Configuring Timeouts for WebSphere Application Server on z/OS

Session 16385

David Follis, Gary Picher, Mike Stephen

IBM

#SHAREorg

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

Important Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE.

IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.

NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:

- CREATING ANY WARRANTY OR REPRESENTATION FROM IBM (OR ITS AFFILIATES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS); OR
- ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF IBM SOFTWARE.

WebSphere Application Server

Session	Title	Time	Room	Speaker
16379	WebSphere Liberty Profle, Windows and z/OS, Hands-on Lab	Monday 4:30	Redwood	Follis/Stephen
16380	z/OS Connect: Opening up z/OS Assets to the Cloud and Mobile Worlds	Tuesday 1:45	Virginia	David Follis
16381	WebSphere Liberty Profile and Traditional WebSphere Application Server – What's New?	Tuesday 3:15	University	Follis/Stephen
16509	Debug 101-Using ISA Tools for Apps in WebSphere Application Server z/OS	Wednesday 3:15	Virginia	Mike Stephen, Joran Siu
16383	IBM Installation Manager for z/OS System Programmers: Web-based Installs, Fix Packs, and How iFixes Really Work.	Thursday 8:30	University	Don Bagwell, Bryant Panyarachun
16384	JSR 352 - The Future of Java Batch and WebSphere Compute Grid	Thursday 10:00	University	David Follis
16382	Common Problems and Other Things You Should Know about WAS on z/OS	Thursday 4:30	Virginia	Mike Stephen
16385	Configuring Timeouts for WebSphere Application Server on z/OS	Friday 10:00	Virginia	Follis/Stephen

Timeouts agenda in more detail

- General approach to setting timeout values
- Avoiding timeouts that terminate the servant
- Minimizing the effects of timeouts
- Debugging problems that cause timeouts
- CPU timer
- Monitoring
- Summary of defensive actions

This presentation is based mainly on Techdocs WP101233, WP101374, and WP102510

http://www.ibm.com/support/techdocs/atsmastr.nsf/Web/Techdocs

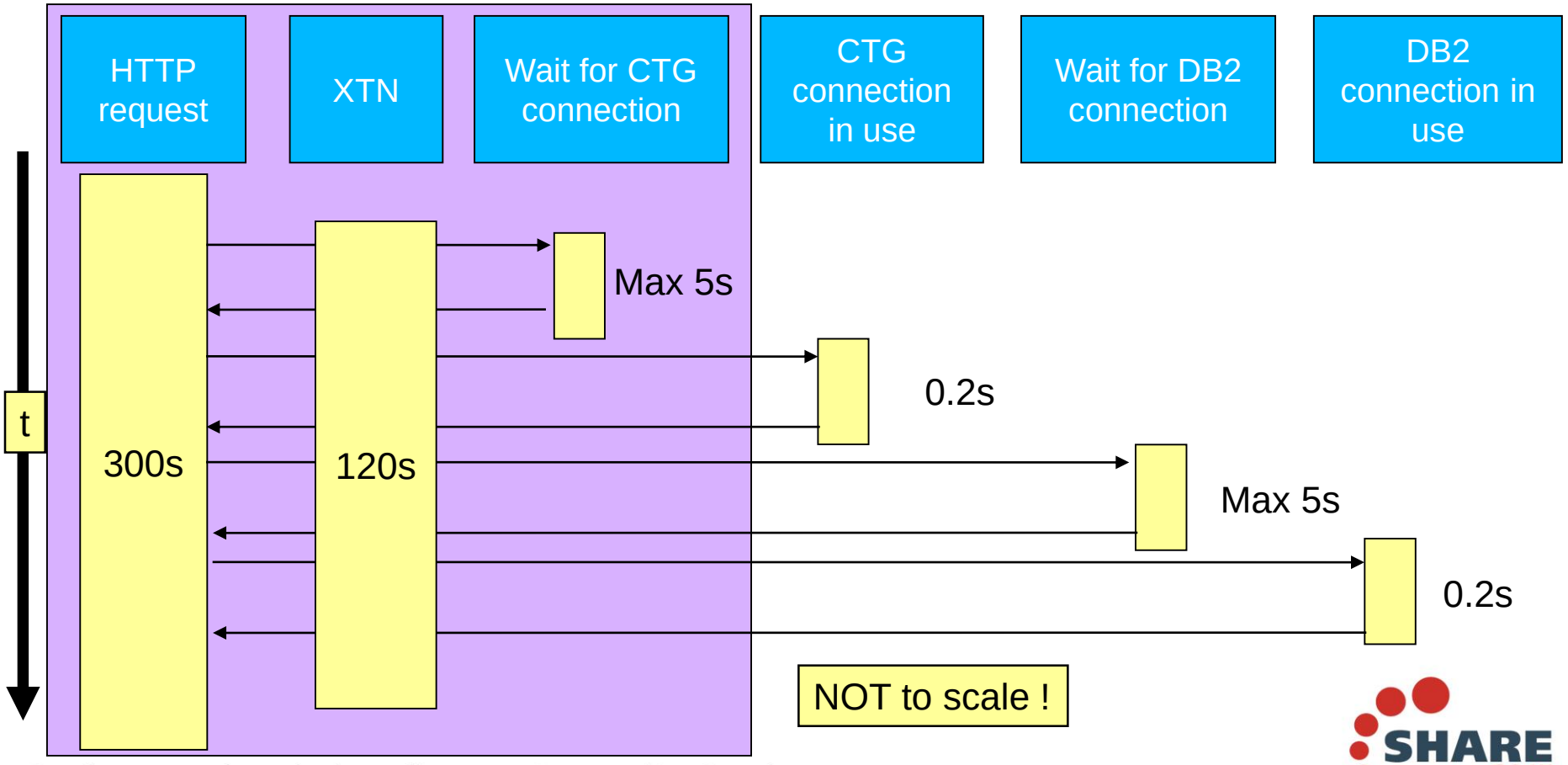
General approach to setting timeout values

General approach

- Set short timeouts close to the data - longer timeouts close to the user
- Avoid disabling timeouts completely because that leads to hangs
- Set timeouts where they can be handled by the application so other timeouts that might cause a servant restart do not occur
  - Especially when making requests to remote services
  - Every new remote service added to the architecture must be evaluated for its potential to cause timeouts or hangs. Connectors must support at least a connection timeout and a request timeout (and maybe a network timeout)
- Application code must be able to handle timeout conditions
- Use the protocol percent queue timeouts to reject work that has been waiting in the dispatch queue for a long time.
- “If work can’t run, don’t allow it in.”
- Don’t think that just because the problems are being experienced in WAS that the solution must be a change in the WAS.

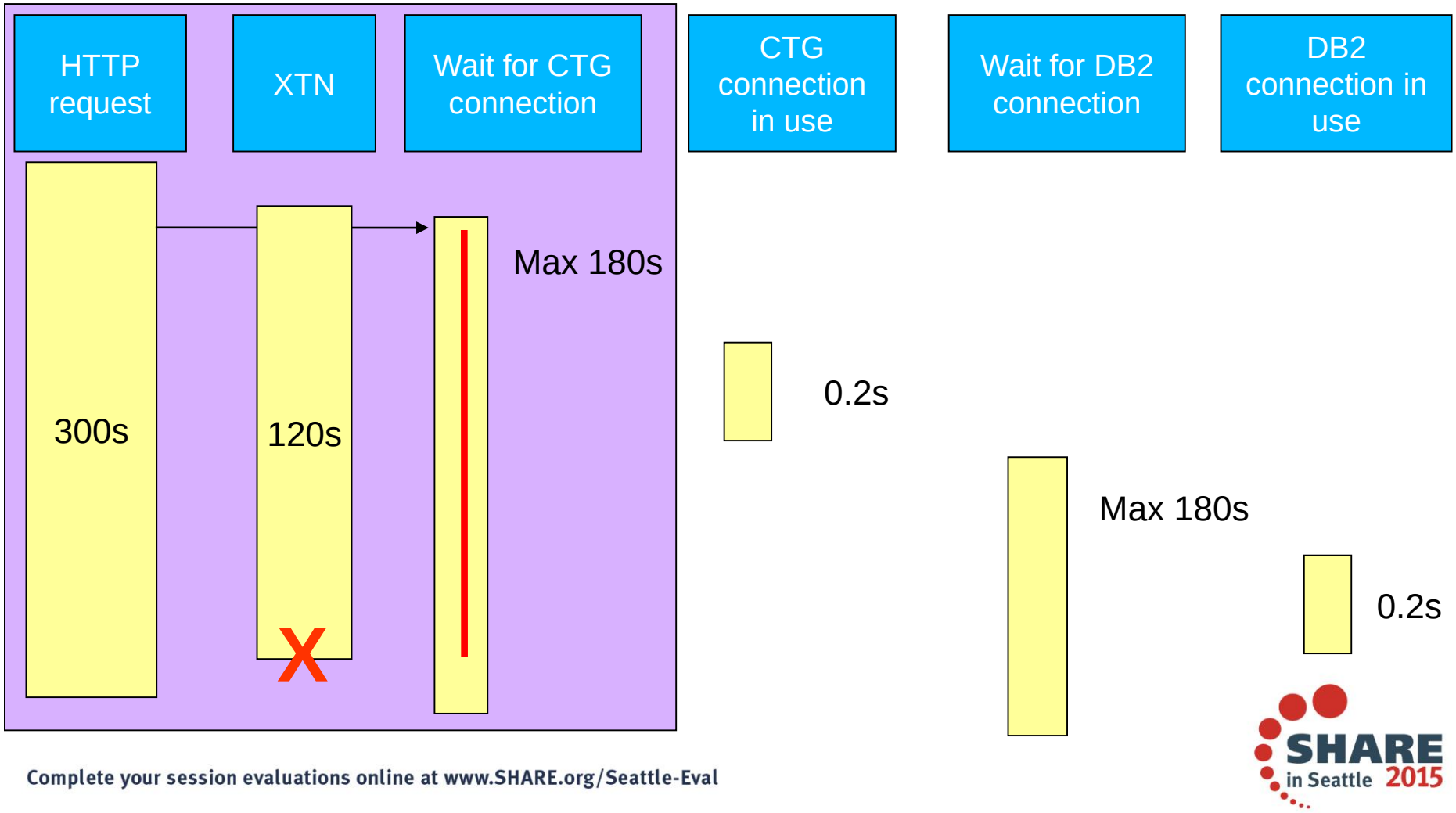
‘Nested’ timeouts concept

- Many timeouts are ‘nested’
- In general, timeouts that are further from the client should be shorter



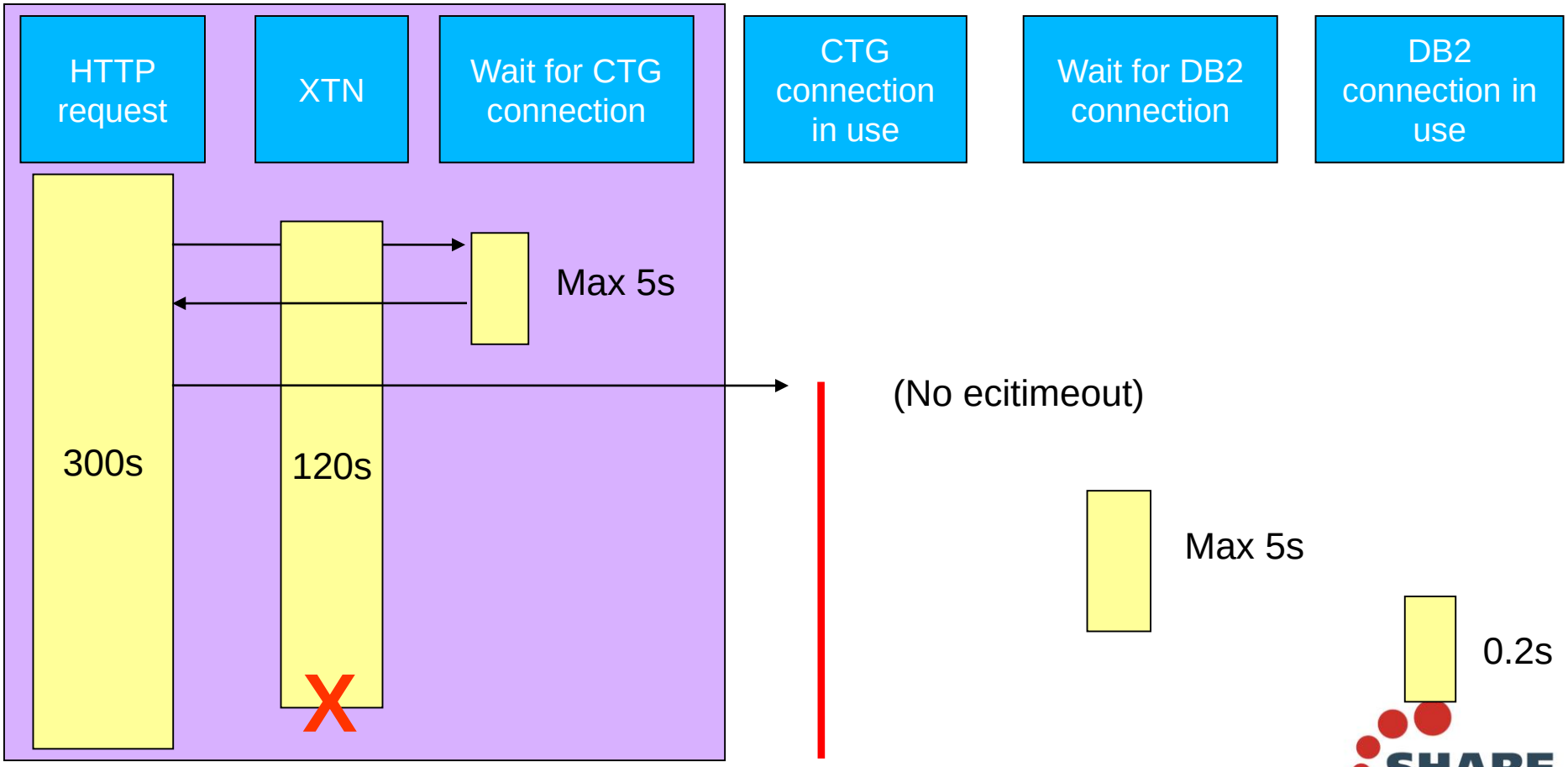
If nested timeouts are too long …

➔ Transaction (XTN) timeout will expire first if within a transaction



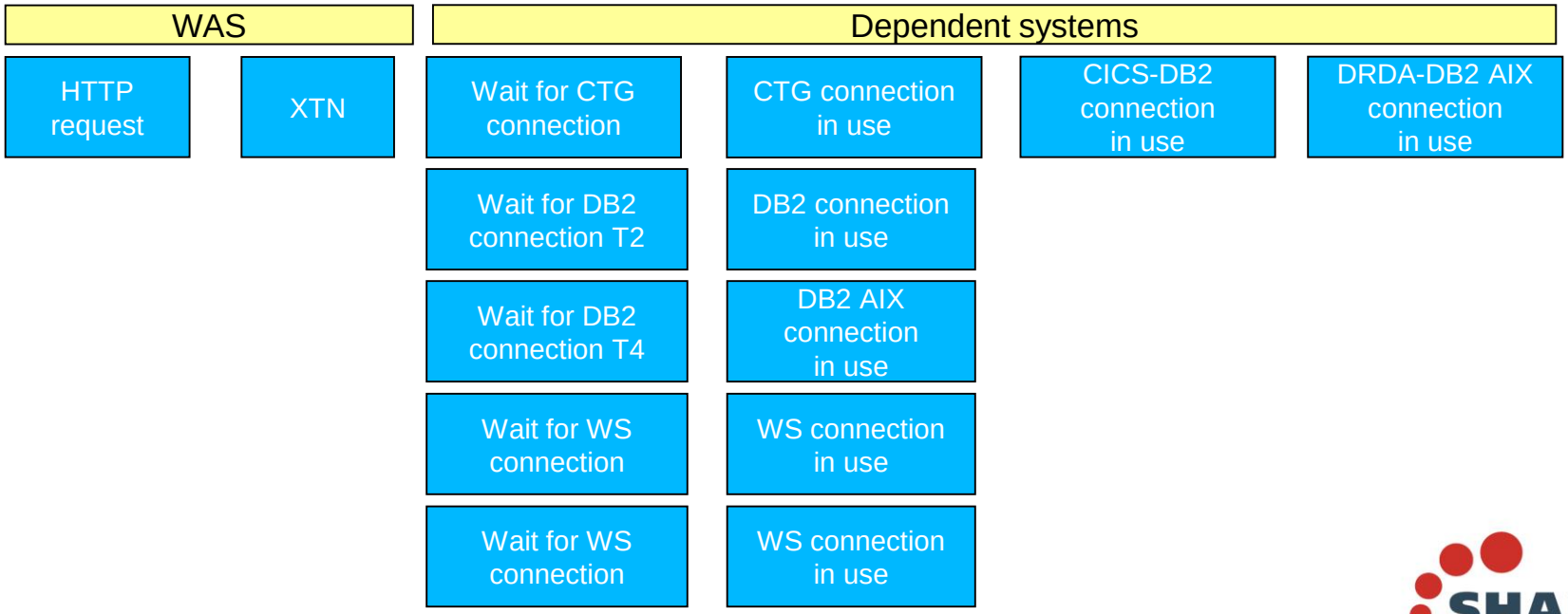
… or if there is no ‘request’ timeout …

➔ Slow-down in a remote system means you ***will*** suffer the transaction timeout or the protocol timeout (whichever is least)



Requests to dependant systems *must* have timeouts set

- **If no timeouts are set, a slow-down in a remote system means:**
  - You will suffer the WAS transaction timeout or the protocol timeout
  - Either a servant will restart or the thread will end up hung
- **Any connection to a dependent system needs:**
  - A connection timeout on the connection pool
  - A request/response timeout on the request
  - A network/socket timeout for network connections

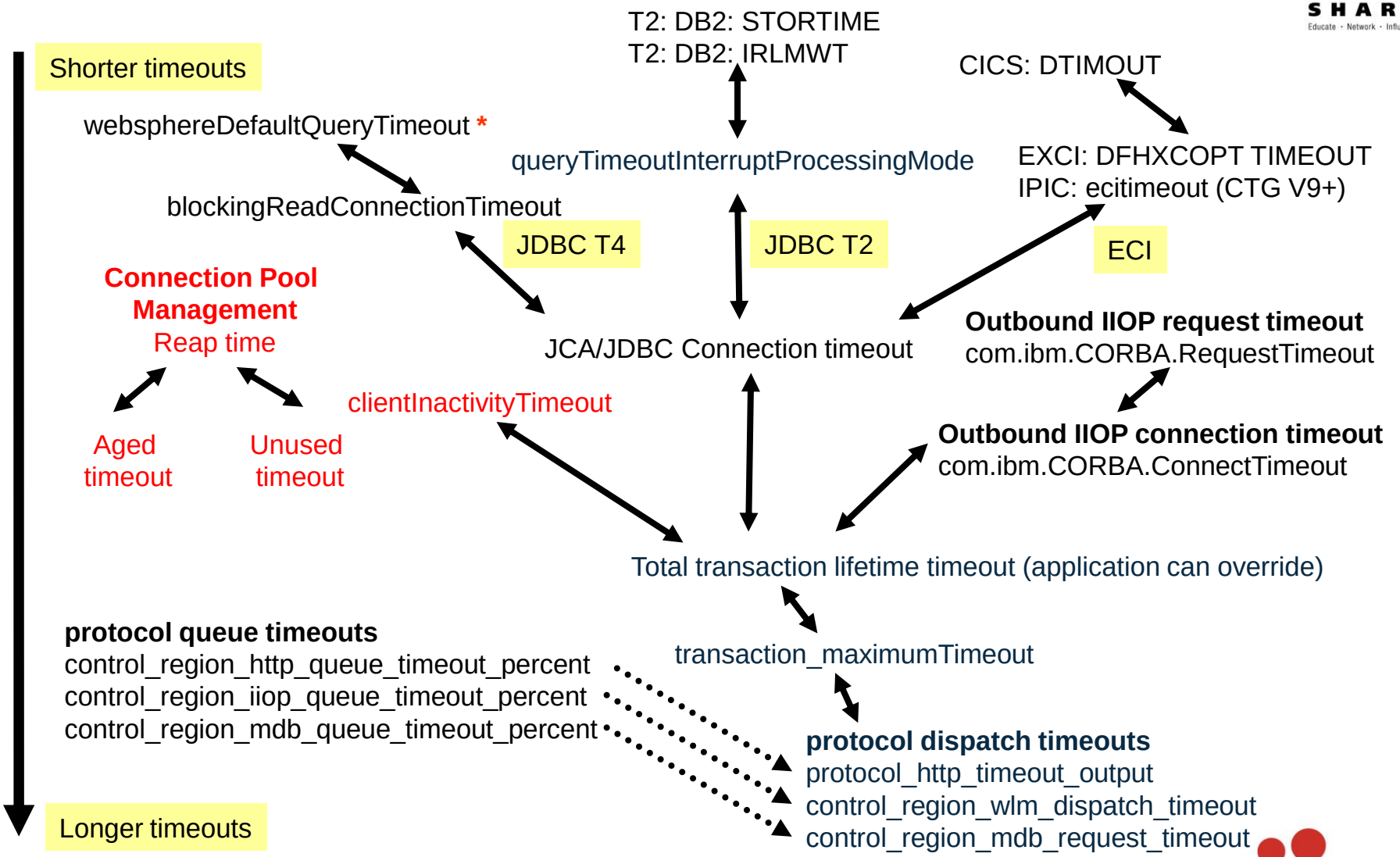


Dependent system timeouts

- Connection timeout
  - Time a request waits for a connection
  - Set by WebSphere connection manager or application
- Request/response timeout
  - Time a request that has a connection is allowed to wait for a response
  - Best set by the application but can be set by WebSphere connection mgr
- Network connections – `blockingReadConnectionTimeout`
  - Timeout a request if no response on the socket (network problems)
  - Custom property for a T4 data source
- Becomes complicated when there is a chain of connections

  `IHS > WAS > CTG > CICS > DB2 z/OS > DB2 AIX`
  - Dependent systems that don't or can't implement any timeout, or implement a timeout that is too long, just pass the problem back up the chain where it is likely to cause more problems.

Connection Pool timeouts



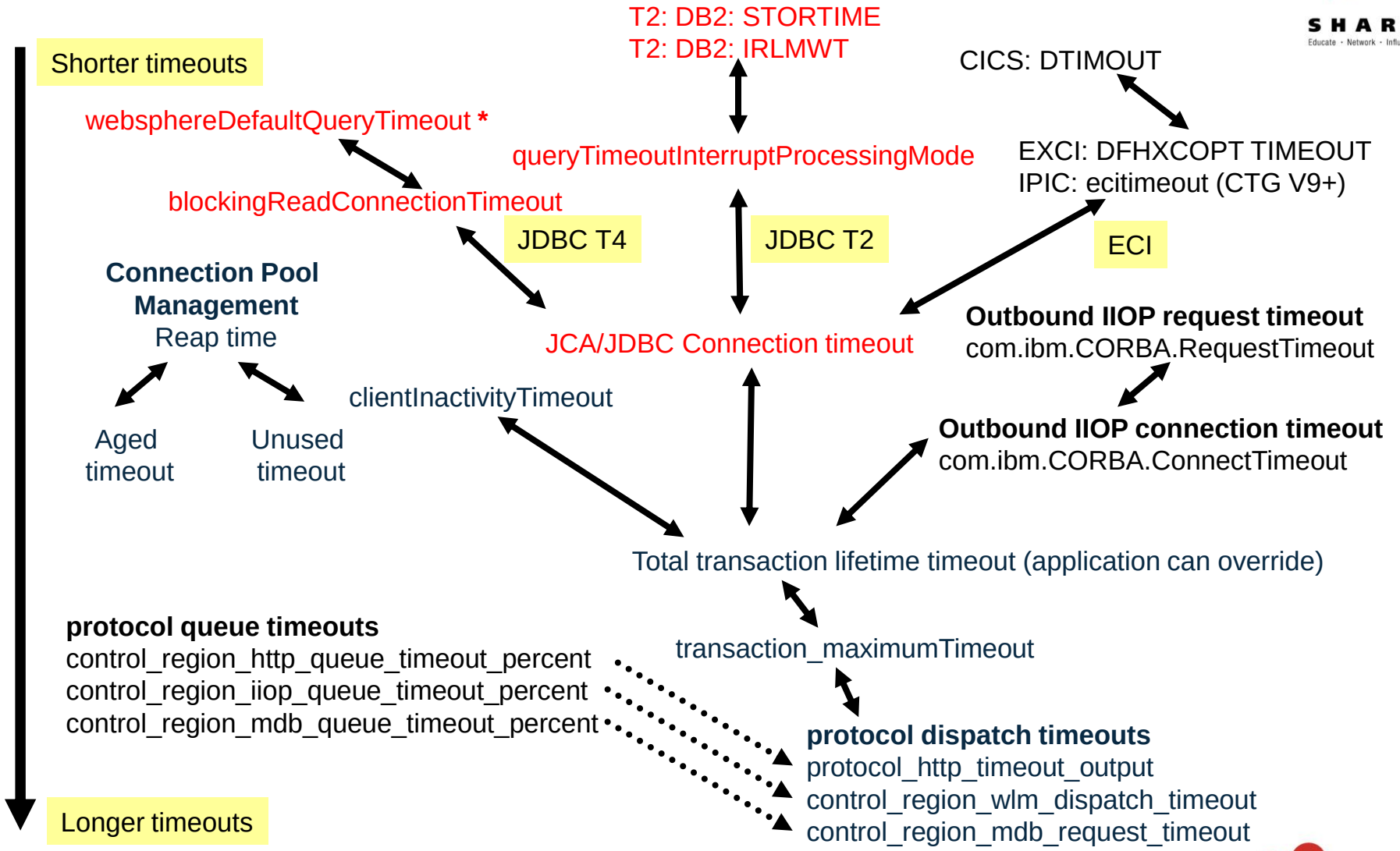
Remember: For most of these, the application can set its own timeout. These are WAS timeouts.

Timeout relationships (Connection pool)

Connection Pool Management

- Aged timeout value and Reap time not = 0
  - If the Reap time parameter has a value of zero, then the pool maintenance thread is disabled, so the Aged timeout parameter is meaningless.
- Reap time < Aged timeout
  - The Reap time parameter should be set to a value less than that of the Aged timeout parameter. The more frequently the pool maintenance thread runs, the more accurate the Aged timeout mechanism will be.
- Unused timeout value and Reap time not = 0
  - If the Reap time parameter has a value of zero, then the pool maintenance thread is disabled, so the Unused timeout parameter is meaningless.
- Reap time < Unused timeout
  - The Reap time parameter should be set to a value less than that of the Unused timeout parameter. The more frequently the pool maintenance thread runs, the more accurate the Unused timeout mechanism will be.

Accessing DB2: JDBC timeouts



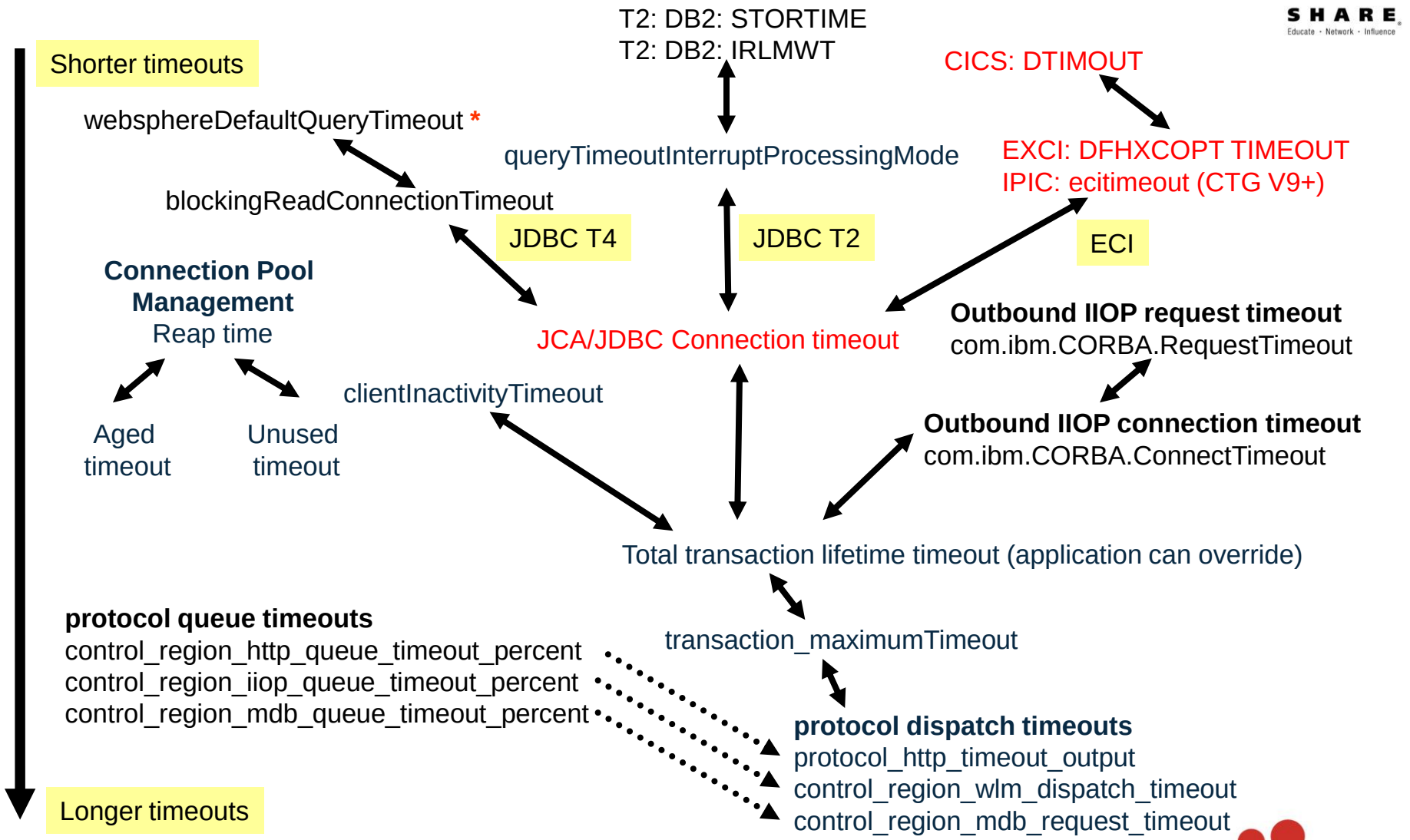
Remember: For most of these, the application can set its own timeout. These are WAS timeouts.

Timeout relationships (JDBC)

JDBC Requests

- Connection timeout < transaction_defaultTimeout
  - Connections are often acquired within a transaction context, so it is meaningless to have a request waiting for a connection for longer than the transaction is allowed to exist.
- Connection timeout > JCA/JDBC request timeouts
  - The specific timeout name depends on the nature of the request but clearly it makes no sense to wait for a connection for less time that it takes one request to finish. A connection pool might be temporarily full because of high volumes of requests but individual requests normally complete in sub-second times so request timeouts are usually very short (2 secs)
- blockingReadConnectionTimeout > websphereDefaultQueryTimeout
  - For type 4 requests, don't time out the connection at the network level before the default query timeout.

Accessing CICS



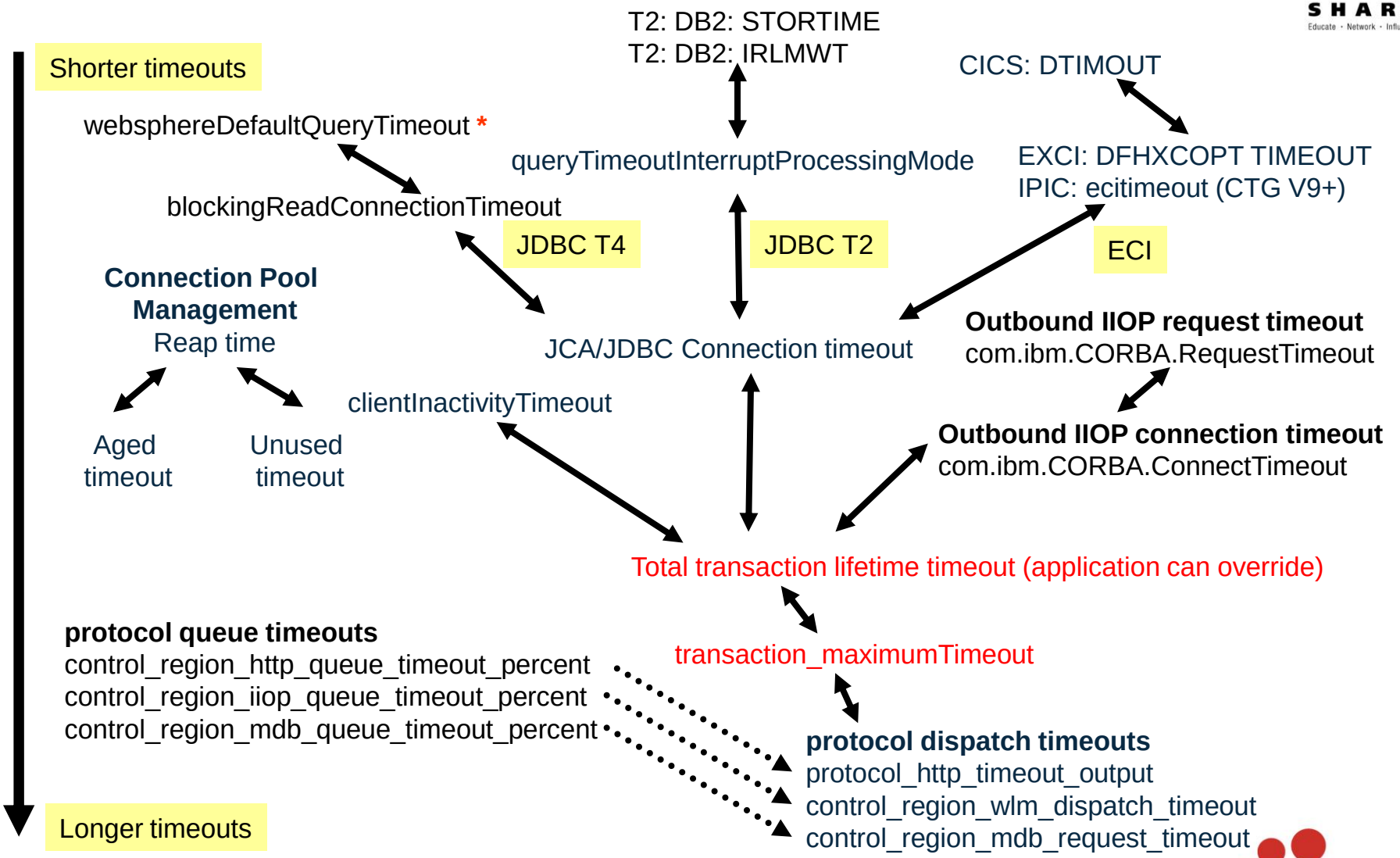
Remember: For most of these, the application can set its own timeout. These are WAS timeouts.

Timeout relationships (CICS ECI)

ECI Requests

- Request timeouts > CICS RTIMOUT/DTIMOUT (plus SPURGE=YES)
  - If the CICS transaction being invoked has RTIMOUT or DTIMOUT set, that value should not be more that the request timeout through CTG

Transaction timeouts



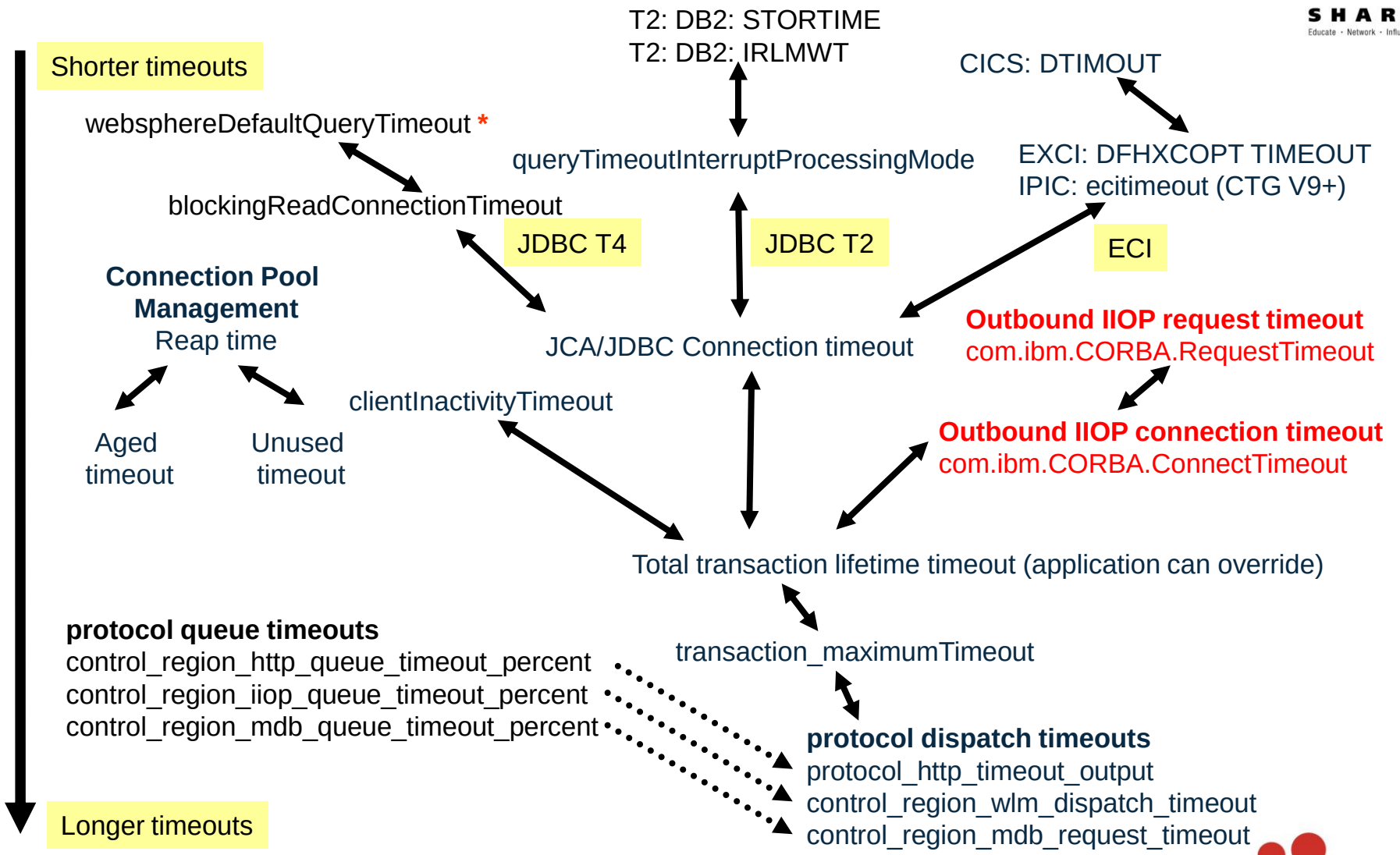
Remember: For most of these, the application can set its own timeout. These are WAS timeouts.

Timeout relationships (WAS transaction timeouts)

Transaction timeouts in relation to others

- clientInactivityTimeout < transaction_defaultTimeout
  - The clientInactivityTimeout is the maximum interval between transactional requests from a remote client. This is meaningless if it is greater than the duration of the whole transaction.
- Transaction_defaultTimeout <= transaction_maximumTimeout
  - (Transaction_defaultTimeout is the Total transaction lifetime field in the admin console)
  - The default value for a transaction timeout should not be greater than the maximum value for a transaction timeout.
- control_region_mdb_request_timeout > transaction_maximumTimeout
  - For message-driven bean requests which run transactions, the maximum transaction timeout is meaningless if it is always pre-empted by the MDB timeout.
- control_region_wlm_dispatch_timeout > transaction_maximumTimeout
  - For IIOP requests which run transactions, the maximum transaction timeout is meaningless if it is always pre-empted by the WLM dispatch timeout.
- protocol_http_timeout_output > transaction_maximumTimeout
  - For HTTP requests which run transactions, the maximum transaction timeout is meaningless if it is always pre-empted by the HTTP response timeout.
- protocol_https_timeout_output > transaction_maximumTimeout
  - For HTTPS requests which run transactions, the maximum transaction timeout is meaningless if it is always pre-empted by the HTTPS response timeout.

Outbound IIOP



Remember: For most of these, the application can set its own timeout. These are WAS timeouts.

Outbound IIOP from WAS distributed

IIOP *client* timeouts (these are for an RMI/IIOP WAS *client*)

- control_region_wlm_dispatch_timeout < com.ibm.CORBA.RequestTimeout
  - The `com.ibm.CORBA.RequestTimeout` ***in a WAS client*** determines how long the WAS client is permitted to wait for a response to a remote method dispatch in another remote WAS server.
  - If WAS distributed calls an EJB in WAS z/OS, the `com.ibm.CORBA.RequestTimeout` should not expire before any of the timers in the remote WAS z/OS, the longest of which would normally be `control_region_wlm_dispatch_timeout`.
- **In WAS distributed ...**
- A timeout will apply to any request inbound to the WAS distributed server. ('Protocol timeout' below)
- For applications that then make outbound IIOP requests to WAS z/OS (for example)
- The protocol timeout > com.ibm.CORBA.ConnectTimeout
- The protocol timeout > com.ibm.CORBA.RequestTimeout

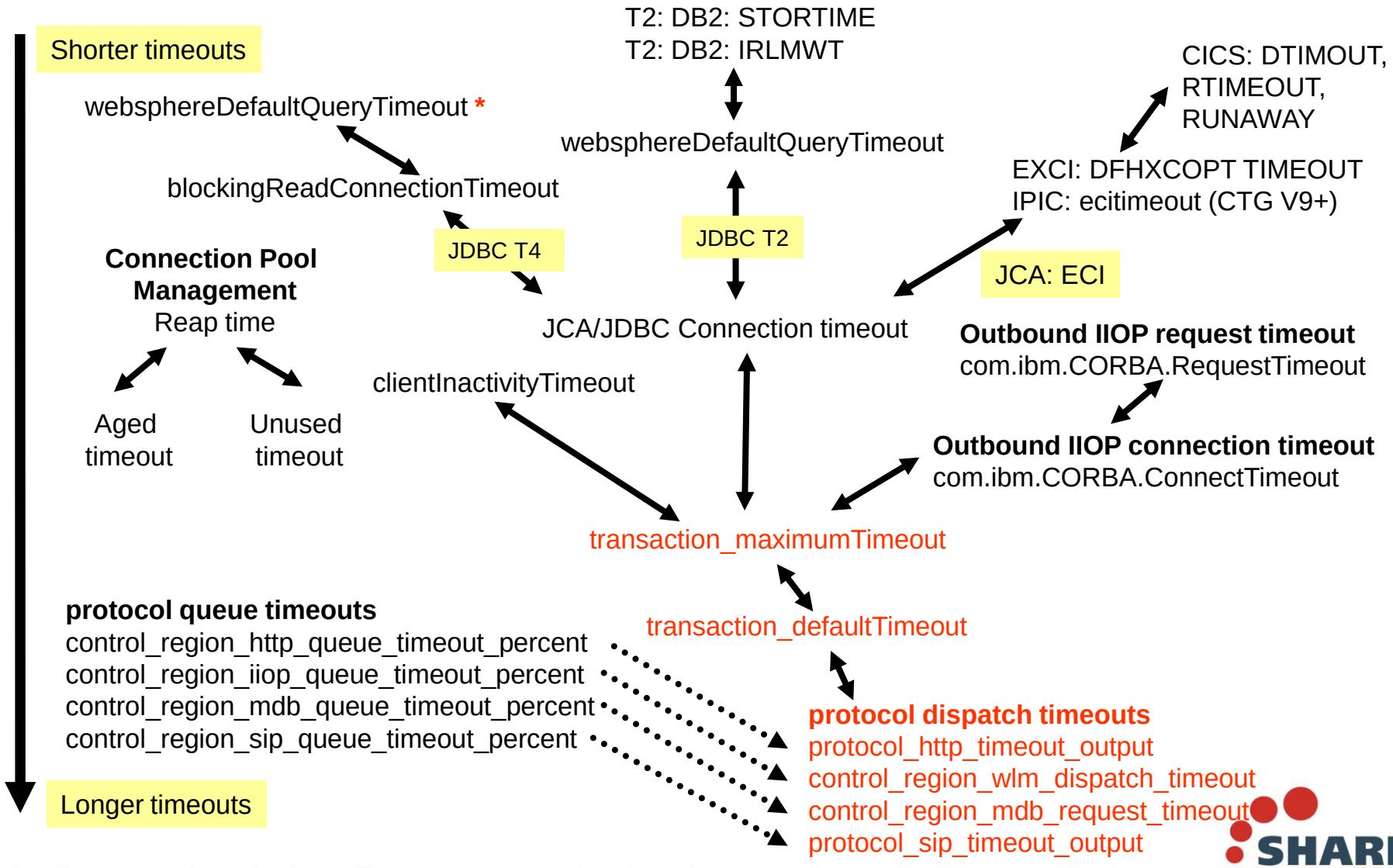
Lack of timeouts cause hung servants

- If you set large timeout values or set them to 0 to disable them, any problem in a dependent system will cause the WebSphere for z/OS servant to “hang”.
- All the WebSphere for z/OS threads will quickly become blocked waiting for a response from the remote server.

  JCA, JDBC, Web Services, IIOP, JMS …
- If the request is within a transaction, the ‘hang’ will be broken in WAS z/OS when the first request reaches the transaction timeout (120 secs)
- If the request is not within a transaction then the ‘hang’ will be broken when the protocol timeout is reached.
- If the protocol timeout is too long, or disabled by setting it to 0, then the thread will hang indefinitely.
- If all work needs access to the remote system that is not responding the whole servant will hang until either it abends or is manually cancelled.
  (Depending on the protocol recovery setting SERVANT | SESSION)

Avoiding timeouts that restart a servant

Scaling timeout values



Timeouts that can cause a servant restart

```
protocol_http_timeout_output
protocol_https_timeout_output
control_region_wlm_dispatch_timeout   (IIOP)
control_region_mdb_request_timeout
transaction_defaultTimeout
transaction_maximumTimeout
```

- Increasing these timeouts or disabling them by setting them to 0 will prevent the timeout, but the result is a thread that is hung forever
- In order to avoid timeouts that cause a servant restart, set timeouts on requests to remote systems so control is returned to the application.
- Why not just “Abend a Thread?” Actually we do, but….
  - The dispatch thread probably has Java on the call stack
  - Java has no MVS recovery (ESTAE or MVS Resource Manager)
  - Java’s signal handler can’t clean up a thread’s resources (synchronize locks remain held)
  - → Loss of a single thread would hang or corrupt the process

KnowledgeCenter rtrb_controllingtimeout

Dispatch timeouts as environment variables

Work type	DispatchTimeout variable used
IIOP	`control_region_wlm_dispatch_timeout`
HTTP	`protocol_http_timeout_output`
HTTPS	`protocol_https_timeout_output`
SIP	`protocol_sip_timeout_output`
SIPS	`protocol_sips_timeout_output`
Message Listener Port MDBs	`control_region_mdb_request_timeout`
SIBus or Activation Spec (MDBs from the CRA)	`control_region_wlm_dispatch_timeout`
WOLA (inbound to WAS)	`control_region_wlm_dispatch_timeout`
Other (Message Listener Port in the SR, Mbeans, other internal work)	`control_region_wlm_dispatch_timeout`

The lowest level of granularity of these properties is a server

KnowledgeCenter rtrb_controllingtimeout

WAS V8 RAS Function Control Down to Request

Granular RAS leverages the existing workload classification file to extend scope of certain RAS functions down to the request level:

z/OS only

Not just HTTP … any classifiable input type (IIOP, MDB, etc.)

```
<http_classification_info transaction_class="_____"
    .
dispatch_timeout="_____"
queue_timeout_percent ="_____"
request_timeout="_____"
stalled_thread_dump_action="_____"
cputimeused_limit="_____"
cputimeused_dump_action="_____"
dpm_interval="_____"
dpm_dump_action="_____"
SMF_request_activity_enabled="__"
SMF_request_activity_timestamps="__"
SMF_request_activity_security="__"
SMF_request_activity_CPU_detail="__"
classification_only_trace="__"
message_tag="_____"
timeout_recovery="_____">
```

Additional XML values permitted in the WLM classification file

These control behavior when classification identification is made by this function

MODIFY command allows you to dynamically enable a new file, or dynamically revert back to previous

Enhances granularity. Previously down to server level; now to request.

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102023

Avoiding servant restarts – IIOP requests

- Avoid inbound IIOP requests causing servant abend after startup
  - HTTP work can't enter the controller until MINSRS have initialized but IIOP work can be queued to the controller before servants are ready
  - Risk that IIOP requests queue for so long that they timeout when the servant is up.
    Avoid this by setting the following property to reject IIOP requests until the servants are up:
    `protocol_accept_iiop_work_after_min_srs=1`
- Avoid outbound IIOP client requests causing WAS timeouts that abend servants
  - Set the timeout for establishing IIOP connections to less than the WLM dispatch timeout or less than the transaction timeout if the request is within a transaction.

Servers > Application servers > <server> > Container services > ORB service > Custom properties

`com.ibm.CORBA.ConnectTimeout`

- Set the outbound IIOP request timeout to less than the WLM dispatch timeout or less than the transaction timeout if the request is within a transaction.

Servers > Application servers > <server> > Container services > ORB service

  - Set the **Request timeout** field. When the timeout expires, the application that issued the request receives an `org.omg.CORBA.COMM_FAILURE` system exception which it should handle.
    `com.ibm.CORBA.RequestTimeout`

Infocentre urun_rproperty_custproperties

Avoiding servant restarts – HTTP requests

- The old way to avoid inbound HTTP requests causing servant restarts

```
protocol_http_timeout_output_recovery=SESSION
protocol_https_timeout_output_recovery=SESSION
```

  - Sends a response to the client and closes the HTTP session but does not restart the servant.
  - The request that has timed out is allowed to carry on running.
  - Risk is that all servant threads become blocked if all requests are waiting for the same reason (problem with a remote system)
- **Since WAS z/OS V7 it's better** to set SERVANT and use stalled thread threshold and/or percent queue timeouts to reject requests that have been queued for too long.
- Don't disable the protocol timeout by setting it to 0
  - Disabling the protocol timeout also prevents servant restarts but doing that disables the property protocol_http_timeout_output_recovery=SESSION
  - If requests never timeout the user just hangs for ever.

  (Unless the request is within a transaction and the transaction timeout is exceeded)

KnowledgeCenter rtrb_controllingtimeout

Avoiding servant restarts – DB2 requests

Set connection and query timeouts to avoid a WAS transaction timeout

Connection timeout

- For WebSphere managed connections:
  - Set the connection timeout on the Data source connection pool properties
- For an unmanaged connection

  `javax.sql.CommonDataSource setLoginTimeout`

Request timeout

`java.sql.Statement.setQueryTimeout` ← this is how the application might set the request timeout

- WAS 7.0.0.21+ T4 JDBC V4 driver connections set data source custom properties
  http://www.ibm.com/support/docview.wss?uid=swg1PM38864

  `blockingReadConnectionTimeout`

  `websphereDefaultQueryTimeout`
- The following is probably not useful. No point to sync with the transaction timeout.

  `syncQueryTimeoutWithTransactionTimeout`
- For T2 DB2 z/OS, setQueryTimeout is supported only with dataSource property:

`queryTimeoutInterruptProcessingMode=INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET`

KnowledgeCenter tdat_querytimeout

KnowledgeCenter imjcc_rjvjdapi.htm

KnowledgeCenter imjcc_r0021073.htm

Stalled thread threshold

- Prior to WAS V7 a dispatch timeout always caused a servant restart
  - The exception being `protocol_http_timeout_output_recovery=SESSION`
- SInce WAS z/OS V7 there are now two dispatch timers:
  - A new servant timer
  - The existing controller timer
- The servant timer can schedule an ODI (Interruption Object) which attempts to unblock the thread. There are several ODIs for different situations.
- The existing controller timer is extended to allow the ODI time to work
- The ODI might not succeed and ultimately WAS may have to ‘give up’
  1. Application performs some operation that causes an ODI to be registered
  2. Application hangs
  3. Servant timer "pops" (expires) and the timer finds and drives the registered ODI
  4. ODI wakes up the thread, which then causes the ODI to be deregistered.
  5. Application receives an exception from the operation (a likely occurrence).
  6. Application retries the requests … and then hangs …
- WAS gives up trying to unblock the request. The thread is stalled.

Stalled thread threshold diagnostics and threshold

- When a servant gives up on a stalled thread, it can gather diagnostics

```
server_region_xxxx_stalled_thread_dump_action=
 SVCDUMP|JAVACORE|HEAPDUMP|TRACEBACK|JAVATDUMP|NONE
```

  where xxxx is the type of request: http, iiop, mdb, https, sip, or sips.

- After the diagnostics have been taken, the controller is notified and decides whether to restart the servant.
- The controller checks the stalled thread threshold

```
server_region_stalled_thread_threshold_percent
```

- This property sets the percentage of servant threads you will tolerate in a stalled state before the servant should be restarted

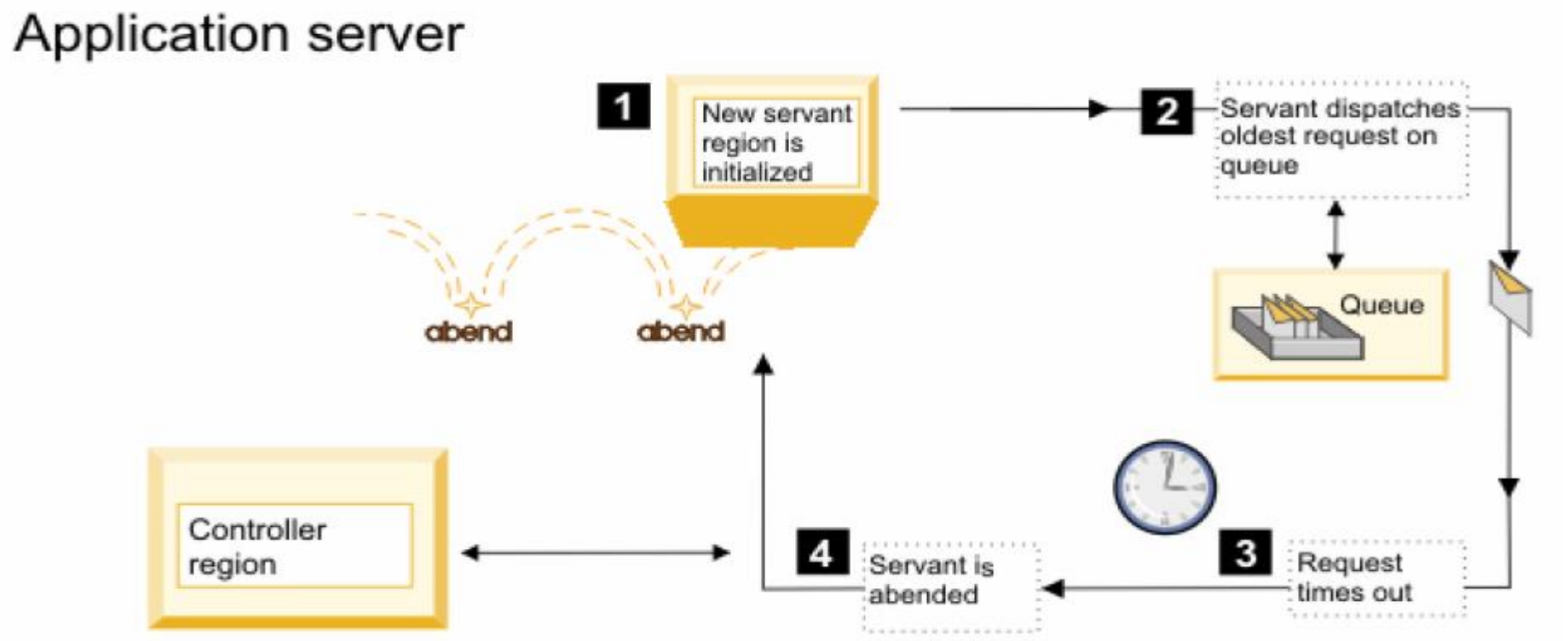
```
protocol_http_timeout_output_recovery=SESSION
```

will disable stalled thread threshold

  - Consider changing the value to SERVANT, which is the default. Depending on the cause of the problem, an ODI scheduled by the new servant timeout function might be able to free the problem requests without a servant restart,

KnowledgeCenter urun_rproperty_custproperties

The bouncing servant problem



0. Servant suffers a protocol timeout so abends and restarts
1. Servant starts and processes work but …
2. Requests that have been queued for a long time are dispatched and …
3. Requests timeout before they have time to run which causes …
4. Another servant restart

How to reject work that can't run

control_region_xxxx_queue_timeout_percent

xxxx is the protocol: http, https, iiop, mdb.

- How long do you want a request to wait for dispatch? 1 minute? Less?
- Set as a percentage of the underlying protocol dispatch timeout
  - `protocol_http_timeout_output`
  - `protocol_http_timeout_output`
  - `control_region_wlm_dispatch_timeout` (for IIOP requests)
  - `control_region_mdb_request_timeout` (for MDB requests)
- The queue timeouts prevent:
  - Work from waiting in the WAS dispatch queue even if it can't run.
  - Servant restarts caused by requests that time out immediately after dispatch.
  - 'Sympathy sickness' where a hung WAS might lead to a hung HTTP Server
- If the protocol dispatch timeout is disabled (set to 0) then the queue timeouts can not be set either, therefore don't set these to 0.
- Default values: `control_region_xxxx_queue_timeout_percent=0` (disabled). Therefore no queue timeouts are used unless you set them.

KnowledgeCenter urun_rproperty_custproperties

HTTP timeout messages in the controller

- When running with **no queue timeouts and protocol_http_output_recovery=SERVANT**
- `BBOO0327I HTTP REQUEST TIMEOUT: (01B7):(FCE2323D):(0001001D):(009AF950):(S0014849):(2013/10/15 10:42:17.075128):2013/10/15 10:42:17.075317):(2013/10/15 10:45:43.580729):(RemoteWebContainer):(httprequest):(ip addr=10.7.14.11 port=50765):():():(/CoProcWeb/coproc2Service)`
- `BBOO0232W A request for Class Name 'RemoteWebContainer' and Method Name 'httprequest', from ip addr=10.7.14.11 port=50765, has timed out. The servant process associated with the request will be terminated. Request Id(FCE2323D)`

2013/10/15 10:42:17.075128 Enters controller
2013/10/15 10:42:17.075317 On WLM queue
2013/10/15 10:45:43.580729 Time in dispatch

- When running with **http queue timeouts and protocol_http_output_recovery=SERVANT**
- `BBOO0327I HTTP REQUEST TIMEOUT: (0000):(FCE02753):(00010005):(00000000):():(2013/10/15 16:22:02.168402):(2013/10/15 16:22:02.168699):():(RemoteWebContainer):(httprequest):(ip addr=10.7.14.2 port=60417):():():(/CoProcWeb/coproc2Service)`

2013/10/15 16:22:02.168402 Enters controller
2013/10/15 16:22:02.168699 On WLM queue
() Time in dispatch

- The ***absence of any time in dispatch*** says that this request was timed-out while waiting for dispatch. Also, since the request was never dispatched, the TCB address is zeros and the servant STC number is blank.

Queue timeouts as environment variables

Work type	Queue Timeout variable used
IIOP	`control_region_iiop_queue_timeout_percent`
HTTP	`protocol_http_queue_timeout_percent`
HTTPS	`protocol_https_queue_timeout_percent`
SIP	`protocol_sip_queue_timeout_percent`
SIPS	`protocol_sips_queue_timeout_percent`
Message Listener Port MDBs	`control_region_mdb_queue_timeout_percent`
SIBus or Activation Spec (MDBs from the CRA)	`control_region_wlm_queue_timeout_percent`
WOLA	`control_region_wlm_queue_timeout_percent`
Other (Message Listener Port in the SR, Mbeans, other internal work)	`control_region_wlm_queue_timeout_percent`

Transaction timeouts

Total Transaction Lifetime Timeout - transaction_ defaultTimeout

- Transaction timers and the protocol output timers are not aware of each other.
- **Total Transaction Lifetime Timeout** can be overridden by applications up to the Maximum Transaction Timeout variable. Default value: 120 secs
- **Maximum Transaction Timeout**
- Limits the amount of time an application can set for its transactions to complete. If the Maximum Transaction Timeout variable is not set, application transactions are controlled by the time limit set on the Total Transaction Lifetime Timeout. Default value: 900 secs

Avoiding transaction timeouts and their impacts

- Only use a transaction when one is needed
  - Sometimes a transaction might be started by default
- Timeout requests before they reach the transaction timeout. Ensure connections to dependent system implement
  - A connection timeout
  - A request timeout which is less than the transaction timeout
- If the transaction timeout is defaulting to 120 secs, you should reduce all connection pool timeouts from the default of 180 to less than 120 secs.
- Avoid setting the transaction timeouts to a very high value
  - Resources are locked until the timeout occurs.
  - It might cause a more serious data contention problem.

Summary of Defensive Actions

WAS z/OS timeout management

- Instead of immediately restarting a servant after a timeout, WAS z/OS will try to get the request to complete by scheduling an ODI.
- It is always important to set protocol timeouts appropriately
- If in WAS V6 timeout values were set to very high values or disabled by setting them to 0, you should review this and probably set lower values in V7 or later
- Use the Service Level Agreement to help decide values for protocol timeouts
- Review all DUMP_ACTION environment variables
- Consider setting a CPU timeout but be careful!
- Instead of the following which you may have coded in WAS V6 …

```
protocol_http_timeout_output_recovery=SESSION
protocol_http_timeout_output=0    (0=disabled)
```

- … use …

```
protocol_http_timeout_output=300     (the default, for example)
protocol_http_timeout_output_recovery=SERVANT   (the default)
control_region_http_queue_timeout_percent=3  (for example … 3%x300=10
   secs)
server_region_stalled_thread_threshold_percent=75  (for example)
```

Standard steps (1)

- Block your server from accepting IIOP requests until MINSRS are started.
  (By default, HTTP work is not allowed to run until MINSRS are started.)

  ```
  protocol_accept_iiop_work_after_min_srs=1
  ```
- Protect your server from outbound IIOP timeouts

  ```
  com.ibm.CORBA.ConnectTimeout=nn
  com.ibm.CORBA.RequestTimeout=nn
  ```

  … where nn < transaction_defaultTimeout and WLM dispatch timeout
- Set reasonable dispatch timeouts and queue percent timeouts
- Stop work being accepted if you lose all servants

  ```
  control_region_dreg_on_no_srs=1
  ```
- If the cause is usually a long-duration problem in a dependent system, consider manually controlling when listeners should be opened.

  ```
  control_region_confirm_recovery_on_no_srs=1
  ```

Standard steps (2)

- Ensure datasource and connection factory connection timeout is changed from the default value of 180 seconds to less than 120 secs (the default transaction timeout). A value like 10 or 15 secs is reasonable
- Review all outbound connections from WAS to any dependent system and set timeout values on the datasource or connection factory that will prevent tasks waiting forever for network issues or no response to requests.
- Remember that there are typically 3 types of timeouts for a remote request:
  1. A timeout when waiting for a thread in a connection pool
  2. A timeout at network level for 'remote' requests over TCPIP
  3. A timeout for the request
- Encourage developers to set requests timeouts in their application because these are more accurate than datasource or CF level timeouts.
- Encourage developers to use WebSphere Connection Manager rather than un-managed connections

Summary

- Timeout management is complicated, especially in a multi-tier topology
- The effects of changing a timeout could require others to be adjusted
- Practices for WAS z/OS V6 or older are very different from WAS z/OS V7+
- Mostly the defaults are not too bad (apart from the connection pool timeout)
- Be sure to set reasonable timeouts on all outbound requests from WAS
- If work can't run, don't allow it to come in
- WAS z/OS has some unique work management features
- WAS z/OS has unique granularity for trace and timeout settings at *request-type* level
- For monitoring, z/OS and WAS z/OS provide some useful displays
- Be familiar with the technique that uses D OMVS,PID= and Javacores. (WP101474)
- Monitor products like ITCAM are more powerful especially the ability to react to thresholds or monitor the ERRORLOG logstream and alert to an central enterprise monitoring console (e.g. IBM Tivoli Enterprise Portal)
- Any monitoring/alert solution requires quite a lot of careful customization and engaging an expert is probably money well spent. It will not do everything you want out-of-the-box.
- Default OMVS properties are usually OK for WAS z/OS since z/OS 1.12

Monitoring / Diagnosis

Monitoring / diagnostics tools

- Alert on BBOO0327I message in the controller.
  - Use `ras_message_routing_copy_console=BBOO0327I` to route timeout messages to the console for action by NETVIEW.
  - Use ITCAM to monitor WAS z/OS ERRROLOG logstream for `BBOO0327I`
  - ITCAM ORB thread monitor and response time alerts

- Set timeouts in the application and handle
- Query the state of threads:
  - D OMVS,PID= with F <server>,JAVACORE
  - F < server >,DISPLAY,WORK
  - F <server>,DISPLAY,THREADS
  - Dispatch Progress Monitor (DPM)
- WLMQUE tool
- Monitor products like ITCAM or Intrascope
- For HTTP Server, Apache mod_status ihs_host:port/server-status

D OMVS,PID= with F <server>,JAVACORE

- Possibly more for diagnosis than monitoring
- Useful for intermittent hangs or high CPU
- D OMVS,PID=
  - Can be issued from the SDSF **PS** screen using the **'d'** line command
  - Most useful when issued repeatedly (10-15 secs) and with JAVACORE

```
BPXO040I 11.09.56 DISPLAY OMVS 545
OMVS     000F ACTIVE              OMVS=(00,FS,0A)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
WSASRU   WSSR01S  01DF   33554734          1 HR---- 06.46.26  1950.574
THREAD_ID        TCB      PRI_JOB  USERNAME   ACC_TIME SC  STATE
. .
268E0D0000000068 009BC0D0 WLM                   12.886 CLO JR V
268E585000000069 009B8E88                       43.288 STE JR V
. .
```

- Use F <server>,JAVACORE together with D OMVS,PID=
  - The JAVACORE tells you what is running
  - Take repeated D OMVS,PID= and JAVACOREs
  - There are tools that can compare JAVACOREs

F <server>,DISPLAY,WORK

- Consider automating this command

  `F <server>,DISPLAY,WORK`
- It shows how many requests are in dispatch and total requests.
- It shows the DELTA since the last command.
- Automation could alert when there's high thread usage (a high value for BBOO0262I and high queuing (a high value for BBOO0263I)
- Automation could alert if there are spikes (high DELTA)

```
f <server>,display,work,summary
BBOO0255I TIME OF LAST WORK DISPLAY Wed Jan 3 19:18:38 2008
BBOO0261I TOTAL REQUESTS TO SERVER     173591    (DELTA 13944)
BBOO0262I TOTAL CURRENT REQUESTS       0
BBOO0263I TOTAL REQUESTS IN DISPATCH   0
BBOO0268I TOTAL TIMED OUT REQUESTS     0     (DELTA 0)
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,WORK,SUMMARY
```

KnowledgeCenter

rtrb_usingzosmodify

F <server>,DISPLAY,THREADS

- Consider automating this command. (Like this it shows SUMMARY)

```
F <server>,DISPLAY,THREADS
```

- The SUMMARY shows the status of threads in each servant

```
BBOJ0111I: REQUEST  ASID JW TO RE DISPATCH  TIME
BBOJ0112I: ffffe453 0X0041 Y N N 2008/04/14 18:38:12.391628
BBOJ0112I: ffffe452 0X0041 N N N 2008/04/14 18:38:27.473191
BBOJ0112I: ffffe454 0X0041 Y N N 2008/04/14 18:38:12.319306
BBOJ0112I: ffffe451 0X003C N N N 2008/04/14 18:38:27.485103
```

Request and ASID identify the request and address space ID

JW Is the thread in a wait?

TO Has the thread timeout?

RE Has WAS given up trying to unblock the request?

- If work is moving normally, this display should change each time it is issued. If you always see lots of the same requests in JW a hang is starting
- A high RE compared to the stalled thread threshold warns of servant restart
- There is a JMX MBean interface that can provide the same information but this may not be useful for cases where all threads are blocked

KnowledgeCenter

rxml_mvsdisplay

F <server>,DISPLAY,THREADS with filters

- How many threads have been in dispatch more than 10 seconds?

  `F <server>,DISPLAY,THREADS,AGE=10`
- How many threads have timed-out?

  `F <server>,DISPLAY,THREADS,TIMEDOUT,SUMMARY`

  `F <server>,DISPLAY,THREADS,TIMEDOUT,DETAILS`
- Details of a particular thread?

  `F <server>,DISPLAY,THREADS,REQUEST=FFFFE4F3`

```
BBOJ0106I: REQUEST ffffe453 ASID 0X0041 TCB 0X008CAE88
BBOJ0119I: CONTROLLER RECEIVED REQUEST AT 2008/04/14 18:38:12.391478
BBOJ0120I: CONTROLLER QUEUED REQUEST TO WLM AT 2008/04/14 18:38:12.391522
BBOJ0107I: SERVANT DISPATCHED REQUEST AT 2008/04/14 18:38:12.391628
BBOJ0108I: JVM THD IS HUNG: ITI INACTIVE
BBOJ0110I: DETAILS FOR JVM INTERRUPTIBLE THREAD: Monitor ACTIVE
```

- ITI INACTIVE means WAS has not yet tried to unblock the request
  ITI ACTIVE means WAS is unblocking and ENDED means its given up.
  - Other ODIs might present different information, for example:
- `BBOJ0110I: DETAILS FOR OTS: URID c25926477e427000000000004201010000`
- `BBOJ0110I: DETAILS FOR GIOP Outbound: Target Operation = resolve_complete_info`

KnowledgeCenter

rxml_mvsdisplay

Dispatch Progress Monitor (DPM)

- How to identify the requests that take longer than average in RMF reports?
- The Dispatch Progress Monitor …
  - Is enabled for a protocol using the MODIFY command
  - When a request for the protocol is dispatched a timer starts
  - If the request is still running when the timer expires, BBOJ0118 is issued with information about the request and diagnostics are gathered
  - The timer is reset for the configured interval
- To enable the DPM …

  `F <server>,DPM,xxxx=<interval>`

  where xxxx is the type of request: HTTP, IIOP, MDB, HTTPS, SIP, SIPS and interval is the interval value you want the DPM to use (0 to 255 secs)
- To disable all DPM monitoring

  `F <server>,DPM,CLEARALL`

KnowledgeCenter

tprf_monitor_dispatch_requests

Dispatch Progress Monitor (DPM) diagnostics

- Set the following WebSphere environment variable to specify the diagnostics to collected when the DPM timer expires:

  ```
  server_region_dpm_dump_action=
  SVCDUMP|JAVACORE|HEAPDUMP|TRACEBACK|JAVATDUMP|NONE
  ```

- The diagnostics collected can be changed using the modify command:

  ```
  F <server>,DPM,DUMP_ACTION=JAVACORE
  ```

- Display the state of a DPM using:

  ```
  F <server>,DISPLAY,DPM
  ```

```
BBOO0361I DISPATCH PROGRESS MONITOR (DPM) SETTINGS:
IIOP(000):HTTP(010):HTTPS(015):MDB(000):SIP(020):SIPS(000),
DUMP_ACTION(JAVATDUMP)
```

Dispatch Progress Monitor (DPM) example

The following is an example of the output from BBOJ0118

```
BBOJ0118I: ThreadDetails: ASID = 005B, TCB = 0X008CBE88, Request = fffff503,
Is JVM Blocked = false, Tried to interrupt = false, Given up = false,
Internal Work Thread = false, Hung Reason = Not Hung,
SR Dispatch Time = 2008/05/05 12:15:31.371625,
CTL Receive Time = 2008/05/05 12:15:31.366693,
CTL Queued to WLM Time = 2008/05/05 12:15:31.371328,
Details = , ODI Details = .JVM INTERRUPTIBLE THREAD, Monitor ACTIVE.

BBOJ0117I: JAVA THREAD STACK TRACEBACK FOR THREAD WebSphere:ORB.thread.pool
t=008cbe88: Dispatch Progress Monitor
Traceback for thread WebSphere:ORB.thread.pool
t=008cbe88:com.ibm.ws390.orb.ClientDelegate.invokeRequestCFW(Native Method)
com.ibm.ws390.orb.ClientDelegate.commonInvoke(ClientDelegate.java:998)
com.ibm.ws390.orb.ClientDelegate.invoke(ClientDelegate.java:845)
org.omg.CORBA.portable.ObjectImpl._invoke(ObjectImpl.java:484)
com.ejb.test.hello.HelloSecondHome_Stub.create(HelloSecondHome_Stub.java:207)
com.ejb.test.hello.HelloFirstBean.sayHelloOne(HelloFirstBean.java:76)
com.ejb.test.hello.EJSRemoteStatelessSayHelloFirst_67c1d243.sayHelloOne
(EJSRemoteStatelessSayHelloFirst_67c1d243.java:41))
```

KnowledgeCenter tprf_monitor_dispatch_requests

WAS V8 Trace filtering

```
<InboundClassification type="http"
          schema_version="1.0" default_transaction_class="M">
   <http_classification_info transaction_class="N"
          host="host.company.com">
     <http_classification_info transaction_class="Q
            uri="/gcs/admin"  classification_only_trace="1"/>
     <http_classification_info transaction_class="R"
            uri="/gcs/admin/1*"/>
   </http_classification_info>
</InboundClassification>
```

After activating this classification XML, enable the trace you want.

F <server>,TRACEJAVA='com.ibm.ws.security.*=all'

The server will trace only threads that match the URL in the classification XML

WAS V8 Timeout recovery action

In Version 7 there was a set of environment variables that controlled the recovery action that WAS z/OS was to take when a timeout occurred:

```
protocol_http_timeout_output_recovery  SERVANT | SESSION
protocol_https_timeout_output_recovery SERVANT | SESSION
protocol_sip_timeout_output_recovery   SERVANT | SESSION
protocol_sips_timeout_output_recovery  SERVANT | SESSION
```

In WAS V8 the recovery action can be set per request in the classification XML

Note: This option is only available for HTTP/HTTPS and SIP/SIPS because an IIOP request flow does not allow a client to get a response while a request is running.
It confuses the transactional logic.
MDBs do not really have a response so the option just does not apply in any of the MDB flows.

WLM classification refresh

- Use the modify command to dynamically activate changed trace or timeout settings in the WLM classification XML file.

```
F <server>,RECLASSIFY
F <server>,RECLASSIFY[,FILE=['/path/to/file.xml']]
```

KnowledgeCenter

rxml_mvsmodify

Other notes

- JAVA_TDUMP_PATTERN should not be left to default else TDUMPs will try to create a dump dataset with HLQ equal to the started task userid. That userid does not normally have an ICFCAT alias so an attempt will be made to catalog the TDUMP dataset in the master catalog. That usually fails due to ICH408I and it will bring down the address space.
- For datasources, the custom properties you get depend on the datasource type when the datasource was created.
  In a T2 datasource, for example, you won't find `blockingReadConnectionTimeout` listed in the custom properties.

Agenda

- WAS timeouts
  - WAS timeout parameters (System z, AIX)
  - Timeout chain from Client to DB2
  - Monitoring and reacting on timeouts

- **Other limits (e.g. OMVS limits like MAXPROCUSER)**

OMVS properties

- SYS1.PARMLIB(BPXPRMxx) has the IPL values for OMVS
- The D OMVS,O command displays the active values
- The D OMVS,L command shows you property limits
- The SETOMVS command can change values dynamically
- Some values can be overridden in the RACF OMVS segment of a userid

D OMVS,O

```
BPXO043I 21.58.46 DISPLAY OMVS 112
OMVS     000F ACTIVE              OMVS=(SP)
CURRENT UNIX CONFIGURATION SETTINGS:
MAXPROCSYS       =       1000    MAXPROCUSER      =        200
MAXFILEPROC      =     131072    MAXFILESIZE      = NOLIMIT
MAXCPUTIME       = 2147483647    MAXUIDS          =        100
MAXPTYS          =        256
MAXMMAPAREA      =       4096    MAXASSIZE        = 2147483647
MAXTHREADS       =       5000    MAXTHREADTASKS   =       2000
MAXCORESIZE      =    4194304    MAXSHAREPAGES    =     131072
IPCMSGQBYTES     =     262144    IPCMSGQMNUM      =      10000
IPCMSGNIDS       =        500    IPCSEMNIDS       =        500
IPCSEMNOPS       =         25    IPCSEMNSEMS      =         25
IPCSHMMPAGES     =      76800    IPCSHMNIDS       =        500
IPCSHMNSEGS      =         10    IPCSHMSPAGES     =     262144
SUPERUSER        = BPXROOT       FORKCOPY         = COW
STEPLIBLIST      =
USERIDALIASTABLE=
PRIORITYPG VALUES: NONE
PRIORITYGOAL VALUES: NONE
MAXQUEUEDSIGS    =       1000    SHRLIBRGNSIZE    =   67108864
SHRLIBMAXPAGES   =       4096    VERSION          = REMVS1D7
SYSCALL COUNTS   = NO            TTYGROUP         = TTY
SYSPLEX          = YES           BRLM SERVER      = N/A
LIMMSG           = NONE          AUTOCVT          = OFF
RESOLVER PROC    = RESOLVER      LOSTMSG          = ON
AUTHPGMLIST      = NONE
SWA              = BELOW         NONEMPTYMOUNTPT = NOWARN
SERV_LINKLIB     =
SERV_LPALIB      =
ALTROOT          =
MAXUSERMOUNTSYS =          0     MAXUSERMOUNTUSER=          0
```

WAS z/OS suggestions:

`MAXTHREADS: >= 10000`
`MAXTHREADTASKS: >= 5000`
`MAXFILEPROC: >= 10000`
`MAXSHAREPAGES: > 38400`

Note: For 64-bit JVMs the amount of above-the-bar memory is limited by property `MEMLIMIT` in `SMFPRMxx`.

Set MEMLIMIT to at least the size of the MAXHEAP but typically 2GB

KnowledgeCenter

tins_prepzos

D OMVS,L

```
HLSP
BPXO051I 22.14.19 DISPLAY OMVS 987
OMVS     000F ACTIVE                OMVS=(SP)
SYSTEM WIDE LIMITS:              LIMMSG=NONE
                    CURRENT  HIGHWATER     SYSTEM
                      USAGE       USAGE      LIMIT
MAXPROCSYS               401         548       1000
MAXUIDS                   75          99        100
MAXPTYS                    2           5        256
MAXMMAPAREA                0         291       4096
MAXSHAREPAGES           8996       30300     131072
IPCMSGNIDS                 7          23        500
IPCSEMNIDS                20          20        500
IPCSHMNIDS                20          21        500
IPCSHMSPAGES               0           3     262144
IPCMSGQBYTES             ---          12     262144
IPCMSGQMNUM              ---           3      10000
IPCSHMMPAGES             ---       10240      76800
SHRLIBRGNSIZE       67108864    67108864   67108864
SHRLIBMAXPAGES             0           0       4096
MAXUSERMOUNTSYS            0           0          0
MAXUSERMOUNTUSER           0           0          0
```

The z/OS Health Checker tool watches a number of system properties including OMVS limits

Overriding OMVS properties in a userid's OMVS segment

- If you can't set large enough values in BPXPRMxx for some reason,
  set them for the WAS started task userids and for the userid of any administrator running java or scripts.
  - `ALTUSER WSSRUQ1  OMVS(ASSIZEMAX(2147483647))`
- For example, the RACF commands generated when defining a new cell assign a larger FILEPROCMAX to the servant to ensure it will start if MAXFILEPROC in BPXPRMxx is not sufficient.
- Few customers use this because generally there is no problem setting larger system-wide values in BPXPRMxx.

Network limits (some key ones)

- BPMXPRMxx
- ```
  NETWORK DOMAINNAME(AF_INET)
   DOMAINNUMBER(2)
   MAXSOCKETS(64000)
   TYPE(CINET)
   INADDRANYPORT(10000)
   INADDRANYCOUNT(2000)
  ```
- Recommendation: `MAXSOCKET >= 12000`
- TCPIP buffersizes >= 64K
  - ```
    TCPCONFIG
    TCPSENDBFRSIZE 65535
    TCPRCVBUFRSIZE 65535
    ```
  - Recommendation: `TCPSENDBFRSIZE = TCPRCVBUFRSIZE >= 65535`
- FINWAIT time
  - Recommendation: `FINWAIT2TIME 60`
- Listen backlog in WAS transport handler
  - Recommendation: `listenBacklog=100`

rrun_chain_tcpcustom

KnowledgeCenter

tprf_tunetcpip

Collect the diagnostics

- Set a delay before a servant is restarted to allow other work to finish

  ```
  control_region_timeout_delay=10
  ```

- If you need to capture a dump at the time a thread times-out, you should remove the control_region_timeout_delay because this will delay taking the dump until some time after the thread timed-out.
- When diagnosing a problem, configure the server to take at least a Javacore automatically, but in normal running TRACEBACK is probably best

  ```
  control_region_timeout_dump_action=JAVACORE
  ```

  If a timeout terminates a servant, this step provides you with a Javacore dump and a traceback of the problem dispatch thread.

Minimizing the effects of timeouts

Defer servant restart: Don't restart last server

- Specify control_region_timeout_delay=5 (for example)
  - Allows a little more time for other requests to finish when a servant has been scheduled for restart because of a timeout
  - The request is cleaned up and a response sent to the client. Other requests are allowed to run but new work is not allowed in. Work with an affinity to the servant is also forced to queue or directed to another servant if one is running
  - Note that for transaction timeouts there is a 4 minute grace period following a transaction timeout before a servant restarts
- Ensure one servant always survives ?
  `control_region_timeout_save_last_servant=1`
  - This may not be a good idea if the reason the servant is scheduled for restart is a looping request or a general hang in a remote system. Usually better to allow all servants to restart but automatically close listeners if all servants are restarting.
    (`control_region_dreg_on_no_srs=1`)

KnowledgeCenter urun_rproperty_custproperties

Reject work when it can't run

- Reject requests that have been waiting so long that they will likely timeout …
  - `control_region_xxxx_timeout_output_percent=xx`
- Reject work automatically if no servants are available

  `control_region_dreg_on_no_srs=1`
  - Closes the listeners until MINSRS is reached
- If the reason the servants have all failed is a general problem with a remote system that must be fixed then you may prefer to have manual control over when the listeners are re-opened

  `control_region_confirm_recovery_on_no_srs to 1`
  - When the server is again ready to accept work after losing and regaining its servants, the server issues message BBOO0297A as a WTOR.
    Reply 'CONTINUE' to permit the server to begin accepting work.
  - The server issues message BBOO0299I whenever it "pauses" because it has no servant regions left. You can use this message as a key for your automation software to begin watching for message BBOO0297A.
- Reject work manually or using automation when there is a pervasive problem

  `F <server>,PAUSELISTENERS`

KnowledgeCenter trun_auto_noservant

Requeing work queued for a terminating servant

- If a servant is scheduled for restart following a timeout, all requests with an affinity to that servant will fail.
- With session replication enabled, requests may run OK on another servant because the session object can be recovered in another servant
- `control_region_http_requeue_enabled` [V8.0 Fix Pack 7 or later]
- This property specifies whether HTTP requests on the WLM Queue can be requeued to any available servant.
- When set to 1:
  - HTTP requests waiting on the WLM queue with an affinity to a servant that is marked for termination can be requeued to any available servant when the servant with affinity has terminated. (HTTP Session recovery will occur)
- When set to 0 (zero):
  - The server will fail any HTTP requests on the WLM queue with an affinity to a servant that is marked for termination immediately. Also, the server will fail any new HTTP request with affinity to the unhealthy servant.

KnowledgeCenter urun_rproperty_custproperties

Debugging problems that cause timeouts

Collecting diagnostics

- Look in the control region log for the `BBOO0327I` messages!
- The following properties avoid an immediate servant restart after timeout:

```
control_region_timeout_delay
protocol_http_timeout_output_recovery=SESSION
protocol_https_timeout_output_recovery=SESSION
```

  … which allows time to manually collect diagnostics but perhaps the problem thread will have been cleaned up by the time you take a dump.

- Set the following environment variable to specify the diagnostics taken when a timeout occurs:

```
control_region_timeout_dump_action=SVCDUMP|JAVACORE|HEAPDUMP|JAVATDUMP|TRACEBACK
control_region_timeout_dump_action_session= . . . |TRACEBACK
```

  - Ideally collect at least a JAVACORE when the timeout occurs because that will tell you directly which thread timed out and why. But if you suffer hundreds of timeouts you will not want hundreds of JAVACOREs so do this only when you need to.

- You can change the action by using these MODIFY (F) commands:

```
F <server>,TIMEOUTDUMPACTION=SVCDUMP|JAVACORE|HEAPDUMP|JAVATDUMP|TRACEBACK|NONE
F <server>,TIMEOUTDUMPACTIONSESSION= . . .
```

KnowledgeCenter rtrb_controllingtimeout

BBOO0327I

BBOO0327I in the control region provides information about the timed-out request.

```
BBOO0327I: {0} REQUEST TIMEOUT:
({1}):({2}):({3}):({4}):({5}):({6}):({7}):({8}):({9}):({10}):({11}):():():
({12})
```

- {0} indicates the type of work request that timed out: HTTP, IIOP, or MDB
- {1} is the ASID
- {2} is the request identifier. Use to correlate this message with other timeout-related messages
- {3} is for internal use.
- {4} is the TCB address of the task that dispatched the timed-out request. (Here `008D41C8`) That address is useful in looking at an SVC dump or Java core dump to find the right task
- {5} is the job or STC number
- {6}, {7}, and {8} are the timestamps for when the request arrived at the controller, was added to the queue, and was received into a servant for dispatch
- {9}, {10} the classname and method name associated with the timeout
- {11} indicates the host and port where the request originated (for IIOP and HTTP requests).
- {12} is blank for IIOP. For HTTP it has the URI and for MDB it has the source msg listener port

```
BBOO0327I HTTP REQUEST TIMEOUT: (003D): (FFFFFF82): (0001001D):
(008D41C8): (STC00059): (2006/06/16 14:00:39.733024): (2006/06/16
14:00:39.733083): (2006/06/16 14:00:39.733433): (RemoteWebContainer):
(httprequest): (ip addr=9.57.7.37 port=1063):
():():(/ivt/ivtserver?parm2=ivtservlet)
```

Visibility of exceptions handled in applications (1)

If applications handle system exceptions, how do you know there's a problem?

1. A request enters the server and is processed, but …

- Something goes wrong (in the server itself or in the application processing)
- The exception is marshaled and returned to the client.

- Set the following property to be notified of these situations:

  `protocol_bboc_log_`**`return`**`_exception=1`

  → WAS writes `BBOO0169W` to the error log when it returns an exception to a client.

`BBOO0169W: Returning response {0} to method {1} request from {2}`

{0} The exception type and minor code

{1} The method

{2} Hostname and port of the client or jobname, asid of a client address space

- For consistency define this environment variables at the cell level.

Visibility of exceptions handled in applications (2)

If applications handle system exceptions, how do you know there's a problem?

2. A client fails to wait for a request to complete and closes the connection

- Set the following property to be notified of these situations:

  `protocol_bboc_log_`**`response`**`_exception=1`

  → WAS writes `BBOO0168W` to the error log if this happens.

  – The message indicates whether a success or failure response is being sent

`BBOO0168W: Unable to return response {0} to method {1} request from {2}`

{0} The response type:
(NO_EXCEPTION, USER_EXCEPTION, LOCATION_FORWARD, or SYSTEM_EXCEPTION)
{1} The method
{2} Hostname and port of the client or jobname, asid of a client address space

- For consistency define this environment variables at the cell level.

CPU timer

CPU Timer

- An ODI can not interrupt a tight loop so WAS V7 has a CPU timer
- Set a CPU time used limit (millisecs) to prevent excess CPU utilization by a request

  `server_region_request_cputimeused_limit=2000`
- At the limit the CPU timer will:
  - Put the thread into a stalled state which means the servant will be restarted when it reaches the stalled thread limit
  - Gather the diagnostics specified in the following property:

    `server_region_cputimeused_dump_action`
  - Lower the priority of the dispatch thread. In a busy system this will stop it running.
  - Notify the controller so it cleans up the request, notifies the client and increments the stalled thread counter
- Limitations:
  - If the loop is in JNI, WAS might not be able to interrupt it.
  - Lowering the priority of a thread may not stop it running
  - When the stalled thread count is incremented because of a timeout it could be decremented if the request eventually completes but a looping request placed into a stalled state can never end. Therefore too many looping thread will cause a servant restart eventually.

KnowledgeCenter rtrb_controllingtimeout