# Web Services Integration Toolkit for OpenVMS
# Exceptions and Errors

**July 2012**

This document provides details about the exception classes in Web Services Integration toolkit. It also provides information about the probable causes for various exceptions as well as other errors.

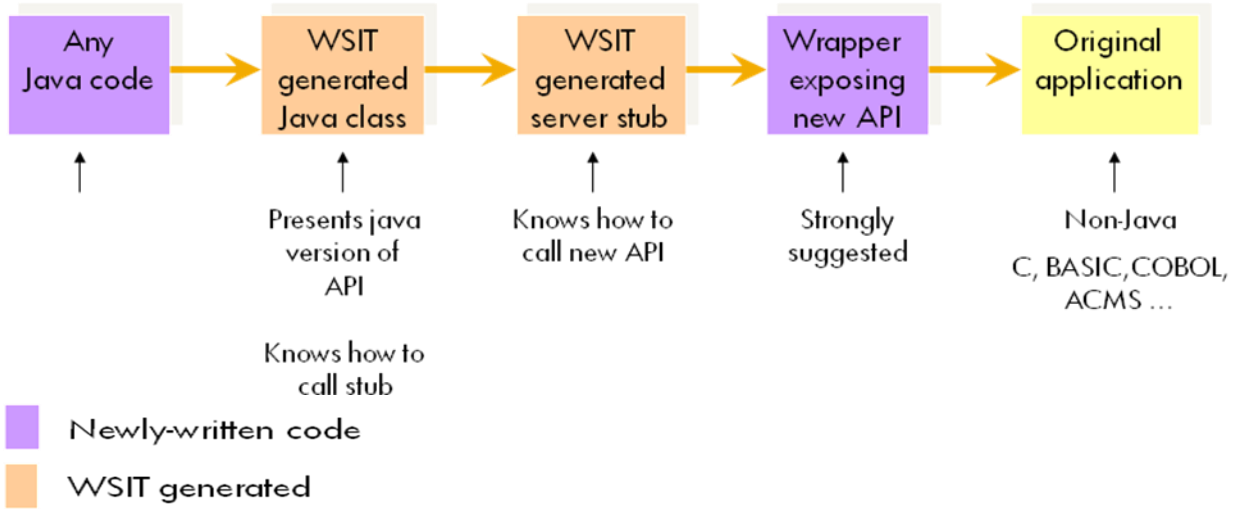
**Hewlett-Packard Company**
**Palo Alto, California**

## Contents

# 1 Introduction

This document provides details about the exception classes in Web Services Integration toolkit. It also provides information about the probable causes for various exceptions as well as other errors.

The general process of wrapping an application and turning it into a reusable service using WSIT consists of four steps.




1. Prepare the application to be wrapped.
2. Describe the application interface to the wrapping tool.
3. Run the code generator tool in order to create the newly packaged application.
4. Write a client for the newly created interface.

Deployment could result in runtime issues for various reasons. The exception classes and other errors described below help in understanding the runtime issues.
Tracing is another way in which applications can be analyzed for issues seen. This is done by passing the "-t" switch to the code generator. This produces two extra files as given below:

- <yourappname>-server-trace.h
- <yourappname>-server-trace.c

This file must be compiled and linked into the applications shareable image. Tracing can be turned on by defining the logical WSI$APPTRACING. The tracing output can be directed to a file by defining the WSI$LOGFILE logical. The log file name is based on the host and application name.

# 2 com.hp.wsi
# Class WsiException

java.lang.Object
  └ java.lang.Throwable
    └ java.lang.Exception
      └ java.io.IOException
        └ java.rmi.RemoteException
          └ **com.hp.wsi.WsiException**

**All Implemented Interfaces:**
java.io.Serializable

**Direct Known Subclasses:**
InvalidStateException, WsiConnectException, WsiServerException

---

public class **WsiException**
extends java.rmi.RemoteException

This class is the base WSIT runtime exception class that all other WSIT exception classes derive from. It contains the standard java exception properties & methods, as well as properties that give the client access to the original OpenVMS exception details, if applicable. It derives from the RemoteException class to work seamlessly within Web Services environments. (These environments, such as AXIS, handle the RemoteException without any extra details needed within the WSDL.)

For more information about “com.hp.wsi.WsiException: Inconsistent use of Protocols” exception, see Section 9.8.

**See Also:**
Serialized Form

---

## *2.1 Field Summary*

| | |
|---|---|
| static int | **WSI$K_ERROR**<br>This identifies the exception or error as having an error level. |
| static int | **WSI$K_FAC_IPC**<br>This identifies the exception or error as originating within the interprocess communication layer. |
| static int | **WSI$K_FAC_MGR**<br>This identifies the exception or error as originating within the wsi$manager process. |
| static int | **WSI$K_FAC_RTL**<br>This identifies the exception or error as originating within the RTL in the javabean process. |

| | |
|---|---|
| static int | **WSI$K_FAC_SVR**<br>This identifies the exception or error as originating within the server process. |
| static int | **WSI$K_INFO**<br>This identifies the exception or error as having an informational(success) level. |
| static int | **WSI$K_SEVERE**<br>This identifies the exception or error as having a severe error level. |
| static int | **WSI$K_SUCCESS**<br>This identifies the exception or error as having a success level. |
| static int | **WSI$K_WARNING**<br>This identifies the exception or error as having a warning level. |

## 2.2 Constructor Summary

| |
|---|
| **WsiException**()<br>Constructs this exception with a default value. |
| **WsiException**(java.lang.Exception ex)<br>Constructs this exception with an embedded java exception. |
| **WsiException**(java.lang.String errMessage)<br>Constructs this exception with a specified error message. |
| **WsiException**(java.lang.String errMessage, java.lang.Exception ex)<br>Constructs this exception with a specified error message and an embedded java exception. |
| **WsiException**(java.lang.String errMessage, int inExcCode, int inVmsCode)<br>Constructs this exception using a given error message, internal exception code, and the original OpenVMS error code. |
| **WsiException**(java.lang.String errMessage, int inExcCode, int inVmsCode, int inSeverity, int inFacility)<br>Constructs this exception using a given error message, internal exception code, the original OpenVMS error code, the severity, and the facility responsible for the exception. |
| **WsiException**(WsiException wsiEx)<br>Constructs this exception with the values from another specified exception object. |

## 2.3 Method Summary

| | |
|---|---|
| int | **getFacility**()<br>Return the Facility Code if available |
| int | **getSeverity**()<br>Return the Severity level if available |
| int | **getStatus**()<br>Return the Internal exception code if one exists |
| int | **getVmsStatus**()<br>Return the OpenVMS exception status if one exists |
| boolean | **isConnectionValid**()<br>Return a flag indicating if the connection to the server process is still valid. |

| Methods inherited from class java.rmi.RemoteException |
| --- |
| getCause, getMessage |

| Methods inherited from class java.lang.Throwable |
| --- |
| fillInStackTrace, getLocalizedMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

## *2.4 Field Detail*

**WSI$K_WARNING**
public static final int **WSI$K_WARNING**

This identifies the exception or error as having a warning level.

**See Also:**
Constant Field Values

---

**WSI$K_SUCCESS**
public static final int **WSI$K_SUCCESS**

This identifies the exception or error as having a success level.

**See Also:**
Constant Field Values

---

**WSI$K_ERROR**
public static final int **WSI$K_ERROR**
This identifies the exception or error as having an error level
**See Also:**
Constant Field Values

---

**WSI$K_INFO**
public static final int **WSI$K_INFO**

This identifies the exception or error as having an informational(success) level.

**See Also:**
Constant Field Values

---

**WSI$K_SEVERE**
public static final int **WSI$K_SEVERE**

This identifies the exception or error as having a severe error level.

**See Also:**
Constant Field Values

---

**WSI$K_FAC_IPC**
public static final int **WSI$K_FAC_IPC**

This identifies the exception or error as originating within the interprocess communication layer.

**See Also:**
Constant Field Values

---

**WSI$K_FAC_RTL**
public static final int **WSI$K_FAC_RTL**

This identifies the exception or error as originating within the RTL in the javabean process.

**See Also:**
Constant Field Values

---

**WSI$K_FAC_MGR**
public static final int **WSI$K_FAC_MGR**

This identifies the exception or error as originating within the wsi$manager process.

**See Also:**
Constant Field Values

---

**WSI$K_FAC_SVR**
public static final int **WSI$K_FAC_SVR**
This identifies the exception or error as originating within the server process.
**See Also:**
Constant Field Values

## *2.5 Constructor Detail*

**WsiException**

public **WsiException**()

Constructs this exception with a default value.

---

**WsiException**

public **WsiException**(java.lang.String errMessage)

Constructs this exception with a specified error message.

**Parameters:**
errMessage - error message to assign to this exception object.

---

**WsiException**

public **WsiException**(java.lang.String errMessage,
java.lang.Exception ex)

Constructs this exception with a specified error message and an embedded java exception.

**Parameters:**
errMessage - error message to assign to this exception object.
ex - exception object to embed within this object.

---

**WsiException**

public **WsiException**(java.lang.Exception ex)

Constructs this exception with an embedded java exception. The error message for this object will be pulled from the exception being embedded.

**Parameters:**
ex - exception object to embed within this object.

---

**WsiException**

public **WsiException**(java.lang.String errMessage,
int inExcCode,
int inVmsCode)

Constructs this exception using a given error message, internal exception code, and the original OpenVMS error code.

**Parameters:**
errMessage - error message to be assigned to this exception object.
inExcCode - internal exception code identifying facility, severity, and code
inVmsCode - original OpenVMS error code that triggered the exception process

---

**WsiException**

public **WsiException**(java.lang.String errMessage,
int inExcCode,
int inVmsCode,
int inSeverity,
int inFacility)

Constructs this exception using a given error message, internal exception code, the original OpenVMS error code, the severity, and the facility responsible for the exception.

**Parameters:**
errMessage - error message to be assigned to this exception object.
inExcCode - internal exception code identifying facility, severity, and code
inVmsCode - original OpenVMS error code that triggered the exception process
inSeverity - code identifying the severity of the exception. (See WSI$K_* definitions.)
inFacility - code identifying the facility that originated the exception. (See WSI$K_FAC_* definitions.)

---

**WsiException**

public **WsiException**(WsiException wsiEx)

Constructs this exception with the values from another specified exception object. (A copy constructor.) It does not embed this other object.

**Parameters:**
wsiEx - exception object to copy values from.

## 2.6 Method Detail

**getStatus**

public int **getStatus**()

Return the Internal exception code if one exists.

**Returns:**
internal exception code

---

**getVmsStatus**

public int **getVmsStatus**()

Return the OpenVMS exception status if one exists.

**Returns:**
original OpenVMS error status

---

**getSeverity**

public int **getSeverity**()
Return the Severity level if available
**Returns:**
severity level of this exception. (Refer to the WSI$K_* status level definitions.)

---

**getFacility**

public int **getFacility**()
Return the Facility Code if available
**Returns:**
facility that originated this exception. (Refer to the WSI$K_FAC_* definitions.)

---

**isConnectionValid**

public boolean **isConnectionValid**()
Return a flag indicating if the connection to the server process is still valid. This can be used by a client to determine if it needs to close the connection and reestablish a fresh connection to the server.

# 3 Constant Field Values

## com.hp.*

| com.hp.wsi.ServerConfig | | |
|---|---|---|
| public static final int | LIFETIME_SESSION | 2 |
| public static final int | NO_SESSION | 0 |
| public static final int | TX_LIFETIME | 3 |
| public static final int | TX_SESSION | 1 |

| com.hp.wsi.WsiException | | |
|---|---|---|
| public static final int | WSI$K_ERROR | 2 |
| public static final int | WSI$K_FAC_IPC | 1 |
| public static final int | WSI$K_FAC_MGR | 3 |
| public static final int | WSI$K_FAC_RTL | 2 |
| public static final int | WSI$K_FAC_SVR | 4 |
| public static final int | WSI$K_INFO | 3 |
| public static final int | WSI$K_SEVERE | 4 |
| public static final int | WSI$K_SUCCESS | 1 |
| public static final int | WSI$K_WARNING | 0 |

# 4 com.hp.wsi
# Class NoServerException

java.lang.Object
  └ java.lang.Throwable
    └ java.lang.Exception
      └ java.io.IOException
        └ java.rmi.RemoteException
          └ com.hp.wsi.WsiException
            └ com.hp.wsi.WsiConnectException
              └ **com.hp.wsi.NoServerException**

**All Implemented Interfaces:**
java.io.Serializable

---

public class **NoServerException**
extends WsiConnectException

This class is thrown when a connection to the server process and/or image cannot be established. Possible causes of this exception are: the Server Wrapper was never built and deployed on the machine; the account does not have read/execute access to the server image.

**See Also:**
Serialized Form

---

## 4.1 Field Summary

| **Fields inherited from class com.hp.wsi.WsiException** |
|---|
| WSI$K_ERROR, WSI$K_FAC_IPC, WSI$K_FAC_MGR, WSI$K_FAC_RTL, WSI$K_FAC_SVR, WSI$K_INFO, WSI$K_SEVERE, WSI$K_SUCCESS, WSI$K_WARNING |

## 4.2 Constructor Summary

| |
|---|
| **NoServerException**()<br>Creates an exception with default values. |
| **NoServerException**(java.lang.String err)<br>Creates an exception with a specified error message. |

## 4.3 Method Summary

| **Methods inherited from class com.hp.wsi.WsiException** |
|---|
| getFacility, getSeverity, getStatus, getVmsStatus, isConnectionValid |

| Methods inherited from class java.rmi.RemoteException |
| --- |
| getCause, getMessage |

| Methods inherited from class java.lang.Throwable |
| --- |
| fillInStackTrace, getLocalizedMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

## *4.4 Constructor Detail*

**NoServerException**

public **NoServerException**()

Creates an exception with default values.

---

**NoServerException**

public **NoServerException**(java.lang.String err)

Creates an exception with a specified error message.

**Parameters:**
err - error message to use for this exception.

---

# 5 com.hp.wsi
# Class WsiConnectException

java.lang.Object
  └ java.lang.Throwable
    └ java.lang.Exception
      └ java.io.IOException
        └ java.rmi.RemoteException
          └ com.hp.wsi.WsiException
            └ **com.hp.wsi.WsiConnectException**

**All Implemented Interfaces:**
java.io.Serializable

**Direct Known Subclasses:**
NoServerException

---

public class **WsiConnectException**
extends WsiException

This class represents an exception that is thrown by the communication channel itself. This exception can be caught and handled from within the client. However this class indicates that the connection is closed, so the client must establish a new connection to the server.

This exception also gets thrown if memory allocation for sending and receiving messages fail. The message seen in that case is “ERROR: Cannot allocate Locate Request message”. The reasons why memory allocation for send/receive can fail are:

- Null client context is passed
- Client context with invalid block identification tags is passed
- Client context state is not the start state
- Protocol specific client buffer allocation failed
- Protocol definition is missing in the client context. In this case the error message “INTERNAL ERROR: Missing protocol definition” would also be seen.
- The IPC connection timed out. In this case the error message “WSIIPC Timeout exceeded” would be seen.

**See Also:**
Serialized Form

---

## *5.1 Field Summary*

| **Fields inherited from class com.hp.wsi.WsiException** |
| --- |
| WSI$K_ERROR, WSI$K_FAC_IPC, WSI$K_FAC_MGR, WSI$K_FAC_RTL, WSI$K_FAC_SVR, WSI$K_INFO, WSI$K_SEVERE, WSI$K_SUCCESS, WSI$K_WARNING |

## 5.2 Constructor Summary

| |
|---|
| **WsiConnectException**()<br>Constructs this exception with default values |
| **WsiConnectException**(java.lang.String errMessage)<br>Constructs this class with a specified error message |
| **WsiConnectException**(java.lang.String errMessage, int inExcCode, int inVmsCode)<br>Constructs this exception with specified values for error message, internal exception code, and original OpenVMS error status. |
| **WsiConnectException**(java.lang.String errMessage, int inExcCode, int inVmsCode, int inSeverity, int inFacility)<br>Constructs this exception using a given error message, internal exception code, the original OpenVMS error code, the severity, and the facility responsible for the exception. |
| **WsiConnectException**(WsiConnectException wsiEx)<br>Constructs this exception with the values from another specified exception object. |

## 5.3 Method Summary

| **Methods inherited from class com.hp.wsi.WsiException** |
|---|
| getFacility, getSeverity, getStatus, getVmsStatus, isConnectionValid |

| **Methods inherited from class java.rmi.RemoteException** |
|---|
| getCause, getMessage |

| **Methods inherited from class java.lang.Throwable** |
|---|
| fillInStackTrace, getLocalizedMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

## 5.4 Constructor Detail

**WsiConnectException**
public **WsiConnectException**()

Constructs this exception with default values.

---

**WsiConnectException**
public **WsiConnectException**(java.lang.String errMessage)

Constructs this class with a specified error message.

**Parameters:**
errMessage - error message associated with this exception.

---

**WsiConnectException**
public **WsiConnectException**(java.lang.String errMessage,
int inExcCode,
int inVmsCode)

Constructs this exception with specified values for error message, internal exception code, and original OpenVMS error status.

**Parameters:**
errMessage - error message to be assigned to this exception object.
inExcCode - internal exception code identifying facility, severity, and code
inVmsCode - original OpenVMS error code that triggered the exception process

---

**WsiConnectException**
public **WsiConnectException**(java.lang.String errMessage,
int inExcCode,
int inVmsCode,
int inSeverity,
int inFacility)

Constructs this exception using a given error message, internal exception code, the original OpenVMS error code, the severity, and the facility responsible for the exception.

**Parameters:**
errMessage - error message to be assigned to this exception object.
inExcCode - internal exception code identifying facility, severity, and code
inVmsCode - original OpenVMS error code that triggered the exception process
inSeverity - code identifying the severity of the exception. (See WSI$K_* definitions.)
inFacility - code identifying the facility that originated the exception. (See WSI$K_FAC_* definitions.)

---

**WsiConnectException**
public **WsiConnectException**(WsiConnectException wsiEx)

Constructs this exception with the values from another specified exception object. (A copy constructor.) It does not embed this other exception.

**Parameters:**
wsiEx - exception object to copy values from.

---

# 6 com.hp.wsi
# Class WsiServerException

java.lang.Object
  └ java.lang.Throwable
    └ java.lang.Exception
      └ java.io.IOException
        └ java.rmi.RemoteException
          └ com.hp.wsi.WsiException
            └ **com.hp.wsi.WsiServerException**

**All Implemented Interfaces:**
java.io.Serializable

---

public class **WsiServerException**
extends WsiException

This exception is thrown when the exception originates within the server image or process. It usually means that an exception was thrown within the user's application being wrapped.

This exception also gets thrown if memory allocation for sending and receiving messages fail. The message seen in that case is “ERROR: Cannot allocate Locate Request message”. The reasons why memory allocation for send/receive can fail are:

- Null client context is passed
- Client context with invalid block identification tags is passed
- Client context state is not the start state
- Protocol specific client buffer allocation failed
- Protocol definition is missing in the client context. In this case the error message “INTERNAL ERROR: Missing protocol definition” would also be seen.
- The IPC connection timed out. In this case the error message “WSIIPC Timeout exceeded” would be seen.

**See Also:**
Serialized Form

---

## *6.1 Field Summary*

| **Fields inherited from class com.hp.wsi.WsiException** |
| --- |
| WSI$K_ERROR, WSI$K_FAC_IPC, WSI$K_FAC_MGR, WSI$K_FAC_RTL, WSI$K_FAC_SVR, WSI$K_INFO, WSI$K_SEVERE, WSI$K_SUCCESS, WSI$K_WARNING |

## 6.2 Constructor Summary

| |
|---|
| **WsiServerException**()<br>Constructs this exception with default values |
| **WsiServerException**(java.lang.String errMessage)<br>Constructs this exception with a specified error message |
| **WsiServerException**(java.lang.String errMessage, int inExcCode, int inVmsCode)<br>Constructs this exception using a given error message, internal exception code, and the original OpenVMS error code. |
| **WsiServerException**(java.lang.String errMessage, int inExcCode, int inVmsCode, int inSeverity, int inFacility)<br>Constructs this exception using a given error message, internal exception code, the original OpenVMS error code, the severity, and the facility responsible for the exception. |
| **WsiServerException**(WsiServerException wsiEx)<br>Constructs this exception with the values from another specified exception object. |

## 6.3 Method Summary

| **Methods inherited from class com.hp.wsi.WsiException** |
|---|
| getFacility, getSeverity, getStatus, getVmsStatus, isConnectionValid |

| **Methods inherited from class java.rmi.RemoteException** |
|---|
| getCause, getMessage |

| **Methods inherited from class java.lang.Throwable** |
|---|
| fillInStackTrace, getLocalizedMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

## 6.4 Constructor Detail

**WsiServerException**
public **WsiServerException**()

Constructs this exception with default values.

---

**WsiServerException**
public **WsiServerException**(java.lang.String errMessage)

Constructs this exception with a specified error message.

**Parameters:**
errMessage - error message to assign to this exception object.

---

**WsiServerException**
public **WsiServerException**(java.lang.String errMessage,
int inExcCode,
int inVmsCode)

Constructs this exception using a given error message, internal exception code, and the original OpenVMS error code.

**Parameters:**
errMessage - error message to be assigned to this exception object.
inExcCode - internal exception code identifying facility, severity, and code
inVmsCode - original OpenVMS error code that triggered the exception process

---

**WsiServerException**
public **WsiServerException**(java.lang.String errMessage,
int inExcCode,
int inVmsCode,
int inSeverity,
int inFacility)

Constructs this exception using a given error message, internal exception code, the original OpenVMS error code, the severity, and the facility responsible for the exception.

**Parameters:**
errMessage - error message to be assigned to this exception object.
inExcCode - internal exception code identifying facility, severity, and code
inVmsCode - original OpenVMS error code that triggered the exception process
inSeverity - code identifying the severity of the exception. (See WSI$K_* definitions.)
inFacility - code identifying the facility that originated the exception. (See WSI$K_FAC_* definitions.)

---

**WsiServerException**
public **WsiServerException**(WsiServerException wsiEx)

Constructs this exception with the values from another specified exception object. (A copy constructor.) It does not embed this other object.

**Parameters:**
wsiEx - exception object to copy values from.

---

# 7 com.hp.wsi
# Class InvalidStateException

java.lang.Object
  └ java.lang.Throwable
    └ java.lang.Exception
      └ java.io.IOException
        └ java.rmi.RemoteException
          └ com.hp.wsi.WsiException
            └ **com.hp.wsi.InvalidStateException**

**All Implemented Interfaces:**
java.io.Serializable

---

public class **InvalidStateException**
extends WsiException

This exception indicates that the JNI context handle was found to be in an invalid state. This probably indicates an internal logic error.

**See Also:**
Serialized Form

---

## 7.1 Field Summary

| **Fields inherited from class com.hp.wsi.WsiException** |
|---|
| WSI$K_ERROR, WSI$K_FAC_IPC, WSI$K_FAC_MGR, WSI$K_FAC_RTL, WSI$K_FAC_SVR, WSI$K_INFO, WSI$K_SEVERE, WSI$K_SUCCESS, WSI$K_WARNING |

## 7.2 Constructor Summary

| |
|---|
| InvalidStateException()<br>Creates new InvalidStateException without detail message. |
| InvalidStateException(java.lang.String msg)<br>Constructs an InvalidStateException with the specified detail message. |

## 7.3 Method Summary

| **Methods inherited from class com.hp.wsi.WsiException** |
|---|
| getFacility, getSeverity, getStatus, getVmsStatus, isConnectionValid |

| Methods inherited from class java.rmi.RemoteException |
| --- |
| getCause, getMessage |

| Methods inherited from class java.lang.Throwable |
| --- |
| fillInStackTrace, getLocalizedMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString |

| Methods inherited from class java.lang.Object |
| --- |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

## *7.4 Constructor Detail*

**InvalidStateException**
public InvalidStateException()

Creates new InvalidStateException without detail message.

---

**InvalidStateException**
public InvalidStateException(java.lang.String msg)

Constructs an InvalidStateException with the specified detail message.

**Parameters:**
msg - the detail message.

---

# 8 com.hp.wsi
# Class ServerConfig

java.lang.Object
└ **com.hp.wsi.ServerConfig**

**All Implemented Interfaces:**
java.io.Serializable

**Direct Known Subclasses:**
WsiIpcContext

---

public class **ServerConfig**
extends java.lang.Object
implements java.io.Serializable

An instance of this class contains all the configuration information needed to start up and initialize the native server(s) for an EIS. All of these attributes are settable by the application deployer, and not hardwired into any code module.

**See Also:**
Serialized Form

---

## 8.1 Field Summary

| | |
|---|---|
| static int | **LIFETIME_SESSION** |
| static int | **NO_SESSION** |
| static int | **TX_LIFETIME** |
| static int | **TX_SESSION** |

## 8.2 Constructor Summary

| |
|---|
| **ServerConfig**()<br>Creates new ServerConfig |

## 8.3 Method Summary

| | |
|---|---|
| void | **addPropertyChangeListener**(java.beans.PropertyChangeListener l)<br>Add a PropertyChangeListener to the listener list. |
| java.lang.String | **getAppName**()<br>Getter for property appName. |
| java.lang.String | **getAppUuid**()<br>Getter for property appUuid. |
| java.lang.String | **getBinding**()<br>Retrieve the Binding property. |
| java.lang.String | **getDomain**()<br>Getter for property Domain. |

| | |
|---|---|
| int | **getLeaseTimeout**()<br>Retrieve the Lease Timeout property. |
| int | **getMajorVersion**()<br>Retrieve the Major Version property. |
| int | **getMinorVersion**()<br>Retrieve the Minor Version property. |
| java.lang.String | **getScsnodeName**() |
| int | **getSessionType**() |
| java.lang.String | **getTcpipName**() |
| java.lang.String | **getTxBinding**()<br>Retrieve the TX Binding property. |
| java.lang.String | **getTxScsnodeName**() |
| java.lang.String | **getTxTcpipName**() |
| boolean | **isAppMultithreaded**() |
| boolean | **isLocal**()<br>Indicate if the NativeShell instance is to be local to this JVM or at a remote RMI server. |
| boolean | **isTransport**()<br>Return whether the server sharable is to be loaded into the current process space or not. |
| void | **removePropertyChangeListener**(java.beans.PropertyChangeListener l)<br>Removes a PropertyChangeListener from the listener list. |
| void | **setAppName**(java.lang.String value)<br>Setter for property appName. |
| void | **setAppUuid**(java.lang.String value)<br>Setter for property appUuid. |
| void | **setBinding**() |
| void | **setBinding**(java.lang.String value)<br>Update the Binding property |
| void | **setBinding**(java.lang.String IPHost, java.lang.String ICCHost) |
| void | **setDomain**(java.lang.String value)<br>Setter for property Domain. |
| void | **setLeaseTimeout**(int value)<br>Update the Lease Timeout property |
| void | **setMajorVersion**(int value)<br>Update the Major Version property |
| void | **setMinorVersion**(int value)<br>Update the Minor Version property |
| void | **setScsnodeName**(java.lang.String mICCHostName) |
| void | **setSessionType**(int value) |

| | |
|---|---|
| void | **setSessionType**(java.lang.String value) |
| void | **setTcpipName**(java.lang.String mIPHostName) |
| void | **setTxBinding**(java.lang.String value)<br>Update the Binding property |
| void | **setTxBinding**(java.lang.String IPHost, java.lang.String ICCHost) |
| void | **setTxScsnodeName**(java.lang.String mTxICCHostName) |
| void | **setTxTcpipName**(java.lang.String mTxIPHostName) |
| java.lang.String | **toString**()<br>Custom string formatting method. |

| **Methods inherited from class java.lang.Object** |
|---|
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

## *8.4 Field Detail*

**NO_SESSION**
public static final int **NO_SESSION**

NO_SESSION maintains no open session between JavaBean & Server. This is a session control value stating that no connection should be maintained to the server between calls.

**See Also:**
Constant Field Values

---

**TX_SESSION**
public static final int **TX_SESSION**

TX_SESSION maintains session for duration of ACMS and/or OpenVMS Login only. This is a session control value stating that a connection should be maintained to the server between calls only as long as there is an ACMS sign in session or OpenVMS login session currently active.

**See Also:**
Constant Field Values

---

**LIFETIME_SESSION**
public static final int **LIFETIME_SESSION**

LIFETIME_SESSION maintains an open session for the duration of the JavaBean. This is a session control stating that a connection should be maintained to the server between calls, and until the javabean itself goes away.

**See Also:**
Constant Field Values

---

**TX_LIFETIME**
public static final int **TX_LIFETIME**
TX_LIFETIME is a combination of TX_SESSION and LIFETIME_SESSION.

**See Also:**
Constant Field Values

## 8.5 Constructor Detail

**ServerConfig**

public **ServerConfig**()
Creates new ServerConfig

## 8.6 Method Detail

**getBinding**

public java.lang.String **getBinding**()

Retrieve the Binding property. The Binding property indicates the connection binding being used to connect to the server. |ncacn_ip_tcp:16.32.0.1|wsicn_icc:GALAXY|

Note that value of "" is considered special, and indicate that the user's sharable will be loaded into the current process.

**Returns:**
Name of the binding to locate the java server.

**See Also:**
setBinding(String), isLocal()

---

**setBinding**

public void **setBinding**(java.lang.String value)
throws WsiException
Update the Binding property

**Parameters:**
value - New value for the Binding property

**Throws:**
WsiException

**See Also:**
getBinding(), #isNonTransport()

---

**setBinding**

public void **setBinding**(java.lang.String IPHost,
java.lang.String ICCHost)
throws WsiException

**Throws:**
WsiException

---

**setBinding**

public void **setBinding**()
throws WsiException

**Throws:**
WsiException

---

**getTxBinding**

public java.lang.String **getTxBinding**()

Retrieve the TX Binding property. The Binding property indicates the connection binding being used to connect to the commitment server.
|ncacn_ip_tcp:16.32.0.1|wsicn_icc:GALAXY|

**Returns:**
Name of the binding to locate the java server.

**See Also:**
setTxBinding(String), isLocal()

---

**setTxBinding**

public void **setTxBinding**(java.lang.String value)

Update the Binding property

**Parameters:**
value - New value for the Binding property

**See Also:**
getBinding(), #isNonTransport()

---

**setTxBinding**

public void **setTxBinding**(java.lang.String IPHost,
java.lang.String ICCHost)

---

**getMajorVersion**

public int **getMajorVersion**()

Retrieve the Major Version property.

**Returns:**
Major Version value for the connection.

---

**setMajorVersion**

public void **setMajorVersion**(int value)

Update the Major Version property

**Parameters:**
value - New value for the Major Version property

---

**getMinorVersion**

public int **getMinorVersion**()

Retrieve the Minor Version property.

**Returns:**
Minor Version value for the connection.

---

**setMinorVersion**

public void **setMinorVersion**(int value)

Update the Minor Version property

**Parameters:**
value - New value for the Minor Version property

---

**getLeaseTimeout**

public int **getLeaseTimeout**()

Retrieve the Lease Timeout property.

**Returns:**
lease timeout value for the connection.

---

**setLeaseTimeout**

public void **setLeaseTimeout**(int value)

Update the Lease Timeout property

**Parameters:**
value - New value for the Lease Timeout property

---

**isLocal**

public boolean **isLocal**()

Indicate if the NativeShell instance is to be local to this JVM or at a remote RMI server.

**Returns:**
true, if the JNI contact is to be in this JVM, and false, if not.

**See Also:**
#getHostName(), #setHostName(String)

---

**getAppName**

public java.lang.String **getAppName**()

Getter for property appName.

**Returns:**
Value of property appName.

---

**setAppName**

public void **setAppName**(java.lang.String value)

Setter for property appName.

**Parameters:**
appName - New value of property appName.

---

**getAppUuid**

public java.lang.String **getAppUuid**()

Getter for property appUuid.

**Returns:**
Value of property appUuid.

---

**setAppUuid**

public void **setAppUuid**(java.lang.String value)

Setter for property appUuid.

**Parameters:**
appUuid - New value of property appUuid.

---

**getDomain**

public java.lang.String **getDomain**()

Getter for property Domain.

**Returns:**
Value of property Domain.

---

**setDomain**

public void **setDomain**(java.lang.String value)

Setter for property Domain.

**Parameters:**
Domain - New value of property Domain.

---

**isAppMultithreaded**

public boolean **isAppMultithreaded**()

**Returns:**

---

**isTransport**

public boolean **isTransport**()

Return whether the server sharable is to be loaded into the current process space or not.

---

**setSessionType**

public void **setSessionType**(int value)
throws java.lang.IllegalArgumentException

**Parameters:**
value –

**Throws:**
java.lang.IllegalArgumentException

---

**setSessionType**

public void **setSessionType**(java.lang.String value)
throws java.lang.IllegalArgumentException

**Parameters:**
value –

**Throws:**
java.lang.IllegalArgumentException

---

**getSessionType**

public int **getSessionType**()

**Returns:**

---

**toString**

public java.lang.String **toString**()

Custom string formatting method.

**Overrides:**
toString in class java.lang.Object

---

**addPropertyChangeListener**

public void **addPropertyChangeListener**(java.beans.PropertyChangeListener l)

Add a PropertyChangeListener to the listener list.

**Parameters:**
l - The listener to add.

---

**removePropertyChangeListener**

public void **removePropertyChangeListener**(java.beans.PropertyChangeListener l)

Removes a PropertyChangeListener from the listener list.

**Parameters:**
l - The listener to remove.

---

**getTcpipName**

public java.lang.String **getTcpipName**()

---

**setTcpipName**

public void **setTcpipName**(java.lang.String mIPHostName)

---

**getScsnodeName**

public java.lang.String **getScsnodeName**()

---

**setScsnodeName**

public void **setScsnodeName**(java.lang.String mICCHostName)

---

**getTxTcpipName**

public java.lang.String **getTxTcpipName**()

---

**setTxTcpipName**

public void **setTxTcpipName**(java.lang.String mTxIPHostName)

---

**getTxScsnodeName**

public java.lang.String **getTxScsnodeName**()

---

**setTxScsnodeName**

public void **setTxScsnodeName**(java.lang.String mTxICCHostName)

# 9 Other Exceptions or Errors

This section describes some common exceptions or errors and their solutions. For more information about any other exceptions or erros, see “Advanced Debugging Techniques” in *Web Services Integration Toolkit for OpenVMS Developer’s Guide.*

## *9.1 IllegalArgumentException*

This exception can be seen for the following reasons:

- When the variables NO_SESSION, TX_SESSION & LIFETIME_SESSION take values other than 0, 1 & 2 respectively. This class allows the client to specify properties that affect instantiation and session control for out-of-process connections. Every WSIT generated by JavaBean optionally takes one of these objects. The properties that can be specified within this context object include:

  ```
  public  static final int        NO_SESSION = 0;
  ```

  Session control value states that a connection should be maintained to the server between calls only as long as there is an ACMS sign in session or OpenVMS login session currently active.

  ```
  public  static final int        TX_SESSION = 1;
  ```

  Session control value states that a connection should be maintained to the server between calls, and until the JavaBean itself go away.

  ```
  public  static final int        LIFETIME_SESSION = 2;
  ```

  To improve performance in the NO_SESSION and TX_SESSION cases, a lease timeout value (in seconds) can be specified to keep often used connections readily available. These connections only need to be re-established if left idle for longer than the timeout period. Session Lifetime specification and an associated Lease Timeout Session control is used to specify the duration or type of connection between the JavaBean and the Server Wrapper when out-of-process communication is used. Session control value states that no connection should be maintained to the server between calls.

  The message “sessionType must be NO_SESSION, TX_SESSION, LIFETIME_SESSION or TX_LIFETIME” will be seen in this case.

- An attempt is made to shorten the internal encoding buffer size. The new size must be at least as large as the amount currently in use. The message “Attempt to shorten encoding buffer” will be seen in this case.

## *9.2 No valid Protocol Found*

This error message is seen for the following reasons:

- Bind to a manager fails for all valid protocol sequences
- Bind to the specified server fails for all valid protocol sequences
- Server start, which starts all of the protocol sequences fails

## 9.3 UnsatisfiedLinkError: No WSI$JNISHR in java.library.path

Exception in thread "main" java.lang.UnsatisfiedLinkError: no WSI$JNISHR in java.library.path
    at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1578)

The most likely cause of this problem is that WSIT was not started, or is not running. The WSIT startup procedure does a number of things, including the defining of the system wide logical, WSI$JNISHR.

The solution is to run the WSIT startup procedure.

## 9.4 WsiConnectException: Cannot allocate Locate Request message

```
com.hp.wsi.WsiConnectException: ERROR: Cannot allocate Locate Request
message No valid protocol found at
com.hp.wsi.WsiJNIShell.n_attach(Native Method)
```

This indicates that the WSI$MANAGER process is not running or is hung. The Manager is started by the WSIT startup procedure, but may have stopped for some reason. If you would like to determine why the Manager was stopped, you can review the log file, sys$manager:wsi$manager.log.

The solution is to restart the Manager by rerunning the WSIT startup procedure.

## 9.5 WsiConnectException: Transceive failure EndPointLocate

### 9.5.1 User Sharable Not Deployed

```
com.hp.wsi.WsiConnectException: ERROR: Transceive failure
EndPointLocate: %WSI-F-FAILED_LOADING, Attempt to dynamically load user
image failed at com.hp.wsi.WsiJNIShell.n_attach(Native Method)

                      OR

com.hp.wsi.WsiServerException: %LIB-E-ACTIMAGE, error activating image
DKA0:[SYS0.SYSCOMMON.WSIT.][DEPLOY]MYAPP.EXE;
        at com.hp.wsi.WsiJNIShell.n_getContext(Native Method)
```

Both of the above errors indicate that the user's image is not correctly deployed to the WSIT deploy directory, or it is not loadable using the current account. The top error is given when running the server out-of-process, while the bottom is given when running the server in-process.

The solution is to first verify that the image exists within the wsi$root:[deploy] directory. If it is there, verify that the image's protections are not keeping the current account from accessing the image.

### 9.5.2 WSI$COMMON_SS.EXE Not Installed

```
com.hp.wsi.WsiConnectException: ERROR: Transceive failure
EndPointLocate: %SYSTEM-F-PROTINSTALL, protected images must be
installed at com.hp.wsi.WsiJNIShell.n_attach(Native Method)

                        OR

com.hp.wsi.WsiServerException: %LIB-E-ACTIMAGE, error activating image
WISDEM$DKA0:[SYS0.SYSCOMMON.][SYSLIB]WSI$COMMON_SS.EXE;1
        at com.hp.wsi.WsiJNIShell.n_getContext(Native Method)
```

Both errors indicate that the required WSIT installed sharable (wsi$common_ss.exe) is not correctly installed. The top error is given when running the server out-of-process, while the bottom is given when running the server in-process. The WSIT startup procedure installs this image.

The solution is to rerun the WSIT startup procedure to have it re-install the sharable image. If this does not work, then the sharable can be installed manually by using the install utility as follows:

```
$ install
Install> add/open/head/prot/shared
sys$library:wsi$common_ss.exe
Install> exit
```

## *9.6 java.lang.NoClassDefFoundError: MyApp/IMyApp*

```
Exception in thread "main" java.lang.NoClassDefFoundError: MyApp/IMyApp
```

This indicates that the jar file containing the user's javabean interface is not specified in the classpath, or the jar file is missing. (Note that this jar file was built when the generated javabean component was built.)

The solution is to find & add this jar file to the classpath. WSIT provides a tool to help with this. You can run it as follows:

```
$ @wsi$root:[tools]wsi-setenv -  Disk:[MyDir.MyApp]MyApp.jar
                    "JAVA$CLASSPATH" = "WSI$ROOT:[LIB]WSIRTL.JAR"
(LNM$JOB_894D2E00)
                                = "[]"
                                = "WSI$ROOT:[LIB]VELOCITY-DEP-1_4.JAR"
                                = "WSI$ROOT:[TOOLS]IDL2CODE.JAR"
                                = "Disk:[MyDir.MyApp]MyApp.jar "
```

## *9.7 java.lang.NoClassDefFoundError: com/hp/wsi/WsiException*

```
Exception in thread "main" java.lang.NoClassDefFoundError:
com/hp/wsi/WsiException
```

This indicates that the WSIT runtime jar file (wsirtl.jar) is not correctly in the classpath. Either the file is not specified in the classpath, or the jar file is missing. This jar file can be found in wsi$root:[lib]wsirtl.jar. However, if running within an application server environment, such as JBoss, Tomcat, or WLS, this file should be in that server's common/lib area as well.

The solution is to find & add this jar file to the classpath. WSIT provides a tool to help with this. You can run it as follows:

```
            $ @wsi$root:[tools]wsi-setenv -  "”
                 "JAVA$CLASSPATH" = "WSI$ROOT:[LIB]WSIRTL.JAR"
(LNM$JOB_894D2E00)
                                 = "[]"
```

## *9.8 Inconsistent use of Protocols*

```
com.hp.wsi.WsiException: Inconsistent use of Protocols
        at com.hp.wsi.WsiJNIShell.n_getContext(Native Method)
```

This indicates that the client was attempting to run the application both in-process, and out-of-process. WSIT allows you to choose one or the other at runtime, but not both at the same time. Once the user's server application has been loaded, it cannot be loaded differently until the client application is restarted. This includes application server environments, such as Tomcat, which act as WSIT clients. (Note that this is an OpenVMS restriction.)

The solution is to decide whether to use in-process or out-of-process for an application, then stay consistent with that decision when instantiating your application.

## *9.9 DOM Error during parsing*

The error message “DOM Error during parsing! DOM can't find document root” would be reported in wsi$manager.log when there is some issues while parsing the .WSI file. When wsi$manager.exe starts, it parses all .WSI file present inside wsi$root:[deploy] directory.

The error message would also print the name of .WSI file which would help indicate the problematic WSI file.

DOM Error indicates that the error occurred while parsing the .WSI file. The DOM error can occur while parsing the WSI file in the following two scenarios:

- When the file is empty
- When there is a syntax error in root record (First record) of .WSI file. As WSI file is a XML file, it should adhere to the syntax of XML.

User should rectify the syntax error in the root node or make sure that the file is not empty.

# 10 Serialized Form

| Package com.hp.wsi |
| --- |
| Class com.hp.wsi.InvalidStateException extends WsiException implements Serializable |
| Class com.hp.wsi.NoServerException extends WsiConnectException implements Serializable |
| Class com.hp.wsi.ServerConfig extends java.lang.Object implements Serializable |

## *10.1 Serialized Fields*

**propertySupport**

java.beans.PropertyChangeSupport **propertySupport**

---

**mAppName**

java.lang.String **mAppName**

---

**mAppUuid**

java.lang.String **mAppUuid**

---

**mIPHostName**

java.lang.String **mIPHostName**

---

**mICCHostName**

java.lang.String **mICCHostName**

---

**mTxIPHostName**

java.lang.String **mTxIPHostName**

---

**mTxICCHostName**

java.lang.String **mTxICCHostName**

---

**mBinding**

java.lang.String **mBinding**

---

**mTxBinding**

java.lang.String **mTxBinding**

---

**mDomain**

java.lang.String **mDomain**

---

**mIsAppMultithreaded**

boolean **mIsAppMultithreaded**

---

**mIsTransport**

boolean **mIsTransport**

---

**mMaxAppThreads**

int **mMaxAppThreads**

---

**mLeaseTimeout**

int **mLeaseTimeout**

---

**mSessionType**

int **mSessionType**

---

**mMajorVersion**

int **mMajorVersion**

---

**mMinorVersion**

int **mMinorVersion**

Class com.hp.wsi.WsiConnectException extends WsiException implements Serializable

Class com.hp.wsi.WsiException extends java.rmi.RemoteException implements Serializable

**m_excCode**

int **m_excCode**
privately stored exception codes.

---

**m_vmsCode**

int **m_vmsCode**

---

**m_facility**

int **m_facility**

---

**m_severity**

int **m_severity**

Class com.hp.wsi.WsiServerException extends WsiException implements Serializable