Content Manager OnDemand

ODWEK troubleshooting – Java API

IBM Information Management software

@business on demand.

Updated August 10, 2009

This presentation covers ODWEK troubleshooting for Java API.

ODWEK Java API troubleshooting topics

- General information
- Connection cannot be established error
- Abend
- Hang
- Native memory leak

This module will cover: Java API troubleshooting topics, general Java API information, a common connection issue, Java API abend or hang situations, and causes for possible native memory leaks in ODWEK.

ODWEK Java APIs: general information

- ODWEK Java API issues are typically easier to troubleshoot then servlet and CGI, since diagnostics are better
  - Javacore current thread will indicate failing ODWEK API
  - Core file's native stack will show pertinent failing native method
  - Almost every issue should be recreatable using a skeleton Java stand-alone testcase
  - ODWEK trace=4 output is quite verbose, but more useful then servlet and CGI trace output

ODWEK Java API issues are typically easier to troubleshoot than servlet and CGI because the Javacore or the HotSpot error report indicates the failing ODWEK API. The servlet will just indicate the thread failed in main method.

Since ODWEK does most work in its native shared library, a stack trace from the core file is very helpful. This will assist you in determining if an abend is in any of the internal non-API helper functions.

Almost every issue should be re-creatable using a skeleton Java stand-alone test case.

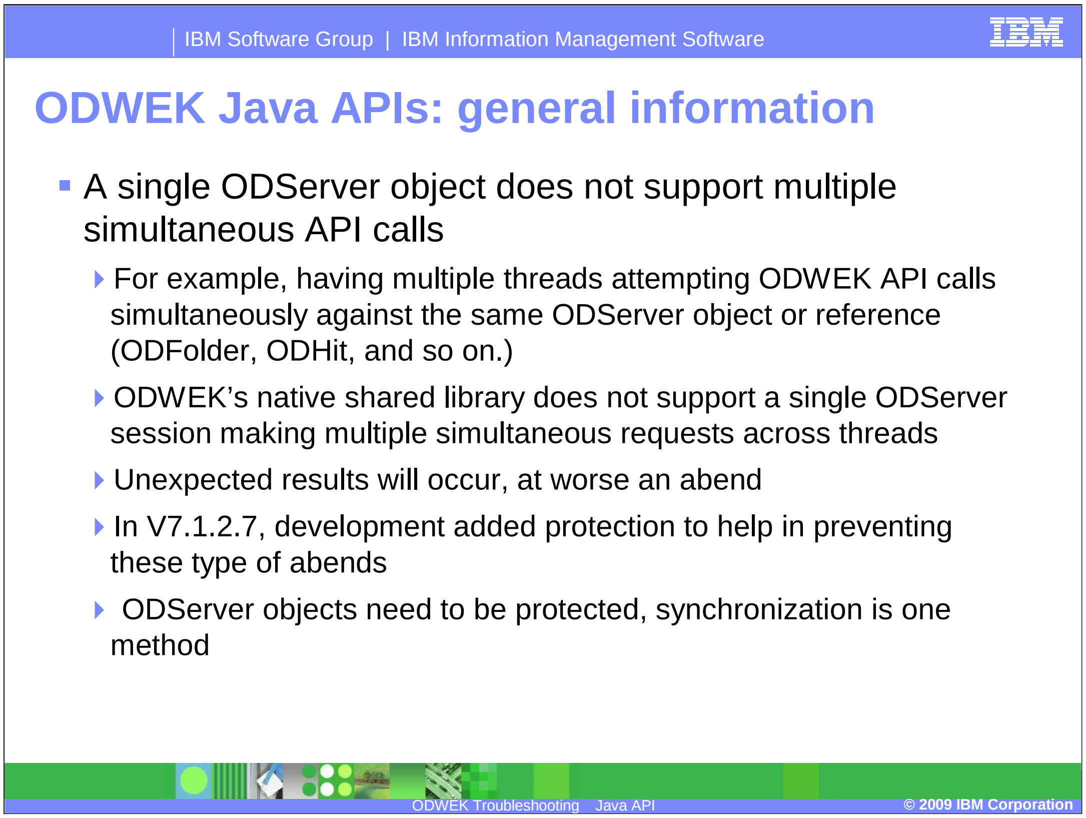
The ODWEK trace= 4 output contains more detailed information for the Java APIs compared to servlet and CGI.

ODWEK Java APIs: general information

- ODWEK Java APIs do not use CacheDir, only TempDir
  - Unlike servlet or CGI, the Java APIs do not have any built in document caching, you must implement this yourself
  - If a document or AFP resource is larger then 1 MB, it is retrieved to file in TempDir and then read into native heap
    - Default size set by the DocSize parameter

While you have to specify the CacheDir in the arswww.ini file, the Java APIs only use the TempDir for processing data. The TempDir is mainly used to retrieve documents larger than 1MB to file.

This is done to prevent fragmenting the native heap. The file is selected, processed, and then it is read back into memory as one large chunk. During document retrieval, ODWEK will typically retrieve document data in 100kb chunks from the OnDemand server. Once retrieved from the OnDemand server, it is processed; uncompressed or converted to another code page. This can be performed in memory or file before the document data is returned to the calling API.



There is a requirement in ODWEK Java APIs that a single ODServer object cannot support multiple simultaneous threads, attempting ODWEK API calls at a given time.

An example of this is through your application create ODServer object per each user session. The user does a search on a folder and they click the button repeatedly very quickly. If you allow your Java application to simultaneously call the search API against the same ODServer object, typically an abend or unexpected results will occur.

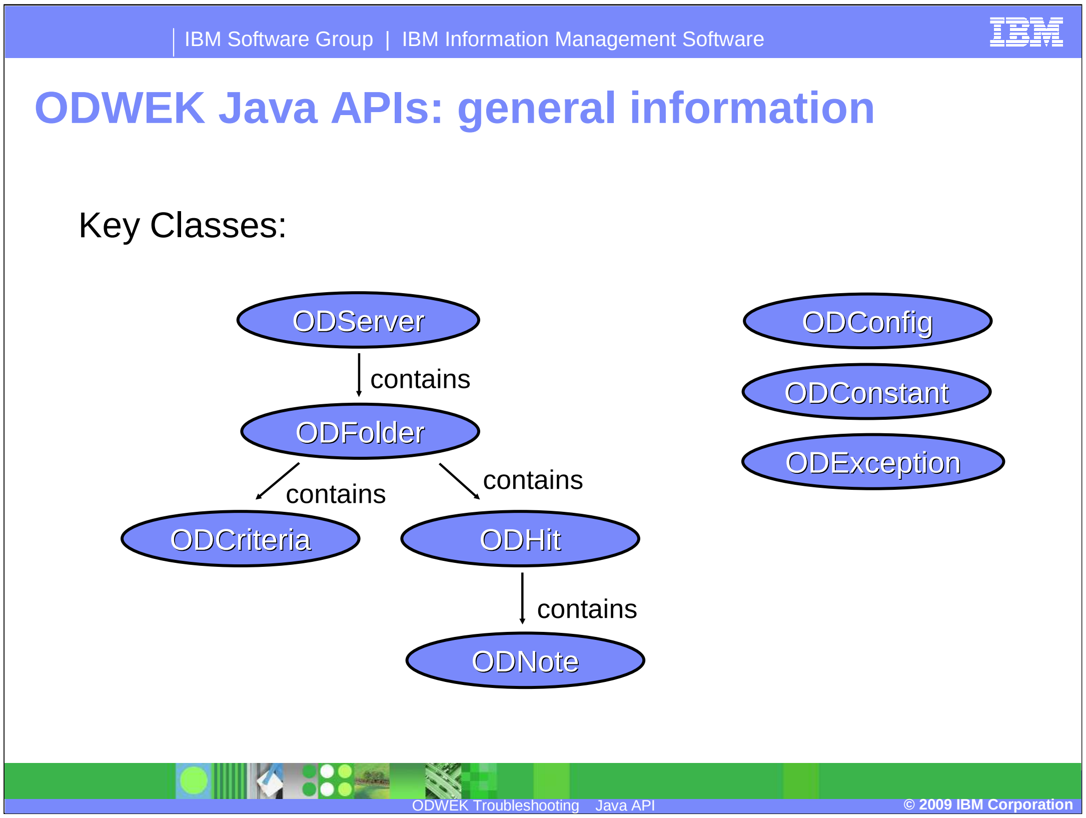
In ODWEK V7.1.2.7 development added protection to help prevent these types of abends, however you might still receive unexpected results if the ODServer object is not protected. One way to protect your application is to use Java synchronization methods.

ODWEK Java APIs: general information

- ODWEK generates and uses DocIDs to uniquely identify a document
  - Can be used to retrieve corresponding document using ODFolder.recreatHit(docID)
  - DocIDs are for IBM internal use only and are subject to change at any time
  - The same information can be obtained using the ODHitProperties and ODHit classes
  - Structure of a DocID:
    - version-agid-aid-fid-tablename-docname-objoff-objlen-compoff-complen-annot-comptype-rid-pri_nid-sec_nid-extra_info
  - Example DocID:
    - V7126-5014-5015-5013-CAA1-8FAAA-0-1593-0-91705-85-79-3-1-0-^SMITH CYCLERY CO 000-000-00089200.96000

DocIDs are only used by ODWEK and contain information such as the application group identifier, application identifier, folder, table, the offsets of the storage object (where the document is stored) and so on.

DocID information is for IBM internal use only. You can obtain the same information through many of the ODWEK APIs.



This is a basic hierarchy of the ODWEK Java API classes. There are roughly twice as many more classes than listed here, but these are the most important ones.

Everything gets started with the ODServer class. The ODServer class initializes and references the arswww.ini file or ODConfig object which specifies the basic ODWEK configuration parameters. Then with the ODServer class you can logon, list folders, open a folder, and so on. The ODFolder class from this point is issued to conduct searches, which returns a vector of ODHit objects. An ODHit object represents an individual document. You can retrieve a document by calling any retrieve API, such as ODHit retrieve. If you want to add an annotation, or retrieve annotations, that's done through the ODNote class.

On the right side of the diagram you can see that there is the ODConfig, ODConstant, ODException classes. ODConfig holds the ODWEK configuration parameters. ODConstant holds constant values that many APIs require. The ODException class holds any exceptions and messages that are thrown.

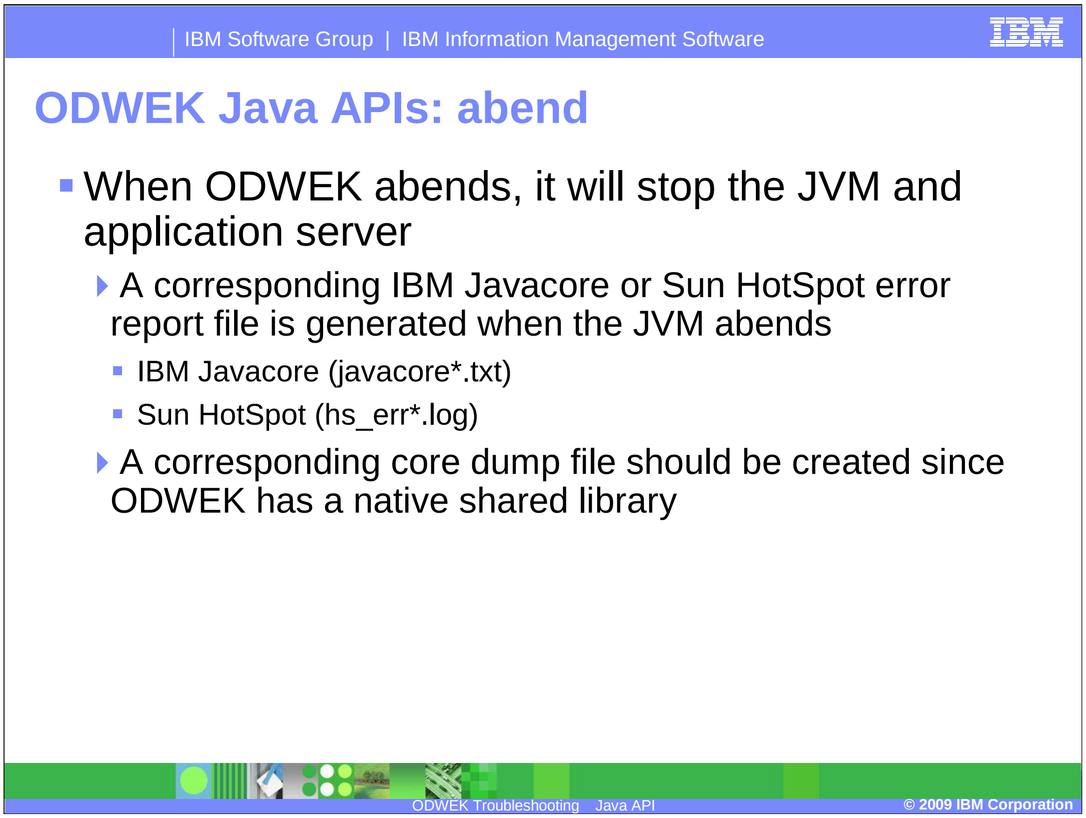
IBM Software Group | IBM Information Management Software

ODWEK Java APIs: connection cannot be established

- Using the ODWEK Java APIs, a “Connection cannot be established” exception keeps being thrown
  - ODWEK application has never been able to connect to the OnDemand server
- OnDemand server typically is running on a non-default port (non-zero or 1445 port)
- Specify ODServer.setPort(port_num) to set port ODWEK should use to connect to the OnDemand server

ODWEK Troubleshooting   Java API

Occasionally a new ODWEK application will experience the exception “Connection cannot be established”. If using ODWEK for the first time to connect to the OnDemand server, the problem is typically due to the OnDemand server running on a non-default port. The port number needs to be specified in ODWEK. This is accomplished by calling the ODServer.setPort API.



When ODWEK abends, the JVM application server will stop too. A corresponding Javacore (IBM Virtual Machine for Java Platforms) or Sun HotSpot error report (Sun JVM) is generated. Also because ODWEK is using a native shared library, a core dump file is created. If no core dump is created, check your OS configuration to allow core dumps to be created.

ODWEK Java APIs: abend

- It is within the design of ODWEK to stop if unable to allocate sufficient native heap
  - Rationale is that system is already in an unstable state, attempting to avoid an abend will make troubleshooting the unstable state more difficult

- Most ODWEK Java API abends are caused by:
  - Native heap exhaustion
  - API misuse
  - API defect

ODWEK Java APIs typically stop unexpectedly because native heap has been exhausted, API misuse, or defect.

ODWEK Java APIs: abend

- Native heap exhaustion
  - Javacore will typically show current thread abended in a document retrieve operation:
    - ODHit.retrieve
    - ODServer.viewerPassThru
    - ODHit.getDocument
    - ODHit.getResource
    - ODHit.retrieveSegment
  - Native call stack will show abend in getData, outputdriver, or retrieve named method
  - Review ODWEK trace=4 output to confirm size of last document being retrieved during the abend of an ODWEK API call
    - If large document that was last retrieved, recommend alternative write to file ODWEK APIs

Most ODWEK abends are due to native heap exhaustion.

If the Javacore's current thread is in one of the document retrieval operations - ODHit.retrieve, ODServer.viewerPassThru, ODHit.getDocument, ODHit.getResource, or ODHit.retrieveSegment - this typically indicates the abend was caused by native heap being exhausted. Native heap can be exhausted if a large document or documents was retrieved at a given time. Capture an ODWEK trace, look at the end of the trace and see the size of the document being retrieved. If the document size is very large, native heap is likely either not sized properly or the system is not sized sufficiently to support user volume or document size requirements. Moving to 64-bit, increasing native heap (by lowering maximum Java heap), or using another ODWEK retrieval API (write to file) are possible solutions if sizing is the problem.

If the native call stack shows an abend in getData, outputdriver, or a retrieve named method, this is indicative of native heap exhaustion during a document retrieval as well.

ODWEK Java APIs: abend

- If each abend shows different ODWEK APIs as the Current Thread or failing native method, there might still be a native heap exhaustion
- Check core file size, if very large this is an indication that native heap is exhausted
- Ensure the number of ODServer.logon and ODServer.terminate calls in the ODWEK trace=4 are close in number
- An accumulation of ODServer objects will show as a native heap leak

A good method to check if native heap might be exhausted and the 32-bit limitation on the amount of memory a process can allocate has been reached, is to check the core file size. The core is just a snapshot of memory that the process has allocated at the time of the application stop. For example, if the core size is 3 GB, the process had 3 GB of memory allocated at the time of the stop. Since this is a Java process, the 3 GB comprises Java heap and native heap.

Also, another common cause of native leaks is an accumulation of ODServer objects. What this means is that the ODServer objects are being initialized and logged on, but they are never being logged off and terminated. Each ODServer object has it own native references underneath the covers. By accumulating ODServer objects, a native heap leak will occur. It is necessary to check that a similar number of logons and terminate calls are being made in the ODWEK trace.

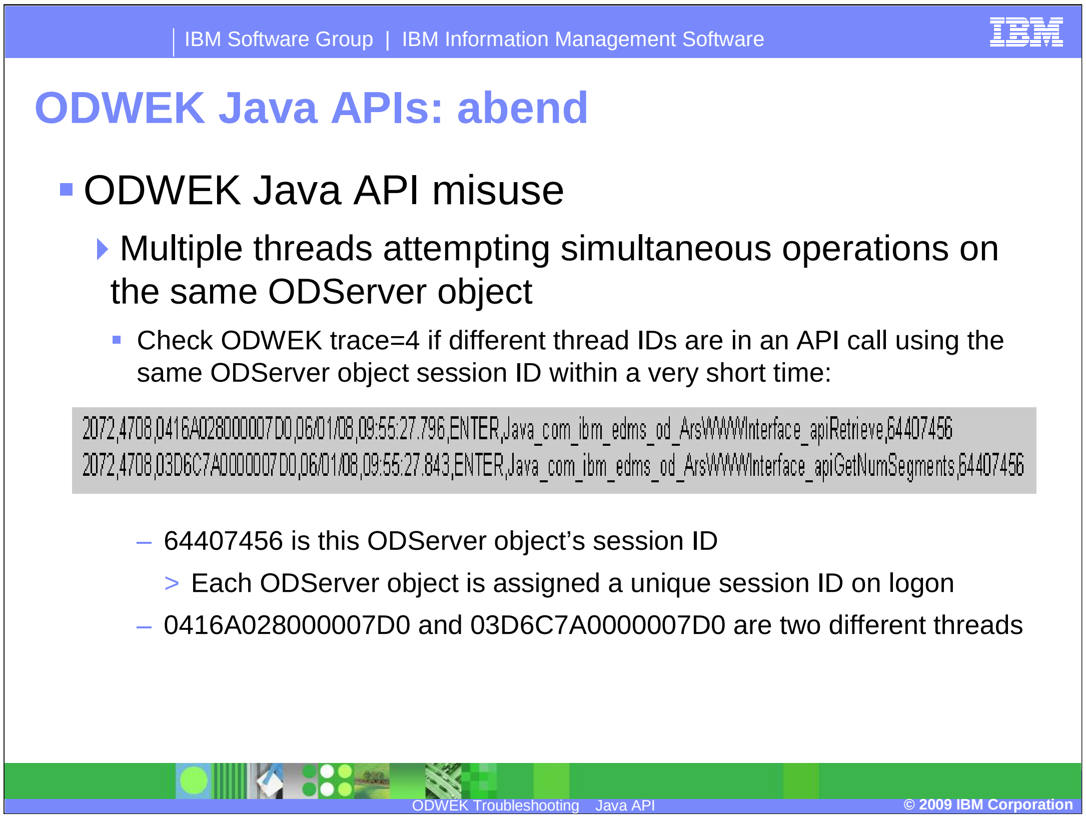
ODWEK Java APIs: abend

- Gather WebSphere Out of Memory MustGathers to monitor memory utilization
- If a native leak is suspected, see ODWEK Native Leak slide
- Review the Enabling Tracing, Logging and Requirements module for more information on ODWEK memory requirements

If native heap is exhausted, you can gather the “Out of Memory MustGather” items to monitor memory. Native heap will need to be monitored on the running Java process to determine if native heap increases steadily over time or if memory spikes right before ODWEK stops. If native memory constantly increases over a long period of time with constant load, investigation towards a memory leak should be performed.

The full set of abend files are required to provide IBM Software Support with the best opportunity to diagnose the problem. If only a few items are sent from you, it might be said that this is indicative of native heap exhaustion, however it is necessary to have a full set of abend files to confirm.

Finally, review the Enabling Tracing, Logging, and Requirements module for more information on ODWEK native memory requirements.



A common problem in new ODWEK Java API applications is protecting the ODServer object and references. Multiple threads accessing the same ODServer objects at the same time will cause unexpected results; at worst, an application stop. An ODWEK trace can be used to diagnose this type of issue.

An ODWEK trace will show when a method has entered and exited. Each ODServer object that has been initialized and logged in, is assigned a unique session ID. Also, the ODWEK trace will provide information such as the process ID, thread ID, and timestamp.

In this example, you see the two trace statements are from different threads.

In the first trace statement, the third value from the left has a thread ID starting with 0416A. The second trace statement has a thread ID starting with 03D6C.

A trace statement reads as such: parent process id, child process id, and thread id. So you can see these two different threads entering different functions, at about the same time, and both are using the same ODServer object, since the same session ID is being used.

The requirement by ODWEK is to protect the ODServer object from simultaneous threads. In this situation, you will want ODServer object operations to be synchronized. The first thread should enter, return….then the next thread should enter and return. This behavior can be re-created and displayed as an example by way of a test case.

ODWEK Java APIs: abend

- Incorrect order of API usage
  - For example, calling ODServer.logoff after ODServer.terminate can cause an ODWEK abend
    - A defect was found in ODWEK V7.1.2.9's error handling
- ODServer objects are not logged off and terminated
  - If a new ODServer object is created per session, will accumulate over time and leak memory

Another example of incorrect API usage that might cause an application stop is if the order of API's called is incorrect.

In this example, an application entered a code path where an ODServer logoff was called after an ODServer terminate. ODServer terminate performs clean up and frees the ODServer object, so calling logoff afterward actually caused ODWEK to stop unexpectedly. An exception should be thrown instead, so a product defect was opened.

Also, as mentioned earlier, native memory is leaked if an accumulation of ODServer objects occurs. This can happen if a new ODServer object is created per session, but never logged off and terminated.

ODWEK Java APIs: abend

- ODWEK API Defect
  - Attempt re-creating issue using skeleton Java stand-alone testcase
  - If issue occurs only through the use of multiple threads, model a multi-threaded Java stand-alone test case

If you suspect a defect in ODWEK Java API, the way to provide evidence of a possible problem is to re-create the issue using a skeleton Java stand-alone test case. Model the test case after your ODWEK trace output. A test case provides IBM support the most evidence of a possible defect.

IBM Software Group | IBM Information Management Software

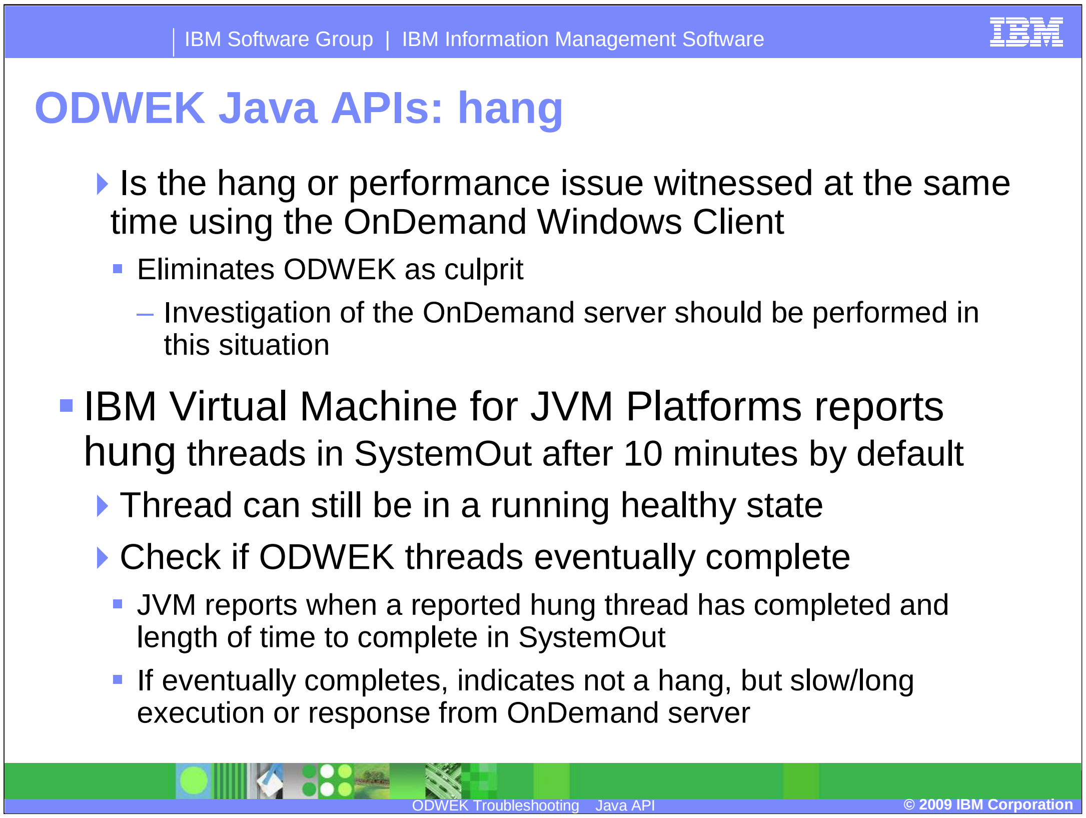
ODWEK Java APIs: hang

- ODWEK hang issues are typically caused by slow OnDemand server response
  - ODWEK is just a client to the OnDemand server
  - ODWEK will wait indefinitely until a response or error is received from the OnDemand server
    - No configurable method within ODWEK to timeout an operation
    - ODServer.cancel() can be used to cancel any search or retrieve operation

ODWEK Troubleshooting Java API

ODWEK is just a client to the OnDemand server, so if the OnDemand server is responding slowly, has performance problems or network issues, ODWEK is going to seem like it is slow or hung.

ODWEK will wait indefinitely for any response from the OnDemand server as there is nothing within ODWEK's configuration to time-out a request.



The best thing to do is ask that during these times of bad performance or hang periods, replicate the problem from the OnDemand server using the OnDemand Windows client. If the OnDemand Windows client hangs, this will eliminate ODWEK as the problem and point to the OnDemand server or to the network.

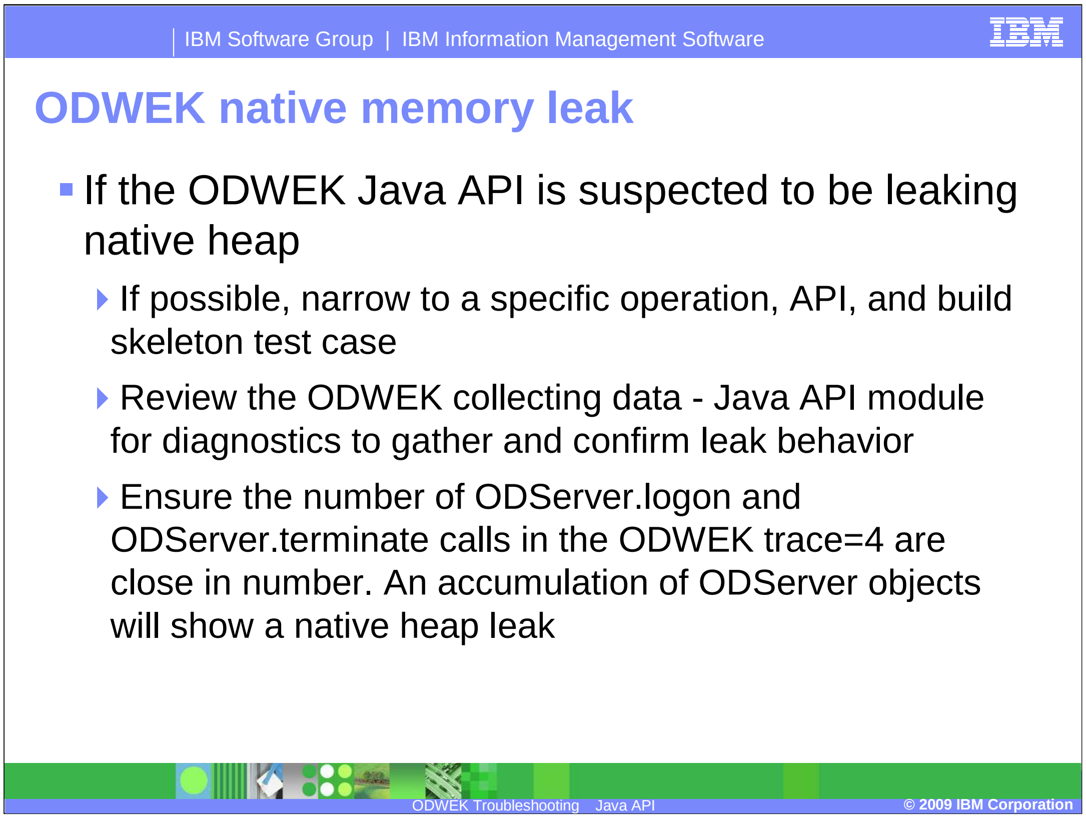
The JVM will report hung threads in the SystemOut.log, by default, after 10 minutes. If the thread eventually completes, the JVM also will indicate this in the SystemOut.log. This indicates that it is not a true hang; there is a performance or response time problem.

ODWEK Java APIs: hang

- Gather the Hang MustGather items to investigate further
  - Hang MustGather includes generating Javacores during hang situation
    - Javacores will report a deadlock condition if witnessed
  - Review the ODWEK collecting data - Java API module for further details

Provide the Hang MustGather items to investigate further. The Hang MustGather information comprises of at least one or more Javacores be taken at regular intervals during the hang condition. A thread dump of every thread is produced in the Javacore. Look to see if the thread activity has changed from Javacore to Javacore. If not, then take a look at the Javacore thread stack and see what it is stuck doing. Review the ODWEK collecting data - Java API module for further details on data to collect.

Regarding a hang for the ODWEK Java APIs, a hang defect has never been found, so it is rare that a hang is caused by a defect in ODWEK.



If the situation gets to the point where you believe there is a memory leak in ODWEK, the best thing to do is isolate it as much as possible. If it's Java APIs, build a test case, loop it, see if the memory is leaking slowly over time. If memory usage increases over time and under a constant load, a memory leak might exist.

If your ODWEK application is stopping unexpectedly, is there a pattern? Does it stop every X days or X hours?

If the answer is yes, there is either a memory leak or scheduled activity causing the abend.

ODWEK native memory leak

- Re-create issue with memory allocation trace instruments/tools
  - Method to trace native memory allocations and free calls
  - Utilities that can be used to trace memory allocation/free calls
    - Solaris – libumem
    - AIX® –malloc debug
    - Linux® – Valgrind
      - Will need to request special build of ODWEK to include symbol tables

Memory allocation instruments and tools can be used to trace native memory allocation and free calls. A listing of utilities can be found in this slide.

ODWEK native memory leak

- Windows – Debug Diagnostics Tool
  - Use Windows Performance Monitor to monitor total process virtual memory usage
  - Located at Control Panel->Administrative Tools->Performance
  - Create a new counter monitor to file or press the '+' button under System Monitor. Choose Process as your Performance objects. Select 'Virtual Bytes' on the left panel and the process in question in the right panel.
  - If Virtual Bytes continues to climb, indicates a leak

On Windows, the Debug Diagnostic Tool can be used to monitor memory allocations. Also, Windows performance monitor can be used to monitor total virtual memory allocated by a single process.

Useful diagnostic tools

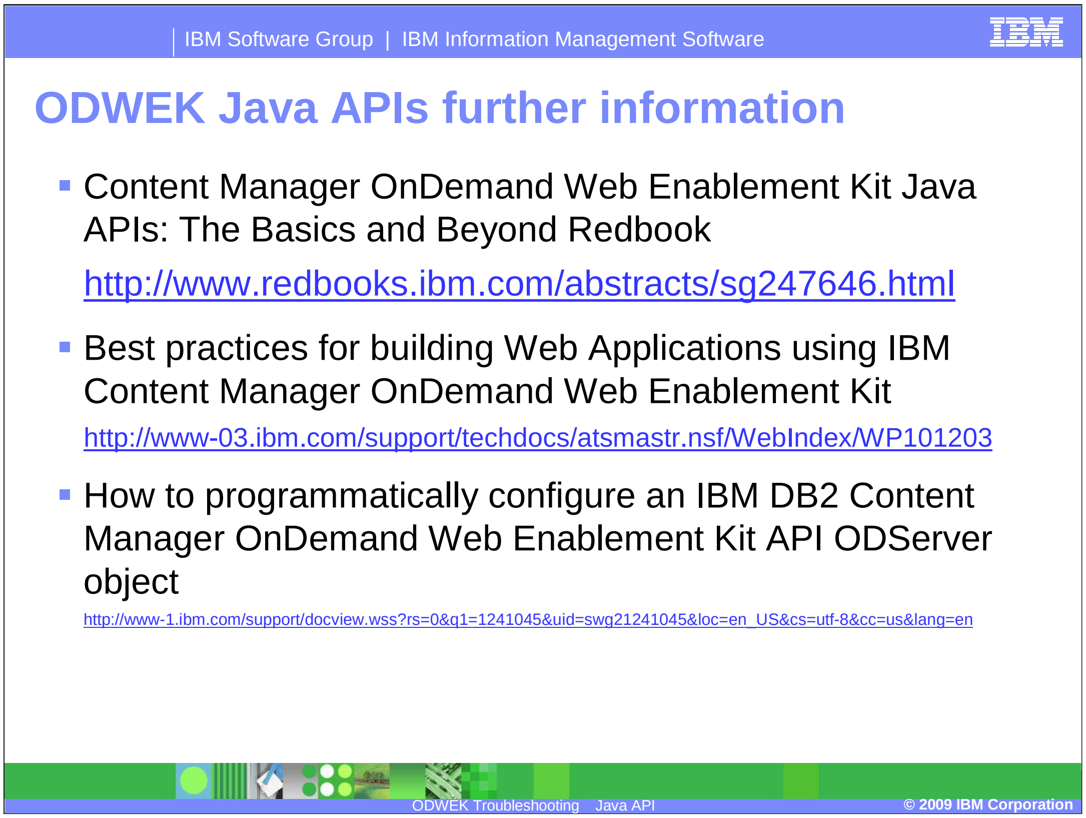
- IBM Thread and Monitor Dump Analyzer
  - Provides graphical and table representation of thread and monitor activity

    http://www.alphaworks.ibm.com/tech/jca
- IBM Heap Analyzer
  - Provides graphical representation of suspected Java heap leak objects

    http://www.alphaworks.ibm.com/tech/heapanalyzer

The IBM Thread and Monitor Dump Analyzer is a tool that reads Javacores and provides a graphical report of thread, monitor lock activity, and other useful information outputted in a Javacore.

The Heap Analyzer analyzes Java heap dumps if you believe there is a Java heap leak. You can use this utility to graphically review Java heap objects.



These links provide further information that is helpful to troubleshoot ODWEK Java API issues, best practices, configuration, and so on.

ODWEK Java APIs further information

- Using the ODWEK API ODConfig class with AFP Transforms
  http://www-1.ibm.com/support/docview.wss?rs=0&q1=1273746&uid=swg21273746&loc=en_US&cs=utf-8&cc=us&lang=en
- Why are you getting the Java exception 'UnsatisfiedLinkError: Native Library' when running two applications that use the ODWEK Java API?
  http://www-01.ibm.com/support/docview.wss?rs=0&q1=1244564&uid=swg21244564&loc=en_US&cs=utf-8&cc=us&lang=en
- New getResource and getDocument ODWEK APIs added in V7.1.2.6
  http://www-1.ibm.com/support/docview.wss?rs=0&q1=1237140&uid=swg21237140&loc=en_US&cs=utf-8&cc=us&lang=en

These are additional links that provide information that are helpful in troubleshooting ODWEK Java API issues.

All ODWEK further information

- ODWEK trace return codes
  - See the ODWEK Redbook chapter 13.1.5

  http://www.redbooks.ibm.com/abstracts/sg247646.html
- All arswww.ini parameter meanings

  http://publib.boulder.ibm.com/infocenter/cmod/v8r4m0//topic/com.ibm.ondemand.mp.doc/ars1y37158.htm
- ODWEK Implementation Guide

  http://publib.boulder.ibm.com/infocenter/cmod/v8r4m0/topic/com.ibm.ondemand.mp.doc/ars1y371.htm

This slide provides links to troubleshoot and analyze ODWEK trace return codes, configuration, and the ODWEK Implementation Guide.

All ODWEK further information

- Scrolling or repeatedly opening an AFP document in the AFP Web Viewer Plug-in causes a browser abend

  http://www-1.ibm.com/support/docview.wss?rs=0&q1=1295357&uid=swg21295357&loc=en_US&cs=utf-8&cc=us&lang=en

- OnDemand Web Enablement Kit (ODWEK) and OnDemand server performance slows during high usage

  http://www-1.ibm.com/support/docview.wss?rs&uid=swg21223757

This slide provides informative links to troubleshoot and analyze ODWEK known issues.

Miscellaneous links

- IBM Java Diagnostic Guide
  - V1.4.2
  - V1.5
  - V1.6
- Problem determination for Javacore files from WebSphere Application Server
  - Provides detailed explanation on how to read a Javacore

    http://www-1.ibm.com/support/docview.wss?fdoc=aimwas&rs=180&uid=swg21181068

The IBM Java Diagnostic Guide for the last three release levels, is fairly short and provides a lot of good information.

If you are not familiar with reading a Javacore file, the link provided gives you information on how to read it and what each section means.

Miscellaneous links

- MustGather: Read first for all WebSphere Application Server products

  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21145599

- Windows Java address space explanation

  http://www.ibm.com/developerworks/java/jdk/diagnosis/142.html

  - Search for the link “Windows Java address space”

The MustGather document assists you in collecting data for IBM Software Support before opening a problem report. The last link provides detailed information on how the process space for Java is divided between native heap, Java heap, libraries, and kernel.

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_Troubleshooting_JavaAPI.ppt

This module is also available in PDF format at: ../Troubleshooting_JavaAPI.pdf

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

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX Current DB2 WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Windows, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, JVM, and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.