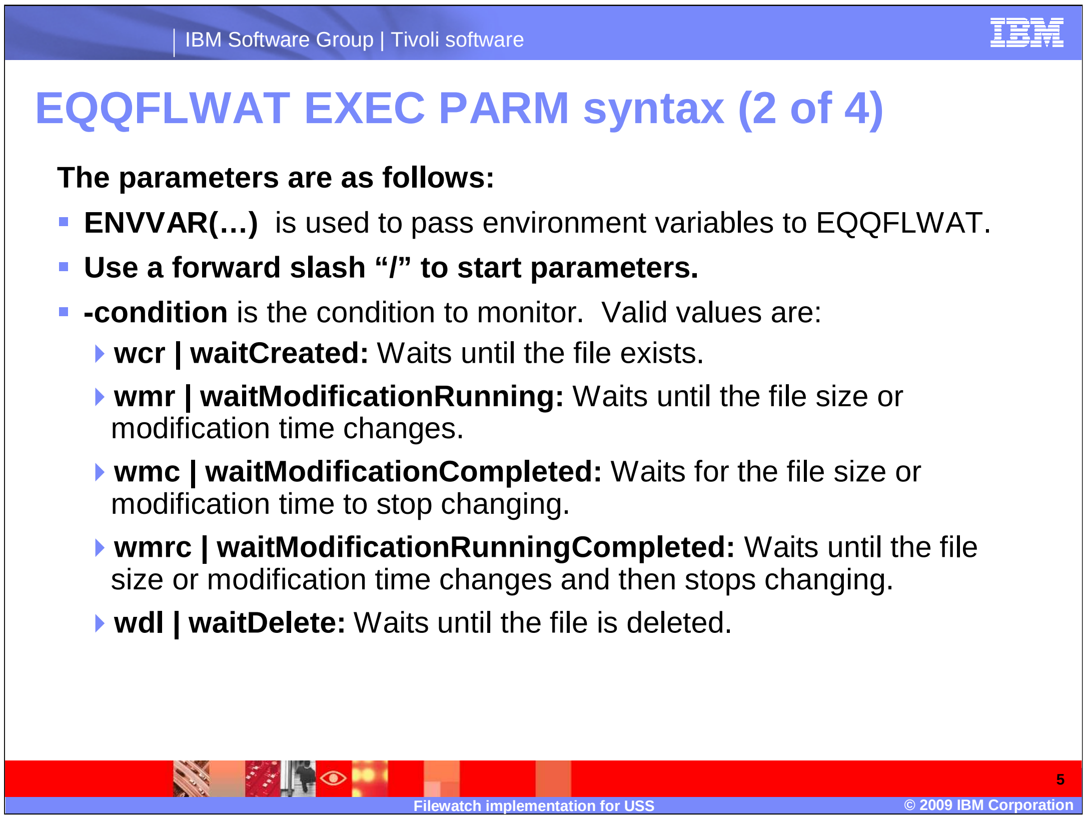
EQQFLWAT EXEC PARM syntax (1 of 4)

- **Enclose the PARM field value in single quotation marks.**
- **Do not use a continuation character. To continue the PARM field, interrupt it with a blank in column 72 and continue in column 16 of the next line in the JCL.**
- **Separate keywords and values by using a blank character.**
- **Use ENVAR() to pass environment variables to EQQFLWAT.**
- **EQQFLWAT returns messages and trace information (if required) in the log of the job used to run the utility.**
  - **Set the User Sysout field to YES in the cleanup options at the operation level in the database and in the current plan.**
- **The parameter arguments begin with a forward slash (/) and they are not positional.**
- **Abbreviated parameter arguments can be used. The full parameter name is not required.**
  - Example: **-c** can be used for **–condition**.
  - Deadline must be at least three characters long, as in **–dea**.

EQQFLWAT EXEC PARM syntax (1 of 4).

Parameters are passed to the EQQFLWAT program using the PARM field on the EXEC statement. The parameter values are specified in a UNIX-like format as keyword values prefixed with a dash. Enclose the entire EXEC statement PARM field value in single quotation marks. End the first line of the PARM field before column 72 and leave a blank character in column 72. Start the continuation of the PARM field's second line in column 16. If you are starting a new parameter option in column 16 of the second line by placing a dash in column 16, then there must be a blank in column 71 of the first line to correctly separate the two parameter options.

Use ENVAR() to pass environment variables to EQQFLWAT. If you use the job log retrieval function, set the User Sysout field in the cleanup options at the operation level in the database and in the current plan. EQQFLWAT returns messages and trace information in the log of the job used to run the utility. The parameter arguments begin with a forward slash and they are not positional. Abbreviated parameter arguments can be used. The full parameter name is not required. For example: **-c** can be used for –condition. However, deadline must be at least three characters long, as in **–dea**.



EQQFLWAT EXEC PARM syntax (2 of 4)

The parameters are as follows:

- **ENVVAR(…)** is used to pass environment variables to EQQFLWAT.
- **Use a forward slash "/" to start parameters.**
- **-condition** is the condition to monitor. Valid values are:
  - **wcr | waitCreated:** Waits until the file exists.
  - **wmr | waitModificationRunning:** Waits until the file size or modification time changes.
  - **wmc | waitModificationCompleted:** Waits for the file size or modification time to stop changing.
  - **wmrc | waitModificationRunningCompleted:** Waits until the file size or modification time changes and then stops changing.
  - **wdl | waitDelete:** Waits until the file is deleted.

EQQFLWAT EXEC PARM syntax (2 of 4).

ENVAR() Use ENVAR() to pass environment variables to EQQFLWAT.

EQQFLWAT accepts the following parameters. The condition and file name parameters are required.

-condition is the condition to be checked and is a required parameter. Valid values are:

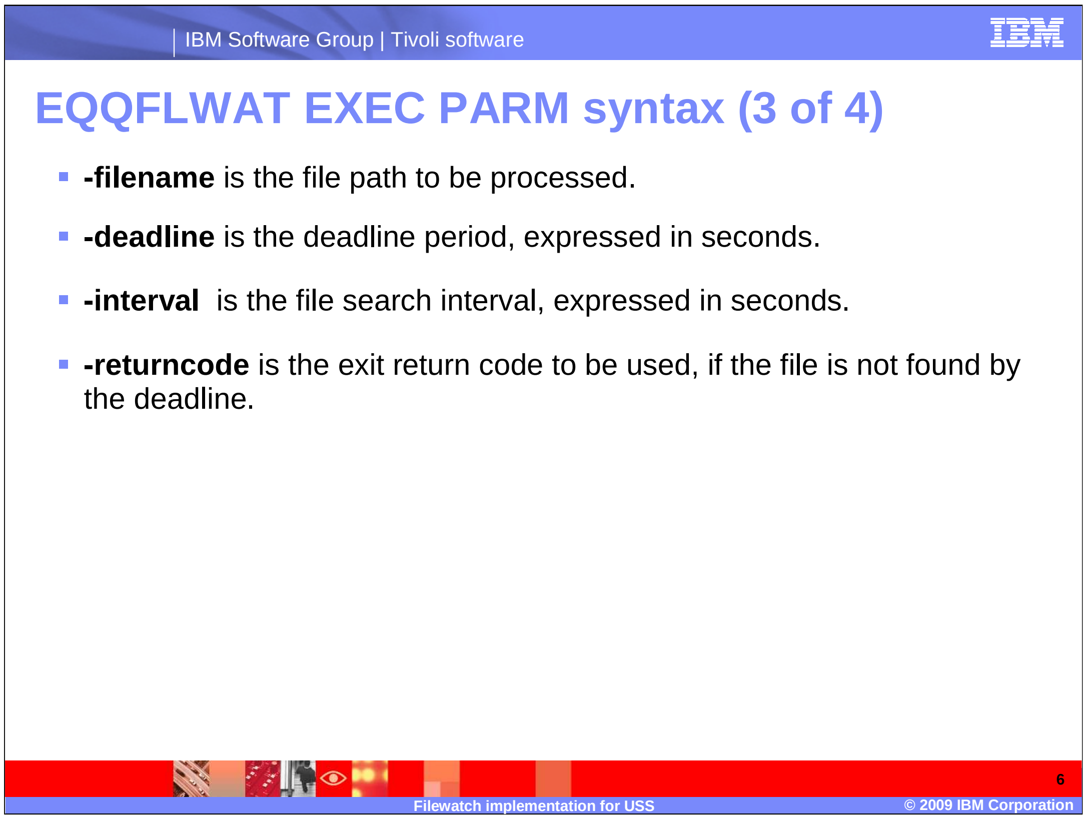
wcr | waitCreated causes EQQFLWAT to wait until the file exists. If the file already exists, filewatch exits immediately. If the -filename argument specifies a directory, the process waits until the directory exists and contains a new file.

wmr | waitModificationRunning causes EQQFLWAT to wait until the file size or modification time changes for the specified file. If the –filename argument specifies a directory, EQQFLWAT waits until the size or modification time changes for any file in the directory. Also, if a file or subdirectory is created or deleted in the specified directory, the EQQFLWAT program will complete with condition code zero. If the specified directory contains subdirectories and files are created or deleted in a subdirectory, it will also cause EQQFLWAT to complete with condition code zero. Note that a modification to a file in a subdirectory does not affect EQQFLWAT processing.

wmc | waitModificationCompleted causes a wait until the file size or modification time stops changing. EQQFLWAT waits until there are three consecutive search intervals without any changes to the file. If the -filename argument specifies a directory, EQQFLWAT waits until the size or modification time has no changes to any file or subdirectory in the specified directory for three consecutive intervals. Note that the EQQFLWAT utility does not have to detect an actual initial change with this option; it only has to detect that there has been no change for three consecutive intervals. If any file or subdirectory is created, deleted, or modified, the interval counting starts over. A subdirectory is modified by creating or deleting a file in that subdirectory.

wmrc | waitModificationRunningCompleted causes a wait until the file size or modification time changes and then stops changing. After the first detected change, EQQFLWAT waits until there are three consecutive search intervals without any further change. If the –filename argument specifies a directory, EQQFLWAT monitors the files and subdirectories in the specified directory for an initial change and then three consecutive search intervals without a change. Again, as with the wmc option, a change detected for any file or subdirectory restarts the interval counting.

wdl | waitDelete stops waiting when the file is deleted. If the -filename argument specifies a directory, the process waits until a file is deleted from the directory.



EQQFLWAT EXEC PARM syntax (3 of 4)

- **-filename** is the file path to be processed.
- **-deadline** is the deadline period, expressed in seconds.
- **-interval** is the file search interval, expressed in seconds.
- **-returncode** is the exit return code to be used, if the file is not found by the deadline.

EQQFLWAT EXEC PARM syntax (3 of 4).

-filename is the file path to be processed. You can embed blank or special characters by using double quotation marks. Wildcard characters are not supported. To include more files in the monitoring process, you can store those files in a specific directory and use a file path specifying that directory.

-deadline is the deadline period, expressed in seconds. The allowed formats are:

- An integer in the range 0 to 315360000 (the upper value corresponds to 1 year). To have filewatch perform an infinite loop, specify 0.

- hh:mm:ss, in the range 00:00:01 to 24:00:00, to select a time within the same day when filewatch is started.

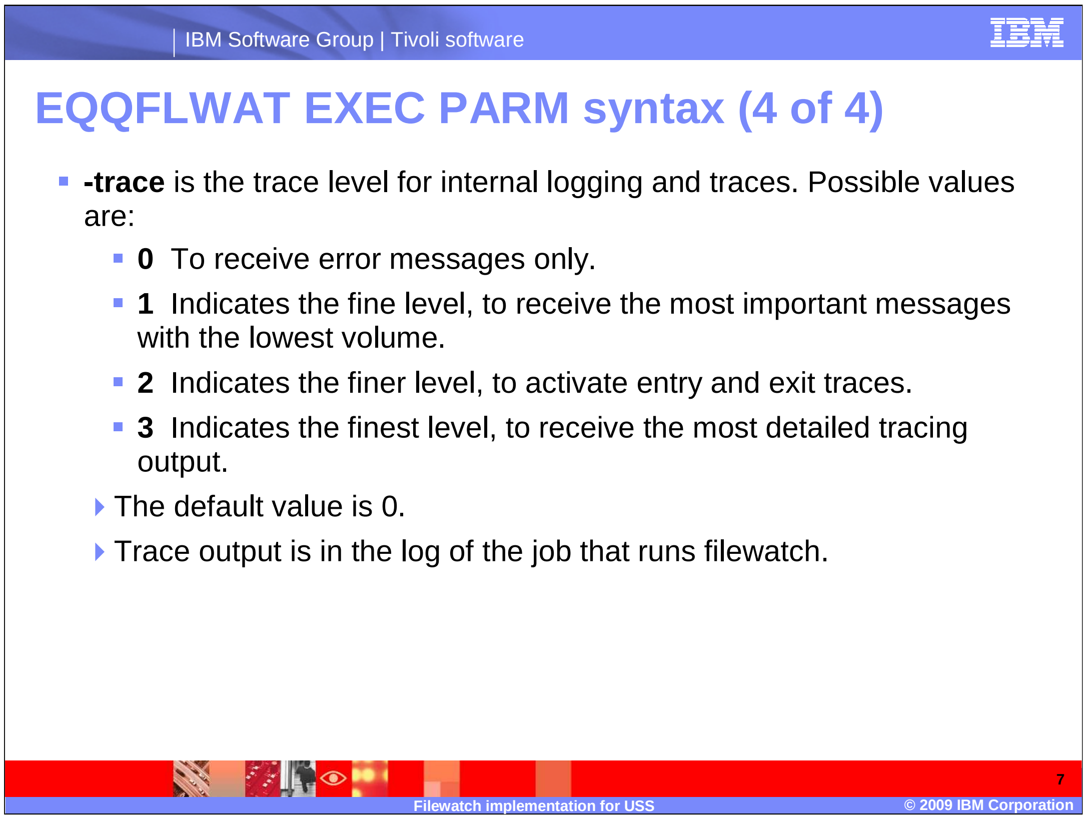
-interval is the file search interval, expressed in seconds. Specify an integer in the following range:

• 5 through 3600 is valid when wcr or wdl are specified as the condition value.

• 30 through 3600 is valid otherwise.

The default is 60.

-returncode is the exit return code that EQQFLWAT should return if the file is not found by the deadline. Specify an integer in the range 0 through 256. The returncode value is ignored if you specify 0 as deadline value. The default is 4.



EQQFLWAT EXEC PARM syntax (4 of 4)

- **-trace** is the trace level for internal logging and traces. Possible values are:
  - **0** To receive error messages only.
  - **1** Indicates the fine level, to receive the most important messages with the lowest volume.
  - **2** Indicates the finer level, to activate entry and exit traces.
  - **3** Indicates the finest level, to receive the most detailed tracing output.
  - The default value is 0.
  - Trace output is in the log of the job that runs filewatch.

EQQFLWAT EXEC PARM syntax (4 of 4).

-trace is the trace level for internal logging and traces. Possible values are as follows:

• 0 To receive error messages only.

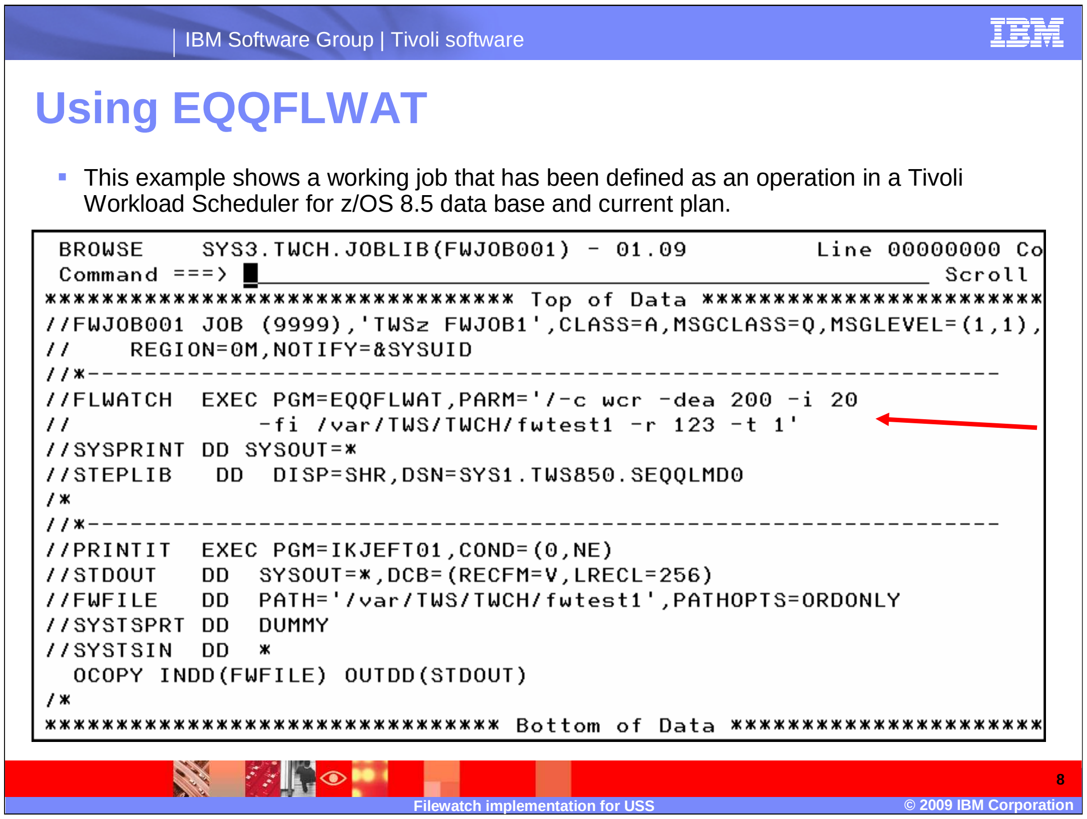
• 1 Indicates the fine level, to receive the most important messages with the lowest volume.

• 2 Indicates the finer level, to activate entry and exit traces.

• 3 Indicates the finest level, to receive the most detailed tracing output.

The default value is 0.

You find the trace output in the log of the job that runs filewatch.



IBM Software Group | Tivoli software

Using EQQFLWAT

- This example shows a working job that has been defined as an operation in a Tivoli Workload Scheduler for z/OS 8.5 data base and current plan.

```
 BROWSE    SYS3.TWCH.JOBLIB(FWJOB001) - 01.09         Line 00000000 Co
 Command ===>                                                  Scroll
********************************* Top of Data ************************
//FWJOB001 JOB (9999),'TWSz FWJOB1',CLASS=A,MSGCLASS=Q,MSGLEVEL=(1,1),
//     REGION=0M,NOTIFY=&SYSUID
//*-------------------------------------------------------------------
//FLWATCH  EXEC PGM=EQQFLWAT,PARM='/-c wcr -dea 200 -i 20
//             -fi /var/TWS/TWCH/fwtest1 -r 123 -t 1'
//SYSPRINT DD SYSOUT=*
//STEPLIB   DD  DISP=SHR,DSN=SYS1.TWS850.SEQQLMD0
/*
//*-------------------------------------------------------------------
//PRINTIT  EXEC PGM=IKJEFT01,COND=(0,NE)
//STDOUT    DD  SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//FWFILE    DD  PATH='/var/TWS/TWCH/fwtest1',PATHOPTS=ORDONLY
//SYSTSPRT DD   DUMMY
//SYSTSIN  DD   *
  OCOPY INDD(FWFILE) OUTDD(STDOUT)
/*
******************************** Bottom of Data **********************
```

8

Filewatch implementation for USS

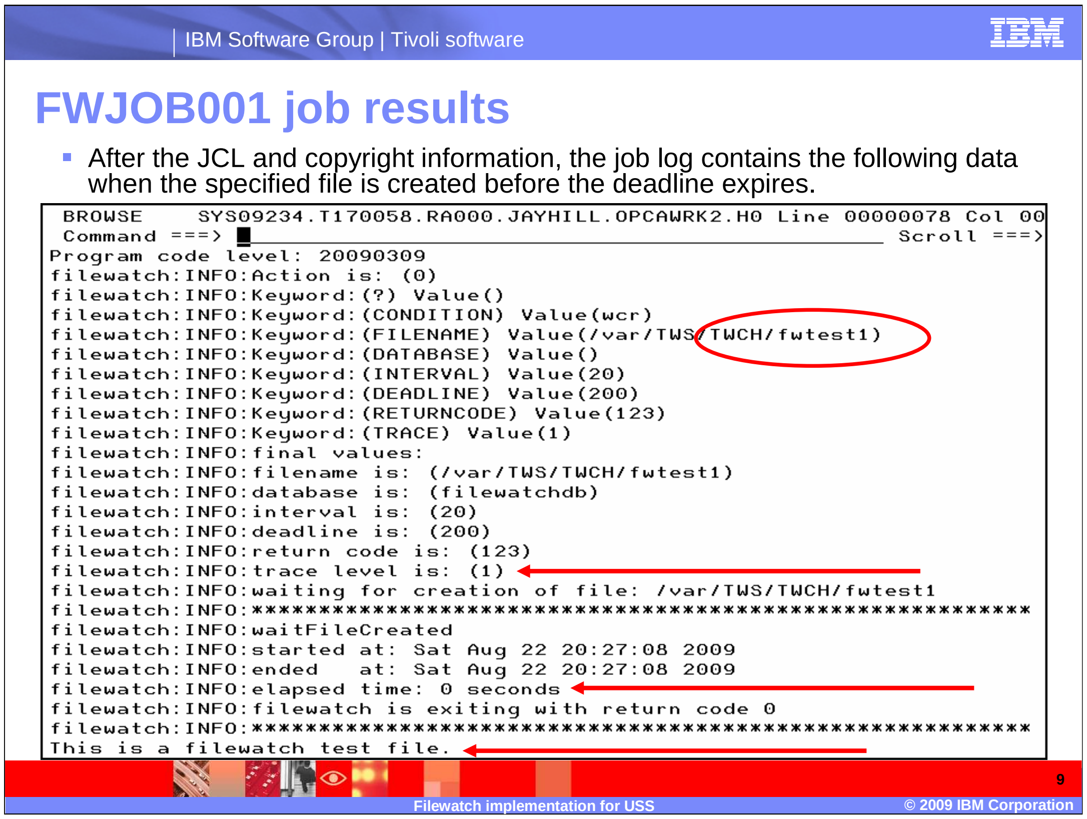
© 2009 IBM Corporation

Using EQQFLWAT.

You can use EQQFLWAT in a job that has been defined in Tivoli Workload Scheduler for z/OS. Most likely you would have multiple operations in an application and use the filewatch job as a predecessor job. However, you can also use the EQQFLWAT return code by using the COND parameter of the EXEC statement as shown on this slide. This example shows a two-step job called FWJOB001, where the first step runs EQQFLWAT and waits for a file to be created. The second step checks for a condition code zero return code from the filewatch step and then writes the file to SYSOUT. If the condition code is something other than a zero, the second step in the job does not run.

In this example the filewatch condition used is –wcr, which tells EQQFLWAT to wait until the file fwtest1 is created. A deadline of 200 seconds is used. If the file is not created before 200 seconds elapses, EQQFLWAT will end with a return code of 123, which was provided by the –r option. EQQFLWAT will check to see if the file has been created every 20 seconds based on the –i option. Notice also that the trace option –t has a value of 1.

Remember that the user running EQQFLWAT must have USS execute access to the directory path that contains files to be monitored.


IBM Software Group | Tivoli software

FWJOB001 job results

- After the JCL and copyright information, the job log contains the following data when the specified file is created before the deadline expires.

```
 BROWSE    SYS09234.T170058.RA000.JAYHILL.OPCAWRK2.H0 Line 00000078 Col 00
 Command ===> _                                                  Scroll ===>
Program code level: 20090309
filewatch:INFO:Action is: (0)
filewatch:INFO:Keyword:(?) Value()
filewatch:INFO:Keyword:(CONDITION) Value(wcr)
filewatch:INFO:Keyword:(FILENAME) Value(/var/TWS/TWCH/fwtest1)
filewatch:INFO:Keyword:(DATABASE) Value()
filewatch:INFO:Keyword:(INTERVAL) Value(20)
filewatch:INFO:Keyword:(DEADLINE) Value(200)
filewatch:INFO:Keyword:(RETURNCODE) Value(123)
filewatch:INFO:Keyword:(TRACE) Value(1)
filewatch:INFO:final values:
filewatch:INFO:filename is: (/var/TWS/TWCH/fwtest1)
filewatch:INFO:database is: (filewatchdb)
filewatch:INFO:interval is: (20)
filewatch:INFO:deadline is: (200)
filewatch:INFO:return code is: (123)
filewatch:INFO:trace level is: (1)
filewatch:INFO:waiting for creation of file: /var/TWS/TWCH/fwtest1
filewatch:INFO:***********************************************************
filewatch:INFO:waitFileCreated
filewatch:INFO:started at: Sat Aug 22 20:27:08 2009
filewatch:INFO:ended   at: Sat Aug 22 20:27:08 2009
filewatch:INFO:elapsed time: 0 seconds
filewatch:INFO:filewatch is exiting with return code 0
filewatch:INFO:***********************************************************
This is a filewatch test file.
```

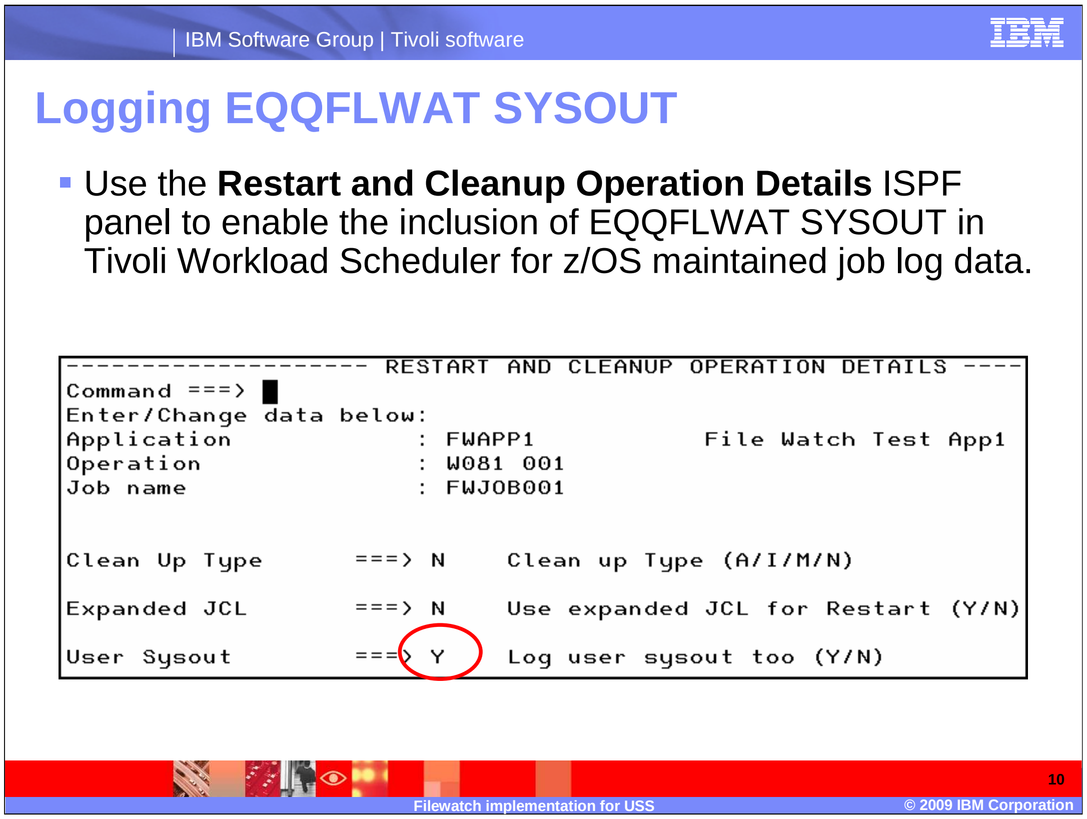
9

Filewatch implementation for USS © 2009 IBM Corporation

FWJOB001 job results.

This slide shows the output of the of FWJOB001 job from the previous slide. This output is being displayed using the Tivoli Workload Scheduler for z/OS job log retrieval function in the Modify Current Plan ISPF dialog. To use the job log retrieval function, set the User Sysout field in the cleanup options at the operation level in the database and in the current plan.

Because the filewatch trace level value was set to 1, EQQFLWAT has provided some additional information in the output. The filewatch options supplied in the EXEC PARM statement have been provided as part of the output. The results of the EQQFLWAT run have also been displayed, showing an elapsed time of 0 seconds and a return code of 0. The reason these two values are both 0, is that the –wcr condition option was used and the file already existed, so EQQFLWAT detected the file immediately and then ended with condition code 0. The output from the second job step that writes the file to SYSOUT is shown as the last line.



Logging EQQFLWAT SYSOUT

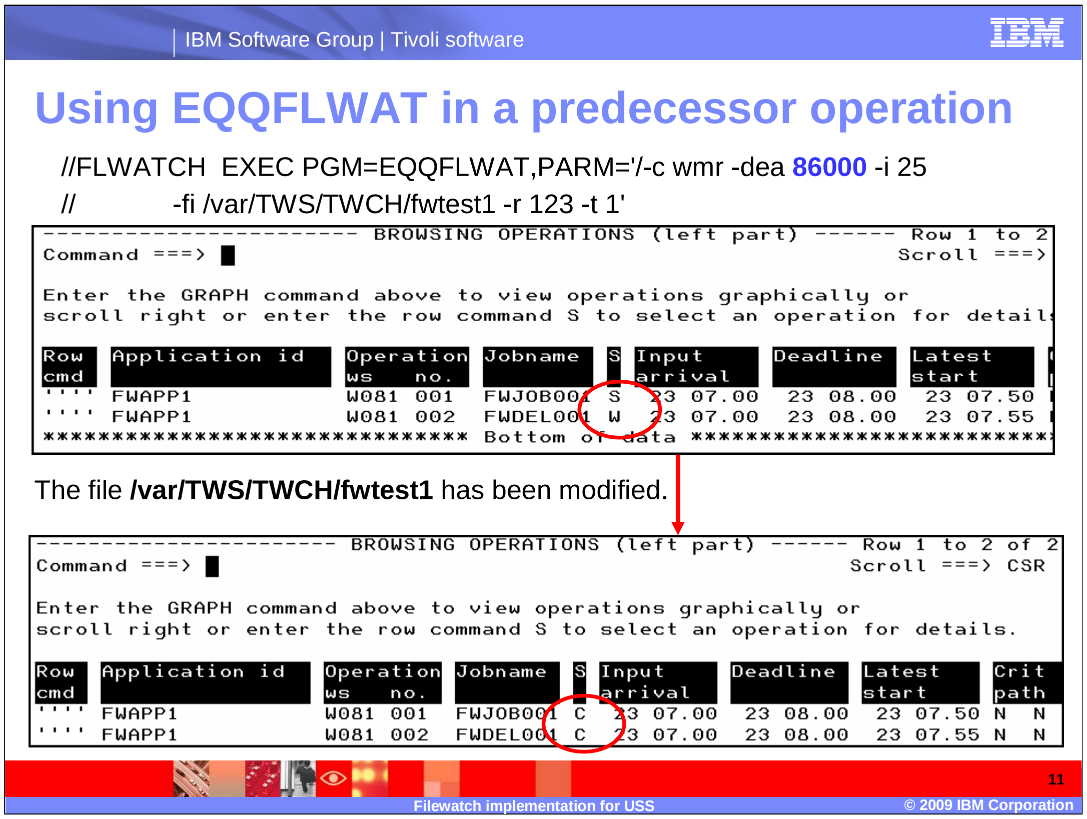
- Use the **Restart and Cleanup Operation Details** ISPF panel to enable the inclusion of EQQFLWAT SYSOUT in Tivoli Workload Scheduler for z/OS maintained job log data.

```
-------------------- RESTART AND CLEANUP OPERATION DETAILS ----
Command ===>
Enter/Change data below:
Application              : FWAPP1           File Watch Test App1
Operation                : W081 001
Job name                 : FWJOB001


Clean Up Type        ===> N    Clean up Type (A/I/M/N)

Expanded JCL         ===> N    Use expanded JCL for Restart (Y/N)

User Sysout          ===> Y    Log user sysout too (Y/N)
```

Logging EQQFLWAT SYSOUT.

Use the **Restart and Cleanup Options Details** ISPF panel to enable the inclusion of EQQFLWAT Sysout in Tivoli Workload Scheduler for z/OS maintained job log data. The example on this slide shows the User Sysout set to Y for the FWJOB001 job that was discussed on the previous slide.

The User Sysout to be included in the Tivoli Workload Scheduler for z/OS provided job log is the Sysout output from the EQQFLWAT program. The User Sysout option on this ISPF panel enables Tivoli Workload Scheduler for z/OS to collect and maintain this program-specific Sysout in addition to the standard JESJCL, JESMSGLG, and JESYSMSG job log data. If you have job log retrieval set up in Tivoli Workload Scheduler for z/OS and you set User Sysout to Y, then you can view the EQQFLWAT program-specific Sysout in the job log through the Tivoli Workload Scheduler for z/OS ISPF dialogs. Otherwise you must go to SDSF to see the EQQFLWAT Sysout.


IBM Software Group | Tivoli software

Using EQQFLWAT in a predecessor operation

```
//FLWATCH  EXEC PGM=EQQFLWAT,PARM='/-c wmr -dea 86000 -i 25
//            -fi /var/TWS/TWCH/fwtest1 -r 123 -t 1'
```

```
----------------------- BROWSING OPERATIONS (left part) ------ Row 1 to 2
Command ===>                                                    Scroll ===>

Enter the GRAPH command above to view operations graphically or
scroll right or enter the row command S to select an operation for detail

Row  Application id    Operation  Jobname  S Input     Deadline  Latest
cmd                    ws   no.              arrival             start
''''  FWAPP1           W081 001   FWJOB001 S 23 07.00  23 08.00  23 07.50
''''  FWAPP1           W081 002   FWDEL001 W 23 07.00  23 08.00  23 07.55
******************************** Bottom of data **************************
```

The file **/var/TWS/TWCH/fwtest1** has been modified.

```
----------------------- BROWSING OPERATIONS (left part) ------ Row 1 to 2 of 2
Command ===>                                                    Scroll ===> CSR

Enter the GRAPH command above to view operations graphically or
scroll right or enter the row command S to select an operation for details.

Row  Application id    Operation  Jobname  S Input     Deadline  Latest    Crit
cmd                    ws   no.              arrival             start     path
''''  FWAPP1           W081 001   FWJOB001 C 23 07.00  23 08.00  23 07.50  N   N
''''  FWAPP1           W081 002   FWDEL001 C 23 07.00  23 08.00  23 07.55  N   N
```

11

Filewatch implementation for USS

© 2009 IBM Corporation

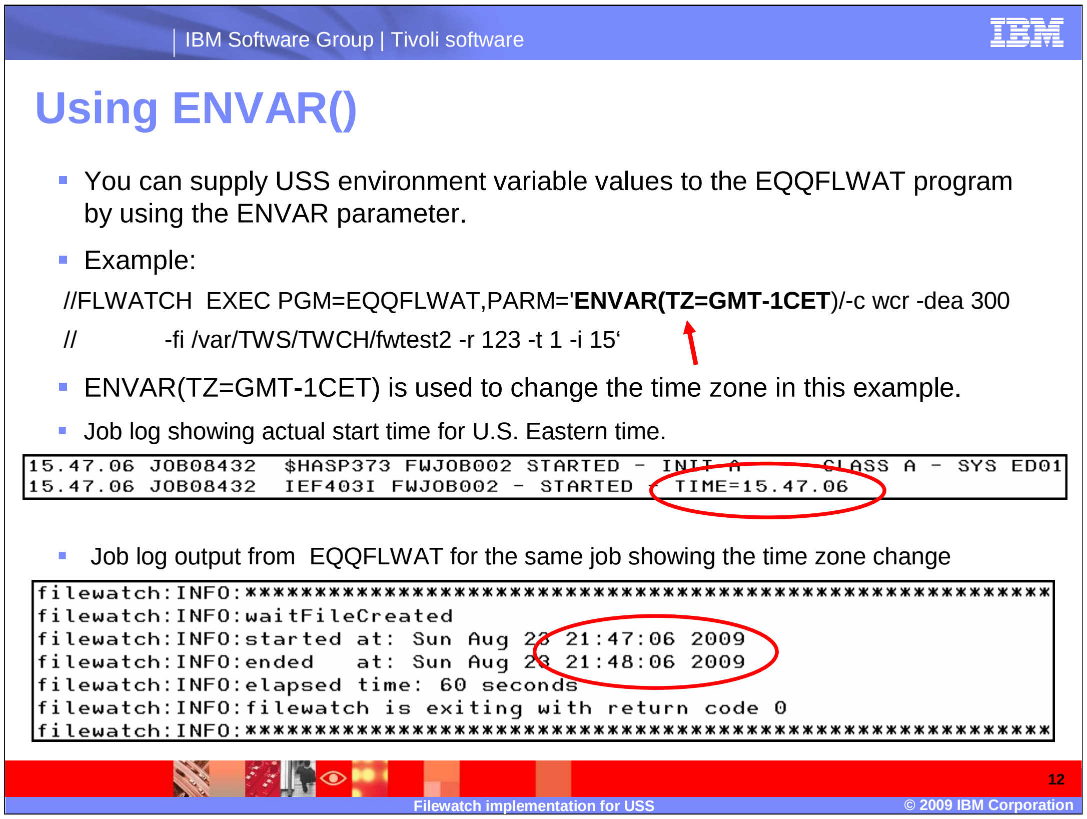
Using EQQFLWAT in a predecessor operation.

As mentioned in a previous slide, you will probably want to use EQQFLWAT in a predecessor operation with Tivoli Workload Scheduler for z/OS. In this way you can establish a dependency between USS-based files and Tivoli Workload Scheduler for z/OS operations and applications.

In the example on this slide, the FWAPP1 application has two operations. The first operation runs EQQFLWAT and waits for the file fwtest1 to be modified. You can see in the first screen capture that the first job FWJOB001 has started. The second job, FWDEL001, will run when EQQFLWAT completes successfully after the fwtest1 file has been modified. The second job, FWDEL001, will archive the modified file before it deletes it. The second screen capture shows both jobs successfully completed.

The EQQFLWAT parameters specify a deadline of 86000 seconds, which will cover almost an entire 24-hour period. If the file is not modified in that period, then the first job will fail and the second job will not run.

In addition to a simple operation dependency like the one shown on this slide, you can define more complex dependencies using the external application dependency and conditional dependency features of Tivoli Workload Scheduler for z/OS 8.5.



Using ENVAR()

- You can supply USS environment variable values to the EQQFLWAT program by using the ENVAR parameter.
- Example:

```
//FLWATCH  EXEC PGM=EQQFLWAT,PARM='ENVAR(TZ=GMT-1CET)/-c wcr -dea 300
//            -fi /var/TWS/TWCH/fwtest2 -r 123 -t 1 -i 15'
```

- ENVAR(TZ=GMT-1CET) is used to change the time zone in this example.
- Job log showing actual start time for U.S. Eastern time.

```
15.47.06 JOB08432  $HASP373 FWJOB002 STARTED - INIT A      CLASS A - SYS ED01
15.47.06 JOB08432  IEF403I FWJOB002 - STARTED - TIME=15.47.06
```

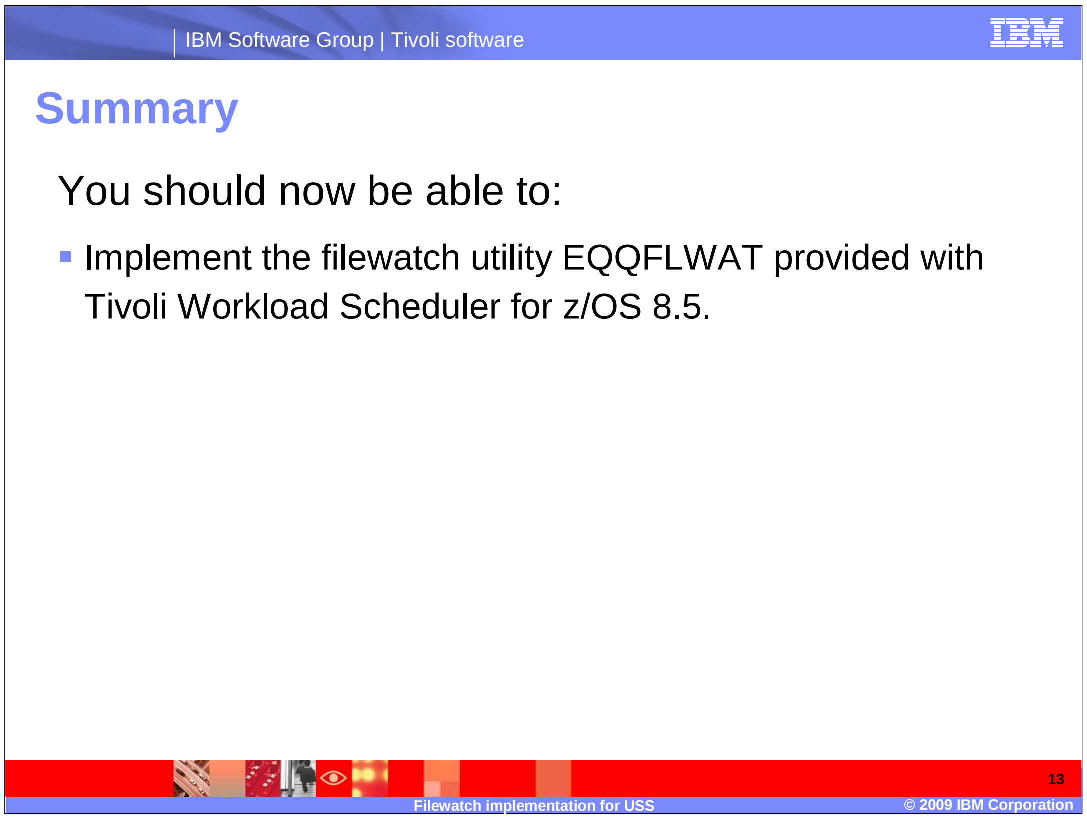
- Job log output from EQQFLWAT for the same job showing the time zone change

```
filewatch:INFO:*************************************************************
filewatch:INFO:waitFileCreated
filewatch:INFO:started at: Sun Aug 23 21:47:06 2009
filewatch:INFO:ended   at: Sun Aug 23 21:48:06 2009
filewatch:INFO:elapsed time: 60 seconds
filewatch:INFO:filewatch is exiting with return code 0
filewatch:INFO:*************************************************************
```

Using ENVAR().

You can supply USS environment variables to the EQQFLWAT program by using the ENVAR parameter. Separate the ENVAR parameter from the EQQFLWAT options with a forward slash. In this example, ENVAR(TZ=GMT-1CET) is used to change the time zone to Central Europe Time. GMT-1 is the time zone value and CET is the selected daylight saving time.

The sample job FWJOB002 was run on a z/OS system in the U.S. Eastern time zone. However, because of the ENVAR parameter the time zone was change to Central Europe daylight savings time, which is six hours later.



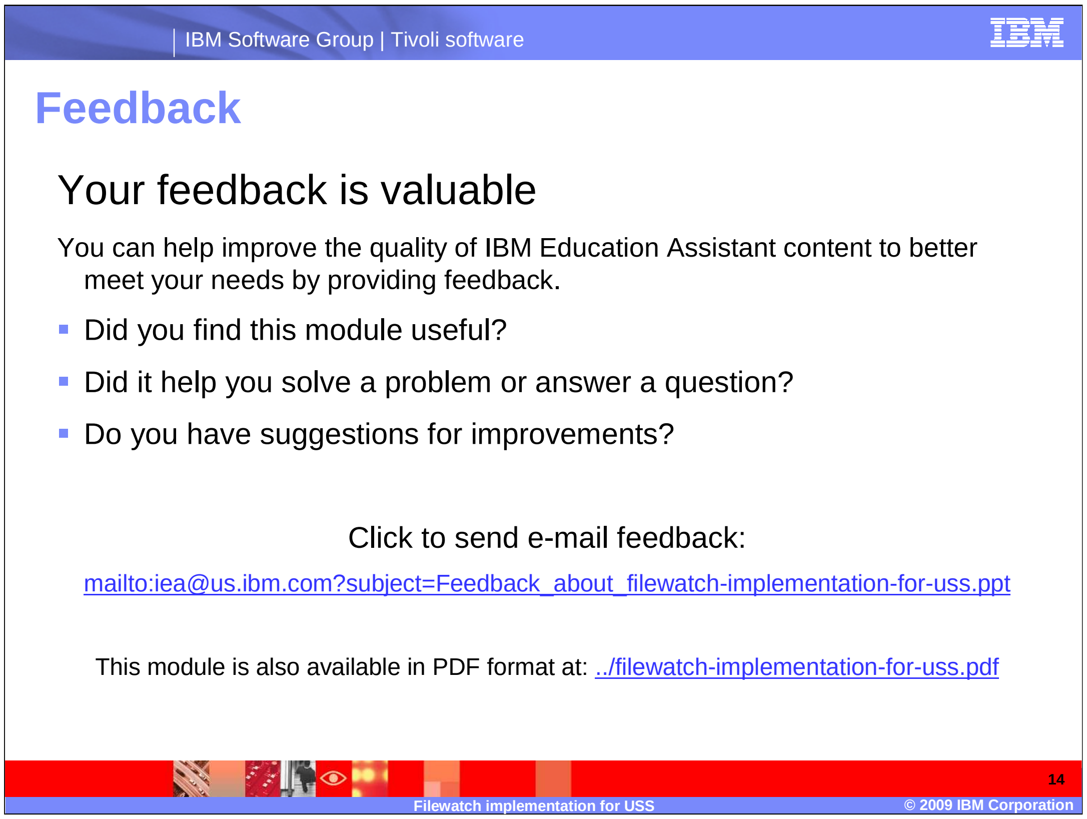
Summary

You should now be able to:

- Implement the filewatch utility EQQFLWAT provided with Tivoli Workload Scheduler for z/OS 8.5.

Summary.

In summary, you should now be able to implement the filewatch utility EQQFLWAT that is provided with Tivoli Workload Scheduler for z/OS 8.5.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_filewatch-implementation-for-uss.ppt

This module is also available in PDF format at: ../filewatch-implementation-for-uss.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers