TIBCO iProcess® Server Objects (Java)

Release Notes

Version 11.10.0 | May 2025

Contents

Release Notes

Check the TIBCO Product Support web site at http://support.tibco.com for product information that was not available at release time. Entry to this site requires a username and password. If you do not have a username, you can request one. You must have a valid maintenance or support contract to use this site.

- New Features
- Other Information
- Closed Issues

New Features

The following new features have been introduced in each release of TIBCO iProcess® Server Objects (Java).

> **Note:** New features are not being added to the EJB interface from version 10.3.0 forward. To make use of new features, you must program to one of the other interfaces.

> **Note:**
>
> ## Earlier Versions
>
> For details of the MRs/CRs implemented in earlier versions of the product, please see the Readme and Release Notes for that particular product release.

The following subsections list the new features in each release.

Version 11.10.0

The following feature has been added in this release of TIBCO iProcess® Server Objects (Java).

Support for Java 17

TIBCO iProcess® Server Objects (Java) now supports Java 17.0.12.

Version 11.9.0

The following feature has been added in this release of TIBCO iProcess® Server Objects (Java).

Encryption Improvements

From this release, the Encryption protocol that is used between iProcess Server Objects and iProcess Objects Server is updated.

Version 11.4.1

New Parameter on Close and Purge Case Methods to Specify Priority (IPSO-242)

A new *aPriority* parameter has been added to the following methods:

- closeCases
- closeCasesBy Criteria
- purgeCases
- purgeCasesBy Criteria
- purgeAndReset

This new parameter can be used to increase or decrease the priority at which cases are closed or purged.

Password Size Limit Has Been Increased (IPSO-239)

If the User Validation API (UVAPI) is used to validate users, the current size limit of 4K may not be large enough to accommodate size of the token. Therefore, the password size limit has been increased to 32K.

Message Interface Versions are now Available Through the API (IPSO-232)

The TIBCO iProcess Objects Server and TIBCO iProcess Server Objects (Java) message interface versions are now available via the following new methods:

- sBase.getServerMsgInterfaceVer
- sBase.getClientMsgInterfaceVer

These message interface versions can be used by GUIs and applications to conditionally show functionality based on the services available in the TIBCO iProcess Objects Server and TIBCO iProcess Server Objects (Java) client.

IPv6 IP Addressing is Now Supported (IPSO-230)

When constructing a **vNodeId** object for connecting to a TIBCO iProcess Objects Server or Director, IPv6 addressing can now be used if the TIBCO iProcess Objects Server or Director is using an IPv6 address.

New Parameter on Close and Purge Case Methods to Trigger Event (IPSO-229)

A new *aDoEvents* parameter has been added to the following methods:

- closeCases
- closeCasesBy Criteria
- purgeCases
- purgeCasesBy Criteria
- purgeAndReset

This new parameter can be used to specify that a procedure-level event is to be triggered when cases are closed or purged with these methods. The procedure can be defined to catch the event, then perform business logic either before or after the cases are closed or purged.

Version 11.4.0

Unlimited Number of Case Numbers Now Possible (IPSO-202)

The case number has been modified internally to allow for an unlimited number of possible case numbers (previously, a limit could be reached).

Note that this has no impact on the publicly available/visible case number (for example, in SW_CASENUM).

Added Ability to Change Client Log File Name via Registry/Environment Variable (IPSO-45)

The following new Registry entry (Windows) and Environment Variable (UNIX) are now created by TIBCO iProcess Server Objects (Java) that allow the specification of the client log file name via the Registry/Environment Variable (it can also be specified programmatically using the **setLogId** method on the **sClientLog** object):

- Registry Entry - LogId
- Environment Variable - SSOClient_LogId

Changes in Platform Support

Supported platforms are not affected in this release. For a complete list of supported platforms, see the Readme file.

Change in Functionality

No functionality changes have been made in this release of TIBCO iProcess® Server Objects (Java).

Deprecated and Removed Features

No features have been deprecated or removed in this release of TIBCO iProcess® Server Objects (Java).

Other Information

Rounding of Large Values May Occur

Because of the data types used in the TIBCO iProcess Server Objects (Java), rounding may occur in data fields that contain very large numbers. The floating point precision provided in the data types used is 15-17 digits of accuracy.

For more information about this issue, see the following websites:

— http://msdn2.microsoft.com/en-us/library/Aa691146(VS.71).aspx

— http://mindprod.com/jgloss/floatingpoint.html

Encoding Using ICU Conversion Libraries

ICU conversion libraries can be used to specify the desired character encoding.

To use the ICU conversion libraries, you must create the following environment variable (UNIX systems) or Registry entry (Windows systems) and set it to the name of the converter you wish to use.

— TISOUnicodeConverterName

On Windows systems, the Registry entry must be located in the following path:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Staffware plc\Staffware SSO Client\
```

Note: If the software is installed on a 64-bit machine, the Registry path will include "Wow6432Node" as follows:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Staffware plc\...
```

For a list of converter names and information about each converter, see List Converter.

Note: When using the ICU libraries, the converter you use must reserve positions 00 through 1F for the standard single-byte ASCII control characters. This ensures that the control characters do not otherwise occur in the byte stream. (The UTF-16 converter, for example, does not satisfy this requirement, and therefore, cannot be used.)

UNIX Systems

If the **TISOUnicodeConverterName** environment variable does not exist, or is set to an invalid value, the ICU libraries are not used. In this case, the system looks for the **TISOMultiChar** environment variable. If the TISOMultiChar environment variable exists and is set to 1, UTF-8 (multi-byte) encoding is used, otherwise extended ASCII (single-byte) encoding is used. The system will only look at the TISOMultiChar environment variable if the TISOUnicodeConverterName environment variable does not exist or is set to an invalid converter name.

Windows Systems

If the **TISOUnicodeConverterName** Registry entry does not exist, or is set to an invalid value, the ICU libraries are not used. In this case, the system looks for the **TISOMultiChar** Registry entry in the following path:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Staffware plc\Staffware SSO Client\
```

If the TISOMultiChar Registry entry exists and is set to 1, UTF-8 (multi-byte) encoding is used, otherwise extended ASCII (single-byte) encoding is used. The system will only look at the TISOMultiChar Registry entry if the TISOUnicodeConverterName Registry entry does not exist or is set to an invalid converter name.

For more information, see ICU.

Upgrading May Result in Two DLL Files

[Windows only.]

Upgrading from pre-version 10.3.1 TIBCO iProcess Server Objects (Java) may result in two **ssoJNI.dll** files on your system. The pre-version 10.3.1 installation program copied the **ssoJNI.dll** file to either the **WINNT** or **system32** directories, depending on how the environment variables were set up. Version 10.3.1 and newer installation programs copy the **ssoJNI.dll** file to the specified installation directory.

If you are upgrading from a pre-version 10.3.1, manually delete the **ssoJNI.dll** file from the **WINNT** or **system32** directory.

Raw Data Buffers

The **sRawData** object allows you to obtain raw data directly from the message buffers. The **sRawData** object contains the following public constructor:

```
public sRawData(vNodeId aNodeId,
                String aUserName,
                String aPassword)
                   throws vException
```

The **sRawData** object can also be obtained using the **create_sRawData** method on the **sSession** object.

The **sRawData** object contains the following methods:

- **getFirstWIBuffer** - This method returns the raw data for the specified number of work items from the specified work queue. There are two signatures for this method, as follows:

  ```
  public synchronized byte[] getFirstWIBuffer(String aWorkQTag,
                                              int aMaxWICnt)
                                              throws vException
  public synchronized byte[] getFirstWIBuffer(String aWorkQTag,
                                   vWICriteria aCriteria,
                                   String[] aCaseFieldNames,
                                   int aMaxWICnt)
                                   throws vException
  ```

  where:

- — **aWorkQTag** is the tag identifying the work queue from which the work items are to be retrieved.
- — **aCriteria** is a **vWICriteria** object, which specifies the filter and sort criteria for the request.
- — **aCaseFieldNames** specifies the case data fields to be returned.
- — **aMaxWICnt** is the number of work items to return from the call to **getFirstWIBuffer**. Passing a -1 in this parameter causes all work items in the buffer to be retrieved.

- **getNextWIBuffer** - This method returns the next block of work items (the size of which is specified in the *aMaxWICnt* parameter with the **getFirstWIBuffer** method). This method returns a null if the entire message has been received and is complete. Its syntax is as follows:

```
public synchronized byte[] getNextWIBuffer()
                                  throws vException
```

- **releaseBuffers** - This method releases any segments if the application did not retrieve all of the raw data from the buffer (i.e., the **getNextWIBuffer** method wasn’t called until a null was returned). Its syntax is as follows:

```
public synchronized void releaseBuffers()
```

Note: If **getFirstWIBuffer** is called again before all of the work items are retrieved from the buffer, the previous raw data is automatically released from the buffer before the work items are retrieved from the server again.

Raw Data Buffer Format

The following describes the format of the raw data buffers so that they can be parsed after acquiring the raw data with the **getFirstWIBuffer** and **getNextWIBuffer** methods.

The message format makes use of the following special byte values:

- ELEMDELIM = Chr(0)
- LISTTERM = Chr(30)
- RECTERM = Chr(29)

- COUNTMARK = Chr(28)

where:

— ELEMDELIM is used to delimit all elements of the message.

— LISTTERM is used to terminate a list.

— RECTERM is used to terminate the data for a record (work item in this case).

— COUNTMARK is used to locate the values of the counters in the message.

The size of each buffer is controlled by the **TCPResponsePages** parameter in the TIBCO iProcess Objects Server. The format of the data in each buffer is a series of data elements. All elements of the message (including the LISTTERM, RECTERM and COUNTMARK) are delimited by the ELEMDELIM character.

All message elements are printable ASCII values. In other words, values of type date, time or numeric are returned as their ASCII string equivalents and not as their native binary values.

The response format of the QQ message is listed below and uses the following syntactical notation.

Syntax	Description
element1, element2	element1 and element2 are separated by NULL (0x00).
{element1, element2, element3, ...}	There are 0 or more occurrences (a list) of the ordered tuple in the data stream. The list is terminated with the LISTTERM, (0x1E) and a NULL (0x00). Lists can be embedded (lists can contain lists). Each list is terminated with 0x1E and 0x00.
element	Elements in **bold** are literal values.
RECTERM	This delimits the end of a logical record (a workitem in this case). The value of the delimiter is 0x1D followed by a NULL (0x00).
LISTTERM	This delimits the end of a list. The value of the delimiter is 0x1E followed by a NULL (0x00).
COUNTMARK	This is used to indicate the beginning of the count data. The value of the delimiter is 0x1C followed by a NULL (0x00).

Each buffer (or message segment) returned by the server has as its first element the message code of the request. For the **getFirstWIBuffer** method, that message code will always be the value QQ. When the application parses the response buffers, it needs to skip the QQ value that is at the beginning of each message segment.

The basic strategy to parsing a list is to grab the first element of the list. If the value is LISTTERM, then you are at the end of the list. If the value is not LISTTERM, then that is the value of the first element; the subsequent elements comprise the remaining elements of the list.

Individual elements and their trailing delimiter never span message segments. However, a logical record (meaning the elements that comprise a work item) can span message segments. This spanning between elements can occur at any point in the logical record.

The format of date and time values is controlled by the TIBCO iProcess Engine settings. You can see the TIBCO iProcess Engine date and time settings via the **getDateFormat**, **getDateSeparator**, and **getTimeSeparator** methods on the **vNodeLocale** object. These are configured in the **staffpms** file. If a date is uninitialized, the value is December 31, 3000.

Some elements of a message are only sent by the TIBCO iProcess Objects Server if both the client and server are at a particular interface level or newer. If this applies to an element, it is notated in the format specification below with the minimum interface level of the client and server that would be needed for the element to appear in the data stream. The current elements this impacts are **IsKeepOnWithdraw**, **isSuspended**, **ProcMajorVer**, **ProcMinorVer**, **FieldIsArray**, **CDQPName**, and **CDQPValue**.

Note:
The data buffer format described here is based on a TIBCO iProcess Objects Server message interface of 3.1.0, and a TIBCO iProcess Server Objects client message interface of 3.1.0. If you upgrade to a newer server and client in the future, you may start receiving new data elements in your message (depending on the message interface versions of the client and server). If this occurs, you will need to adjust your parsing logic to accommodate them. You can determine the TIBCO iProcess Objects Server's interface version by running:

```
$SWDIR/bin/swentobjsv -V
```

You can determine the TIBCO iProcess Server Objects Client's interface version by calling the **getInterfaceVersion** method on **vClientInfo**.

Data Buffer Format

The following is the format of the data stream contained in the buffers:

ComputerName, Status {IsUnopen, IsLocked, IsDeadline, IsDeadlineExp, IsStartStep, IsUndelivered, IsAvailable, IsLongLock, IsUrgent, IsAutoPurge, IsWorkdays, IsNetworked, IsEditable, IsForwardable, IsReleasable, IsDeadlineAWD, IsKeepOnWithdraw, IsSuspended, ArrivedDate, ArrivedTime, DeadlineDate, DeadlineTime, CaseDescription, StepName, StepDescription, CaseNumber, ProcName, ProcDescription, ProcNumber, CaseReference, ProcNode, WQParam1, WQParam2, WQParam3, WQParam4, Priority, LockedBy, StartedBy, AddrToName, MailID, ExtraPackFile, ExtraStepNum, ExtraCaseCtlRec, ExtraReqID, ExtraPNumCnt, ProcMajorVer, ProcMinorVer, {FieldName, FieldType, FieldIsArray, FieldLength, FieldDecimalPlaceCnt, FieldValue}, {CDQPName, CDQPValue}, RECTERM}, COUNTMARKER, Count, OverMaxCnt, InvalidCnt, ExcludeCnt

where:

ComputerName = Name of the computer where server is running

Status = you will always get a value of **2**

IsUnopen = **1** if item is unopened, **0** if item is not unopened

IsLocked = **1** if item is locked, **0** if item is not locked

IsDeadline = **1** if item has deadline, **0** if no deadline

IsDeadlineEXP = **1** if deadline has expired, **0** if not expired

IsStartStep = **1** if item from case start, **0** if not

IsUndelivered = **1** if item is undelivered, **0** if not

IsAvailable = **1** if item is available, **0** if not (orphaned)

IsLongLock = **1** if item is long locked, **0** if not

IsUrgent = **1** if item is urgent, **0** if not

IsAutoPurge = **1** if item's case is auto-purge, **0** if not

IsWorkdays = **1** if item's case uses workdays, **0** if not

IsNetworked = **1** if item's procedure is networked, **0** if not

IsEditable = **1** if item is editable, **0** if not

IsForwardable = **1** if item is forwardable, **0** if not

IsReleasable = **1** if item is directly releasable, **0** if not

IsDeadlineAWD = **1** if item is auto-withdrawn on deadline expiration,

0 if not

IsKeepOnWithdraw = **1** if deadline NOP, **0** if not (interface 3.0.0)

IsSuspended = **1** if item is suspended, **0** if not (interface 2.0.4)

ArrivedDate = Date item arrived in queue (in server format)

ArrivedTime = Time item arrived in queue (in server format)

DeadlineDate = Deadline date (in server format)

DeadlineTime = Deadline time (in client format)

CaseDescription = Case description for case

StepName = Name of step for work item

StepDescription = Description of step for work item

CaseNumber = Case number of case for work item

ProcName = Name of procedure for work item

ProcDescription = Description of procedure for work item

ProcNumber = Internal number of procedure for work item

CaseReference = Case reference

ProcNode = Hosting node for procedure

WQParam1 = Value for work queue parameter 1

WQParam2 = Value for work queue parameter 2

WQParam3 = Value for work queue parameter 3

WQParam4 = Value for work queue parameter 4

Priority = Priority of work item

LockedBy = User who has item locked

StartedBy = User who started the case

AddrToName = Addressee of work item

MailID = An ID unique to this item in this queue only

ExtraPackfile = Pack file name for work item

ExtraStepNum = Number of step within procedure

ExtraCaseCtlRec = Case control record number

ExtraReqID = Unique Identifier for this item

ExtraPNumCnt = Slot usage for ProcNum

ProcMajorVer = Major version number of case that work item is from

(interface 3.0.0)

ProcMinorVer = Minor version number of case that work item is from

(interface 3.0.0)

FieldName = Name of case data field

FieldType = **D** for date

= **A** for ASCII text

= **R** for real number

= **T** for time

= **X** for attachment

= **F** for memo

= **C** for composite (table)

= **N** for comma separated (numeric)

FieldIsArray = **1** if field is an array field, **0** if not (interface 3.0.0)

FieldLength = Length of field

FieldDecimalPlaceCnt = Number of decimal places (for real number and comma

separated)

FieldValue = Value of field

CDQPName = Name of the CDQP field (interface 1.1.1)

CDQPValue = Value of the CDQP field (interface 1.1.1)

Count = Number of items returned (value will always be 0 if

MaxCount <> -1)

OverMaxCnt = Number of items matching expression not returned

because of Max Count specification

InvalidCnt = Number of items not returned because expression was

invalid in the case context

ExcludeCnt = Number of items not returned because the expression

explicitly excluded them

Buffer Data in the Log File

If you want to see what some sample buffers would look like, you can enable MsgTrace in the TIBCO iProcess Objects Server and look at the trace dumps in the log file for a QQ message (which you can generate by iterating calling the getWorkItems method). If you do this, note that each message segment as traced in the TIBCO iProcess Objects Server log file contains a header. This header is stripped off by the **getFirstWIBuffer** and **getNextWIBuffer** methods. This header on each segment is a fixed 17-byte header containing the following elements:

Description	Size	Format
Total Segment Length (not counting this field)	4 bytes fixed	Unsigned Binary in Network Byte Order
Request ID (echoed from client request)	4 bytes fixed	Unsigned Binary in Network Byte Order

Segment ID	1 byte fixed	F = First segment
		M = Middle segment
		L = Last segment
		O = Only segment

Transaction Status (only applicable if Segment ID is L or O). Positive on success, 0 or negative on error.	4 bytes fixed	Signed Binary in Network Byte

Description	Size	Format
		Order
Total Cumulative Length (total number of bytes in all message fragments for this response). Only applicable if Segment ID is L or O.	4 bytes fixed	Unsigned Binary in Network Byte Order

External Rebind Needed When Using WebSphere JAAS Authentication

When using WebSphere JAAS authentication, you must disable the internal rebind and allow an external rebind to the **rsServerFactoryImpl**. This is because with WebSphere security enabled for binding to the naming service, the **InitialContext rebind()** method cannot be called outside of a “privileged” context. The following is provided to allow this:

- When the following command-line argument is specified, the class does not rebind to the naming service.

  ```
  -externalRebind
  ```

- The following static method is provided to get a handle to the **rsServerFactoryImp** instance.

  ```
  public static rsServerFactoryImpl getExternalRebindHandle()
  ```

  This returns the **rsServerFactoryImpl** object so that it can be externally bound to the naming service within the **PrivilegedAction** context of the authenticated subject.

You need to create a class that performs the required WebSphere JAAS authentication to login. This class must call the **rsServerFactoryImpl.main(String[] args)** method (with the **-externalRebind** option), then externally bind the **rsServerFactoryImpl** object to the **InitialContext** in the required **PrivilegedAction** context.

The steps shown below provide a summary of the requirements to login to WebSphere using JAAS and bind the **rsServerFactoryImpl** object.

> **Note:**
> The examples shown in the following steps are taken from the website that explains WebSphere Java Authentication and Authorization Service (JAAS).

Procedure

1. Configure the server for JAAS. For information about how to do this, see the link above.

2. Create the **InitialContext** required for the **SecurityServer**:

   ```
   Hashtable env = new Hashtable();
   env.put(Context.INITIAL_CONTEXT_FACTORY,"com.ibm.websphere.
               naming.WsnInitialContextFactory");
   env.put(Context.PROVIDER_URL,"corbaloc:iiop:myhost.mycompany.
               com:2809");
   Context initialContext = new InitialContext(env);
   Object obj = initialContext.lookup("");
   ```

3. Create a **LoginContext**. This can use one of three **CallbackHandler** classes or a custom handler class:

   — **Non-prompt:**

   ```
   LoginContext lc = new LoginContext("WSLogin",
         new WSCallbackHandlerImpl("userName", "realm",
   "password"));
   ```

   — **GUI prompt:**

   ```
   LoginContext lc = new LoginContext("WSLogin",
          new WSGUICallbackHandlerImpl());
   ```

   — **Stdin prompt:**

   ```
   LoginContext lc = new LoginContext("WSLogin",
          new WSStdinCallbackHandlerImpl());
   ```

4. Do the login:

   ```
   lc.login();
   ```

5. Get the authenticated Subject object from the **LoginContext**:

```
Subject s = lc.getSubject();
```

6. Call **rsServerFactoryImpl.main(String[] args)** with the **-externalRebind** option.
    - All the normal command-line arguments are passed in here except for the **-name=**_FactoryName_ argument. The name used for binding is specified in the **rebind(..)** method in **RegisteredName** (see step 7).

> **Note:** If a name other than "ssoServerFactory" is used, the delegate classes must use methods that specify this name also.

- This initializes the **rsServerFactoryImpl** without doing the **ctx.rebind()** and allows access to the object via the static method:

```
public static rsServerFactoryImpl getExternalRebindHandle()
```

7. Create a **PrivilegedAction** object. This will do the **context.rebind(..)**:

```
PrivilegedAction action = PrivilegedAction() {
     public Object run() {
       initialContext.rebind(RegisteredName,
         rsServerFactoryImpl.getExternalRebindHandle());
   }
}
```

8. Do the **PrivilegedAction** using the authenticated subject:

```
WSSubject.doAs(s, action);
```

JBase Should Not be Run Under an Application Server

The RMI configuration of TIBCO iProcess Server Objects (Java) is J2EE compliant. Therefore, it can be run on any Application Server that meets the J2EE specification. Testing was performed on WebLogic, version 7.0.1.0.

Note, however, that we do not recommend running the Standalone interface (ssoJBase) under an Application Server for the following reasons:

- The Standalone interface is not J2EE compliant — the J2EE/EJB specifications prohibit compliant applications from running native code; ssoJBase makes native calls using JNI.
- The security offered by J2EE compliance is compromised.
- If the application crashes, so will the Application Server.
- Scalability is limited, as follows:
  - With TIBCO iProcess Server Objects (Java) running RMI, multiple **rsServerFactory** processes can be started as load requires and these can run on any machine. This is not possible using JBase.
  - Since the **rsServerFactory** runs in its own JVM if you use RMI, it can be tuned specifically for TIBCO iProcess Server Objects (Java) without concern for the requirements of other processes running under the same JVM. This is not possible with JBase.
- The application is not portable, as follows:
  - An application running under an Application Server is portable if it does not depend on O/S-specific libraries. With TIBCO iProcess Server Objects (Java) RMI, such an application is portable and can be deployed on an Application Server running on Windows, for example, with no modification required. The **rsServerFactory** process runs in its own JVM and it alone loads the O/S-specific libraries. In contrast, if TIBCO iProcess Server Objects (Java) JBase is running under an Application Server, the application is no longer portable because it would require loading the O/S-specific libraries to run.

Note: The last three bullet items above are also valid reasons ssoJBase should not be run under a web server.

Must Normalize Case Data to View Cases

When you install the iProcess Engine, there is a check box in the **Configuration** window that is used to enable a feature called **case data normalization**. If you are using an application developed with iProcess Server Objects, you **must** enable case data normalization. If it is not enabled, cases will not appear in the case list.

Case data normalization can be enabled either during the installation, or at a later time using the Case Data Normalization Utility, **swnormcd**.

For more information about case data normalization and using **swnormcd**, see the *TIBCO iProcess Engine Administrator's Guide*.

Cannot Run in Same Process as iProcess Objects

You cannot run TIBCO iProcess Server Objects in the same process as TIBCO iProcess Objects. Both products in the same process will not run properly, and is not supported.

Closed Issues

The table in this section lists issues that were closed in the named releases.

Closed in Release	MR #	Summary
11.10.0	IPSO-347 (*Parent id IPSO-326*)	RMI server hangs for all the subsequent requests once we get a Winsockerror.
11.10.0	IPSO-319 (*Parent id IPSO-303*)	IPSO/RMI service goes down, keeping the work items from an IPSO Client.
11.10.0	IPSO-320 (*Parent id IPSO-294*)	When UTF-8 is enabled, `libssoJNI undefined symbol` error occurs.
11.4.1	IPSO-242	A means to specify case close and purge priority needs to be added. Implemented. See New Parameter on Close and Purge Case Methods to Specify Priority (IPSO-242).
11.4.1	IPSO-239	The current size limit of 4K for passwords need to be increased. Implemented. See Password Size Limit Has Been Increased (IPSO-239).
11.4.1	IPSO-234	Some audit actions are missing in **SWAuditType**, which

Closed in Release	MR #	Summary
		results in the Action Processor returning exceptions to TIBCO iProcess Workspace (Browser). Corrected.
11.4.1	IPSO-232	The TIBCO iProcess Objects Server and TIBCO iProcess Server Objects (Java) message interface versions need to made available through the API. Implemented. See Message Interface Versions are now Available Through the API (IPSO-232).
11.4.1	IPSO-231	The XSD for **MaxCnt** on the **vACaseCriteria** object indicates that the count is required (minOccurs=”1”). It should indicate that it is optional (minOccurs=”0”). Corrected.
11.4.1	IPSO-230	Support for IPv6 IP addressing needs to be added for connecting iProcess Server Objects to the TIBCO iProcess Objects Server or Director. Implemented. See IPv6 IP Addressing is Now Supported (IPSO-230).
11.4.1	IPSO-229	A means to specify that a procedure-level event be triggered when closing and purging cases needs to be added. Implemented. See New Parameter on Close and Purge Case Methods to Trigger Event (IPSO-229) .
11.4.0	IPSO-202	Internally, the case number needs to be modified to allow for an unlimited number of possible case numbers. Implemented. See Unlimited Number of Case Numbers Now Possible (IPSO-202).
11.4.0	IPSO-194	A number of methods that contain a list as an argument

Closed in Release	MR #	Summary
		can cause a stack overflow. Corrected.
11.4.0	IPSO-188	Some of the legends on configuration values in the client log file are very similar to the environment variable names, but with different casing, which can cause confusion. All legends have been changed to be plain English so there is no confusion with environment variable names.
11.4.0	IPSO-182	There is a memory leak when **sWorkQ.getForwardToWorkQIds** is called. Corrected.
11.4.0	IPSO-178	The username should be included with messages sent to the iProcess Objects Director so that the Director can use the username when making load balancing decisions. Corrected.
11.4.0	IPSO-45	The ability to change the log file name via the Registry or Environment Variable should be added. Implemented. See Added Ability to Change Client Log File Name via Registry/Environment Variable (IPSO-45).

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The following documentation for this product is available on the TIBCO iProcess® Server Objects (Java) Product Documentation page.

How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to TIBCO Community.

Legal and Third-Party Notices

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at https://www.cloud.com/legal.