README for
TIBCO®Staffware Process Objects (SPO) C++ Client
for HP-UX
Version 10.2.2

Contents

1 About This Version

Version 10.2.2 of the TIBCO SPO C++ Client for HP-UX includes the enhancement that is described in section 3.1, as well as the fixes that are listed in section 7.1.

Note - Prior to Version 9.x, this product was called the Staffware Enterprise Objects (SEO) C++ Client. You may still see references to SEO within the software and in some technical documentation.

2 Compatibility

2.1 TIBCO SPO Server

Version 10.2.2 of the TIBCO SPO C++ Client is backward-compatible with older versions of the TIBCO SPO Server. However, if you are not using the most recent version of the TIBCO SPO Server, some of the features listed in the "New Features" section below may not be available to you.

2.2 TIBCO Staffware Process/iProcess Engine

The "type" of TIBCO Staffware Process/iProcess Engine you are using also determines whether or not new functionality described in the "New Features" section is available to you. The two types of engines are:

- **TIBCO Staffware iProcess Engine** - This is also referred to as the "TIBCO Staffware iPE Server." This type of engine is required for some of the new functionality (e.g., EAI steps, immediate case number availability, etc.) that is described in the "New Features" section. If you are using a TIBCO Staffware iProcess Engine, you will also be using a TIBCO SPO Server that supports the functionality provided by the TIBCO Staffware iProcess Engine.

- **TIBCO Staffware Process Engine** - This was previously known as the “Staffware Server.” If you are using this type of engine, some of the new functionality that is described in the “New Features” section (e.g., EAI steps, immediate case number availability, etc.) is not available to you. If you are using a TIBCO Staffware Process Engine, you will also be using a TIBCO SPO Server that supports the functionality provided by the TIBCO Staffware Process Engine.

The TIBCO SPO C++ Client will work with both “types” of TIBCO SPO Servers and Process/iProcess Engines described above.

2.2.1 Engine and Server Version Numbers

As we are transitioning from "Staffware" to "TIBCO," the version numbers of the engines and servers are changing as well. Staffware version numbers included major, minor, maintenance release, and patch numbers, with parentheses (e.g., 10.2(0.0)). TIBCO version numbers include major, minor, and maintenance release numbers, without parentheses (e.g., 10.2.0). Hotfix numbers (the equivalent to a "patch") are not shown in the product version number.

A Staffware version number may also be preceded by an "i" (e.g., i10.0(0.0)), indicating that it is an "iProcess" Engine or an SPO Server that supports the functionality offered by iProcess Engines.

Moving forward from version 10.2.0, all *new releases* of engines, SPO Servers, and SPO Clients will use the 3-digit TIBCO version numbering system. The version number will also not include an "i" to indicate that it is an iProcess Engine or an SPO Server that supports the functionality of an iProcess Engine; by default, all engines from 10.2.0 forward are iProcess Engines, and all SPO Servers from 10.2.0 forward support the functionality of iProcess Engines.

You can determine whether you are using a TIBCO Staffware Process Engine or a TIBCO Staffware iProcess Engine by looking at the version number. The version number can be found in the first line of the *SWDIR*\swdefs (Windows) or $SWDIR/**swdefs** (UNIX) file.

The following summarizes version numbers for engines and servers:

- **TIBCO Staffware iProcess Engine** - Prior to version 10.2.0, these engines had an "i" in the version number, e.g., i10.0-o(5.3). From version 10.2.0 forward, the version number will be 3 digits, with no "i". (The database supported will be indicated in parentheses following the version number, e.g., 10.2.0 (Oracle).)
- **TIBCO Staffware Process Engine** - These will continue to use the Staffware numbering system. Their version number begins with 8 or 9, with no “i”, e.g., 9.0-x(0.7).
- **TIBCO SPO Server** - Prior to version 10.2.0, these servers had version numbers either with or without an "i". SPO Servers without the "i" (e.g., 9.3(5.0)) are used with Process Engines; SPO Servers with an "i" (e.g., i10.0(4.0)) are used with iProcess Engines. From version 10.2.0 forward, the version number will be 3 digits, with no "i"; these will be used with iProcess Engines.

2.3 Must Link to Shared Library

Starting with version 9.3(5.0) of the TIBCO SPO C++ Client, your application must link to the shared library (**libSWEOCPP.sl**), rather than the previously used static library (**libSWEOCPP.a**).

Also, starting with version 10.0(0.1), there is no longer a soft link to **libSWEOCPP.sl** in **/usr/lib**. Instead, you must link with the copy of the library in the install directory. The correct way of doing this is to add the following to the command line when linking your application program:

```
-z -L/install/directory -lSWEOCPP
```

2.4 Compiler

This version of the TIBCO SPO C++ Client for HP-UX was compiled with the HP-UX C++ Compiler, version B3910B A.03.30.

3 New Features

3.1 Version 10.2.2

3.1.1 Support Added for the TIBCO SPO Director (CR 16970)

The TIBCO SPO C++ Client has been updated to support the TIBCO SPO Director, a standalone program that maintains a list of TIBCO SPO Servers that are configured in a node cluster. When a TIBCO SPO C++ Client needs access to a TIBCO SPO Server, it first establishes a connection to the TIBCO SPO Director. The TIBCO SPO Director then decides, based on a “pick method,” which TIBCO SPO Server the client should connect to.

For details about using and configuring the TIBCO SPO Director, see the *TIBCO SPO Programmer's Guide*.

3.2 Version 10.2.0

3.2.1 New TIBCO SPO Server Status Types (CR 16814)

The following constants have been added to the **SWNodeInfoStatusType** enumeration to describe the statuses of TIBCO SPO Servers made available by a TIBCO SPO Director:

Constant	Description	Value
swNotRunning	The TIBCO SPO Server is not running	’N’
swStarting	The TIBCO SPO Server is starting	’I’
swShuttingDown	The TIBCO SPO Server is shutting down	’D’
swStopped	The TIBCO SPO Server is stopped and not available	’S’
swServerSuspended	The TIBCO SPO Server is suspended	’U’
swNoResponse	The TIBCO SPO Server is not responding	’R’

3.2.2 New Error Constants Added (CR 16667)

The following new error constants can be returned to the TIBCO SPO C++ Client from the TIBCO SPO Server :

Dec.	Hex.	Constant	Message
34	0x0022	ER_SAL_INV_VALUE	Invalid process attribute value.
35	0x0023	ER_SAL_INV_FILTER	Invalid filter.
36	0x0024	ER_SAL_XACT_ABORT	Global transaction has been aborted.
37	0x0025	ER_SAL_COMPLEX	Filter is too complex.

3.2.3 New Methods Added to Retrieve Markings When Locking Work Items (CR 16177)

The following methods have been added to allow you to specify which markings to return from the TIBCO SPO Server, regardless of whether they are visible on the form or whether they are the result of conditional statements on the form:

- **lockItemMarkings** - This method on **SWWorkItem** contains a *MarkingNames* parameter that allows you to specify which markings to return from the server when the work item is locked. This allows you to control resources by specifying only the needed markings.
- **lockItemsMarkings** - This method on **SWWorkQ** contains a *MarkingNames* parameter that allows you to specify which markings to return from the server when the work items are locked. This allows you to control resources by specifying only the needed markings.

3.2.4 Extended Description Method Added to SWStep (CR 16065)

The **getDescriptionEx** method has been added to the **SWStep** object. This method returns the extended description that can be entered when defining a normal-type step in the SPD. This new method returns an empty string if the step type does not support an extended description.

3.2.5 Transaction Control Steps Added (CR 16045)

SPO has been modified to support a new step type — transaction control steps. Transaction control steps provide a mechanism, within a TIBCO procedure, to allow more transaction granularity within a sequence of EAI steps.

By default, the background process groups a series of connected EAI steps into one transaction. If a failure occurs in any EAI step in the series, the entire transaction is rolled back. A transaction control step can now be placed within the series of EAI steps to break the single transaction into multiple transactions. When the process flow reaches the transaction control step, the current transaction can be either committed and a new transaction started, or the current transaction can be aborted, depending on how the transaction control step has been defined in the SPD.

See the *Defining Procedures - Integration Techniques* manual for more information about placement and use of transaction control steps in a procedure.

The following changes have been made in SPO to support transaction control steps:

- **SWStepType** - The following constant has been added to this enumeration:

Constant	Description	Value
swTransControl	Transaction control step	‘T’

  This new constant is returned by the **getType** method on **SWStep** if the step is a transaction control step.
- **SWTransControlStep** - This new object represents an outstanding transaction control step. A transaction control step becomes outstanding when the process flow reaches the step. The transaction control step is no longer outstanding when the transaction started by the transaction control step is either committed or aborted. This new object contains the following methods:
  - **getArrived** - Returns the date and time the transaction control step became outstanding.
  - **getCaseNumber** - Identifies the case containing the outstanding transaction control step.
  - **getClassId** - Identifies the object class.

  - **getKey** - Returns the key for the step, in the form: ProcName|StepName|CaseNumber.
  - **getProcMajorVer** - Returns the MajorVersion# portion of the procedure's version number.
  - **getProcMinorVer** - Returns the MinorVersion# portion of the procedure's version number.
  - **getProcName** - Returns the name of the procedure with which the step is associated.
  - **getProcPath** - Provides the complete path to the outstanding transaction control step.
  - **getRetryTime** - Returns the date and time that the transaction will be retried if it fails.
  - **getStepName** - Returns the name of the transaction control step.
- **getTransControlSteps** - This new method on **SWCase** returns an **SWList** of **SWTransControlStep** objects (see above), one for each outstanding transaction control step in the case.
- **getTransControlType** - This new method on **SWStep** returns an enumeration constant (**SWTransControlType** - see below) that identifies the type of transaction control step. The type, which is specified when the step is defined in the SPD, are:
  - **Commit and Continue** - This type specifies that the current transaction be committed, and that a new transaction be started for subsequent steps using the same background process. The benefit of choosing this option is that it is faster, as the same process starts the new transaction.
  - **Commit and Concede** - This type specifies that the current transaction be committed, and that a new transaction be started for subsequent steps, except a different background process will be used for the second transaction.

    The background process processes the first transaction and updates the database. It then sends a message back to the Mbox where the messages are stored. Processing of the process continues when the background process (either the same one that processed the first transaction or another one) reads the message from the Mbox and processes it. The benefit of choosing this option is that it enables load balancing because a different background process can process the second transaction.
  - **Abort** - This option causes the abortion of the current transaction when the process flow reaches the transaction control step. This option is always used with a condition. This means that you can specify a condition on which the transaction should be rolled back. A transaction control step that is configured with the Abort option must always follow an EAI step. It cannot follow any other type of step.

  The **getTransControlType** method returns **swNA** if the step is not a transaction control step.
- **SWTransControlType** - This new enumeration has been added to support transaction control steps. It contains the following constants:

Constant	Description	Value
swNA	Not applicable for this step type.	‘N’
swAbort	Abort transaction control step.	‘A’
swContinue	Commit and continue transaction control step.	‘C’
swConcede	Commit and concede transaction control step.	‘D’

This new enumeration is returned by the **getTransControlType** method on **SWStep**.

- **getRetryDelay** - This new method on **SWStep** returns the number of minutes in which a transaction will be retried if it fails. This value, which is only applicable for **swContinue**-type transaction control steps, is specified when the transaction control step is defined with the SPD. A failed transaction will be retried one time.
- **SWAuditActionType** - The following new constants have been added to this enumeration to support transaction control steps:

Constant	Description	Value
swTransProcessed	Transaction Control Step Processed	54
swTransStarted	Transaction Control Step - New Transaction Started	55
swTransRestart	Transaction Control Step - Retry Time Expired	56
swTransAborted	Transaction Control Step Processed - Transaction Aborted	87

These new constants are added to the case's audit trail (**SWAuditStep->getAction**) when an action is performed against a transaction control step.

3.2.6 Key for SWFwdItem Object Modified (CR 12885)

The key for the **SWFwdItem** object contained a hard-coded "R", indicating that the work item could only be forwarded to released work queues. The key has been modified to include either an "R" or "T" to indicate a released or test work queue, as follows:

```
Qname@HostingNode|QueueType
```

where QueueType = "R" for a released queue, or "T" for a test queue.

3.2.7 New Methods Allow you to Control CDQP Fields Returned from Server (CR 12550)

The following methods have been added to control which Case Data Queue Parameters (CDQPs) to return from the server with work items that reside in an **SWXList**. This allows you to more closely control resources by returning only the CDQP fields that are needed.

- **setCDQPNames** - This new method on **SWCriteriaWI** allows you to specify the field names of Case Data Queue Parameters (**SWCaseDataQParam** objects) that will be returned by the **getCaseDataQParams** method for all **SWWorkItem** objects residing in an **SWXList**. This provides a means of resource control by allowing you to specify which CDQPs to return from the server.
- **getCDQPNames** - This new method on **SWCriteriaWI** returns the names of the fields specified with the **setCDQPNames** method.

3.2.8 New Methods Added to Create Work Items and Specify CDQPs and Case Fields (CR 10642)

The following methods have been added to allow you to create work items and be able to specify which Case Data Queue Parameters (CDQPs) and case fields to return from the server with the work items that are created:

- **makeWorkItemEx** - This new method on **SWNode** extends the **makeWorkItem** method by providing *CDQPNames* and *CaseFieldNames* parameters to specify the CDQPs and case fields to return from the server.

- **makeWorkItemByTagEx** - This new method on **SWNode** extends the **makeWorkItemByTag** method by providing *CDQPNames* and *CaseFieldNames* parameters to specify the CDQPs and case fields to return from the server.
- **makeXListItemsEx** - This new method on **SWWorkQ** extends the **makeXListItems** method by providing *CDQPNames* and *CaseFieldNames* parameters to specify the CDQPs and case fields to return from the server.

All three of these new methods allow you to more closely control resources by specifying only the CDQPs and/or case fields that you need. See the on-line help for specifics about the valid entries for the *CDQPNames* and *CaseFieldNames* parameters.

3.3 Version 10.0(4.2)

3.3.1 SWNode and SWNodeInfo Keys Modified (CR 15469)

All **SWNode** and **SWNodeInfo** keys will now take the form:

```
ComputerName|NodeName|IsDirector|InstanceNumber
```

where:

- *IsDirector* indicates if the node is an SPO Director ("Y") or a TIBCO SPO Server ("N").
- *InstanceNumber* indicates the instance number of the TIBCO SPO Server. This will default to "1" for non-multiple instance servers.

The **itemByKey**, **login**, **logout**, **disconnect**, **makeViewItems**, and **makeViewCases** methods will automatically add the default values for the *IsDirector* ("N") and *InstanceNumber* ("1") components if they are not provided in the *NodeKey* parameter. This provides backward compatibility for applications that use hard-coded keys.

The **addNode** and **makeNodeInfo** methods will automatically add the default *IsDirector* (“N”) component to the *SWNodeInfo* key if it is not provided in the method call. They also always add the default *InstanceNumber* (“1”) component to the *SWNodeInfo* key. These methods always assume instance 1.

3.3.2 Enumeration Constants Added to SWAuditActionType (CR 15247)

The following constants have been added to the **SWAuditActionType** enumerations:

Constant	Description	Value
swReleasedNoAddressees	The step was released, although there is no addressee defined for the step.	39
swReleasedNoSubProcs	The step was released, although there are no sub-procedures defined for the step (applicable only for dynamic sub-procedure call steps and graft steps).	40

3.3.3 Allow Logging Into Multiple Instances of the TIBCO SPO Server (CR 14985)

The following changes have been made to the TIBCO SPO C++ Client to allow logging into multiple instances of the TIBCO SPO Server:

- **getInstanceNumber** - This new method on **SWNodeInfo** returns the instance number of the TIBCO SPO Server represented by the **SWNodeInfo** object. This is only applicable to TIBCO SPO Servers that support multiple instances. For TIBCO SPO Servers that are not multiple-instance capable, this method returns 1.
- **addNode**- This method has a new *InstanceNumber* parameter that is used to indicate a specific server instance to which you are directing the UDP message when the TIBCO SPO Server supports multiple instances.
- **makeNodeInfo** - This method has a new *InstanceNumber* parameter that is used to indicate a specific server instance for which the **SWNodeInfo** object being created is to represent.
- **login** - This method has been modified to log the user into a specific instance of the TIBCO SPO Server if the *NodeKey* passed in the method call includes an instance number. If the *NodeKey* does not include an instance number, the login method automatically adds instance "1" to the key. This allows applications that use the old key format (prior to the instance number being added) to access instance "1" of a server that has the new key format without modifying code.
- **getKey** - When called from **SWNode** and **SWNodeInfo**, the key returned by this method now includes the instance number for TIBCO SPO Servers that support multiple instances. It also includes an instance number when called from **SWNodeInfo** if the **SWNodeInfo** object was created with the optional *InstanceNumber* parameter provided on the **addNode** and **makeNodeInfo** methods.
- **itemByKey** - This method has been modified to accommodate SPO applications that have hard-coded keys that do not include an instance number. The **itemByKey** method will add instance "1" to the key and re-attempt to return the item if it fails to find an item without an instance number. This allows applications that use the old key format to access instance "1" of a server that has the new key format without modifying code.
- **getTag** - This method on **SWNodeInfo** now returns a tag that includes the instance number of the TIBCO SPO Server. The instance number defaults to “1” in the tag if the TIBCO SPO Server does not support multiple instances.

  The tag for an **SWNodeInfo** object has also been modified to always include the *IsDirector* flag: “Y” if the **SWNodeInfo** object represents an SPO Director; “N” if the **SWNodeInfo** object represents a TIBCO SPO Server. (Previously, the *IsDirector* flag was blank if the **SWNodeInfo** object represented a TIBCO SPO Server.)

3.3.4 Ability to Limit Number of Cases Retrieved from TIBCO SPO Server Added (CR 14985)

Methods have been added to allow you to limit the number of cases that are retrieved from the TIBCO SPO Server when using **SWXLists**. The following methods have been added to support this new functionality:

- **setMaxCnt** - The new method on **SWCriteriaC** allows you to specify the maximum number of cases to retrieve from the TIBCO SPO Server and place in the raw data buffers on the SPO client. Setting this value allows you to prevent the wait that results from retrieving all cases when there are a very large number of cases.

- **getMaxCnt** - This new method on **SWCriteriaC** indicates the number of cases that will be retrieved from the TIBCO SPO Server and placed in the raw data buffers on the SPO client. This value is set with the **setMaxCnt** method. Note that the MaxCnt is initialized to –1, which causes all of the requested cases (that satisfy the filter expression, if specified) to be retrieved from the server.
- **getOverMaxCnt** - This new method on **SWCriteriaC** indicates the number of cases excluded from the **SWXList** because they are in excess of the maximum requested number of items (specified with the **setMaxCnt** method).

3.3.5 Methods Added to Obtain Case Start Date and Time (CR 14770)

The following methods have been added to the **SWCase** object so that you can obtain the date and time the case was started:

- **getTimeStarted** - Returns the date and time the case was started.
- **getTimeStartedOffset** - Returns the additional microseconds to calculate the exact time the case was started. For example, if the date/time returned by the **getTimeStarted** method is October 4, 2001 09:13:40 (the time format is HH:MM:SS) and the **getTimeStartedOffset** method returns 500, the case was started at 09:13:40 and 500 microseconds.

3.3.6 Method Added to Perform MOVESYSINFO Function (CR 14619)

The **moveSysInfo** method has been added to the **SWNode** object to allow the client application to explicitly call the **MOVESYSINFO** function. Prior to this new method, the TIBCO SPO Server called the **MOVESYSINFO** function whenever an administrative function was performed, i.e., any function that affected a user, group, role, attribute, or queue supervisor definition. This can tie up the background and WIS/WQS processes for long periods of time if there are lots of users.

This new method is used in conjunction with the **ImplicitMoveSysInfo** TIBCO SPO Server configuration parameter to control when the **MOVESYSINFO** function occurs. See your TIBCO SPO Server readme for information about this configuration parameter.

3.3.7 New Method Added to Set Default Criteria (CR 14585)

The **setDefCriteriaEx** method has been added to the **SWWorkQ** object. This method allows you to set the default filter and sort criteria for the work queue. This causes the criteria you pass in as a parameter to this method to persist on the current instance of the work queue, causing future **SWViews** or **SWXLists** of work items on that instance of the queue to use the default criteria.

Note that the existing **setDefCriteria** method can also still be used to establish default criteria; it sets the default criteria based on the current filter expression and sort fields on the **SWView** that is returned by the **getWorkItems** method on the work queue. The new **setDefCriteriaEx** method is probably a better choice to set default criteria if you are using **SWXLists**.

3.3.8 New TIBCO SPO Server Error for Buffer Overflow (CR 14158)

The following new error is returned by the TIBCO SPO Server if an attempt is made to send a message larger than the value specified for the TIBCO SPO Server message response buffer (which is defined by the **TCPReponsePages** TIBCO SPO Server configuration parameter — the default is 4K bytes).

Dec.	Hex.	Constant	Description
223	0x00DF	ER_BUFFOVERRUN	Internal SPO buffer too small.

3.3.9 Client Log Default Categories Changed (CR 14111)

If the SPO Client Log debug log level (**swLogDebug**) was selected, all log categories were written to the client log by default, including object constructor and destructor information. This could result in a very large log file in a short period of time. Because of this, the default log categories have been changed for the debug log level to include all categories except **swCatConstDestr** (object constructors and destructors).

If you need object constructor/destructor information in the log, you can specify swCatAll using the **setCategories** method on **SWLog**, or use the **enableCategory** method to enable the **swCatConstDestr** category.

3.3.10 Methods Added to List Supervised Work Queues (CR 13785)

The current **getAWorkQs** method on **SWAdmin** returns a list of *all* administrative work queues if the user calling the method is an “Admin” user (the user’s MENUNAME attribute = ADMIN). If the user is a non-Admin user, **getAWorkQs** returns only the administrative work queues the user is authorized to supervise (for the purpose of defining participation and redirection schedules). This complete list of supervised work queues is needed by Admin users so they can add supervisors to or remove supervisors from a work queue. Therefore, the functionality of **getAWorkQs** will not change. There is, however, a need for Admin users to get a list that contains only the work queues they are authorized to supervise. The following new method has been added to **SWAdmin** to allow this:

- **getSupervisedAWorkQs** - This returns an **SWList** of **SWAWorkQ** objects, one for each administrative work queue the user is authorized to supervise. For non-Admin users, this new method returns the same list of work queues as the **getAWorkQs** method.

To provide naming consistency, the **getAdminQNames** method on **SWUser** has been deprecated and replaced by the following method:

- **getSupervisedAWorkQNames** - This returns an **SWList** of strings, one for each work queue the user is authorized to supervise.

The **getAdminQNames** method still exists, for backward compatibility.

3.3.11 Sub-Procedure Precedence Added to SPO (CR 13783)

There is a need to be able to specify which version (released, unreleased, or model) of a sub-procedure is looked for first, second, or third when a sub-procedure is started from the main procedure.

A new *SubProcPrecedence* parameter has been added to the **startCaseEx** method. This parameter specifies the order in which sub-procedure versions that are launched from the main procedure are looked for. It allows you to specify that certain sub-procedure versions are looked for first, second, or third. The default is to start only released procedures. (This new parameter is also added to the **startCase** method, which is a deprecated method. This provides backward compatibility.)

The *SubProcPrecedence* parameter accepts an ASCII value that is defined in the new constant enumeration, **SWSubProcPrecedenceType**. This new constant enumeration defines the following values:

Constant	Description	Value
swPrecedenceR	Released only	‘0’
swPrecedenceUR	Unreleased > Released	‘1’
swPrecedenceMR	Model > Released	‘2’
swPrecedenceUMR	Unreleased > Model > Released	‘3’

Constant	Description	Value
swPrecedenceMUR	Model > Unreleased > Released	‘4’

For example, specifying **swPrecedenceUR** causes the engine to first look for an unreleased version of the sub-procedure, then a released version. If neither is found, the error message “Sub-case started of a procedure that isn’t a sub-procedure” is written to the **sw_warn** file.

3.3.12 New entry added to SWProcStatusType (CR 13768)

The following new enumeration has been added to **SWProcStatusType** to represent a procedure that is incomplete and has been withdrawn:

Constant	Description	Value
swWithdrawnIncomplete	Procedure is incomplete and has been withdrawn.	‘T’

3.3.13 Ability to Identify all Outstanding Steps Added (CR 13678)

The ability to identify outstanding steps for each type of step in a case family, as well as obtain the path to those outstanding steps, has been added to SPO. This ability is needed by certain functions. For instance, the **jumpTo** method requires that you provide a list of the outstanding steps you would like to withdraw. Other methods, such as **setState** and **triggerEvent**, require that you provide a complete path to outstanding sub-procedures if you are updating case data for fields that are defined in those sub-procedures.

You can now determine which normal, event, EAI, sub-procedure call, dynamic sub-procedure call, and graft steps are currently outstanding in the case. You can also determine the complete path, from the main case, to each outstanding step or sub-procedure. To provide this ability, the following additions and changes have been made to the TIBCO SPO C++ Client:

The **SWCase** object has been updated as follows:

- **getDynamicSubProcSteps** - This new method returns an **SWList** of **SWDynamicSubProcStep** objects, one for each dynamic sub-procedure call step that is currently outstanding in the case family.
- **getEAISteps** - This new method returns an **SWList** of **SWEAIStep** objects, one for each EAI step that is currently outstanding in the case family.
- **getEventSteps** - This new method returns an **SWList** of **SWEventStep** objects, one for each event step that is currently outstanding in the case family.
- **getOutstandingItems** - This method has been modified to now return only outstanding normal steps, i.e., it returns an **SWList** of **SWOutstandingItem** objects, one for each normal step that is currently outstanding in the case family.
- **isRecurseGrafts** - This method has been deprecated. The **isRecurseProcPath** method (see below) should now be used for all step types. This method will continue to be supported to provide backward compatibility.
- **isRecurseProcPath** - This new method specifies whether or not the "outstanding step" methods (**getOutstandingItems**, **getSubProcSteps**, **getDynamicSubProcSteps**, **getEAISteps**, **getEventSteps**, and **getGraftSteps**) return a recursive list of outstanding steps, i.e., whether the list also includes outstanding steps from sub-procedures that have been launched from the main case.

The following new objects have been added:

- **SWDynamicSubProcStep** - This new object represents an outstanding dynamic sub-procedure call step. A list of these objects is returned by the **getDynamicSubProcSteps** method on **SWCase**. This object contains the following methods:
  - **getArrived** - Returns the date and time the step became outstanding.
  - **getCaseNumber** - Identifies the case containing the outstanding step.
  - **getClassId** - Identifies the object class.
  - **getDeadline** - Returns the date and time of a deadline on the step, if set.
  - **getKey** - Returns the key for the step: ProcName|StepName|CaseNumber
  - **getProcMajorVer** - Returns the *MajorVersion#* portion of the procedure's version number.
  - **getProcMinorVer** - Returns the *MinorVersion#* portion of the procedure's version number.
  - **getProcName** - Returns the name of the procedure with which the step is associated.
  - **getProcPath** - Provides the complete path to the outstanding dynamic sub-procedure call step. This can be used in the *WithdrawList* parameter in the **jumpTo** method.
  - **getStepName** - Returns the name of the dynamic sub-procedure call step.
  - **getSubProcSteps** - Returns an **SWLocList** of **SWSubProcStep** objects, one for each sub-procedure that was started by the dynamic sub-procedure call step. This returns *all* sub-procedures that were started by the dynamic sub-procedure call step, whether they have completed or not. You can determine which of the sub-procedures are still outstanding (have not completed) by calling the **isOutstanding** method on the **SWSubProcStep** object that represents the sub-procedure you are interested in.
- **SWEAIStep** - This new object represents an outstanding EAI step. A list of these objects is returned by the **getEAISteps** method on **SWCase**. This object contains the following methods:
  - **getArrived** - Returns the date and time the step became outstanding.
  - **getCaseNumber** - Identifies the case containing the outstanding step.
  - **getClassId** - Identifies the object class.
  - **getDeadline** - Returns the date and time of a deadline on the step, if set.
  - **getExternalId** - Returns a string that uniquely identifies an outstanding EAI step (also called an "external work item"). This is the unique identifier that is passed to the third-party application when an external work item is passed to the application. This allows the third-party application to identify the external work item when it passes it back to TIBCO after processing.
  - **getKey** - Returns the key for the step: ProcName|StepName|CaseNumber
  - **getProcMajorVer** - Returns the *MajorVersion#* portion of the procedure's version number.
  - **getProcMinorVer** - Returns the *MinorVersion#* portion of the procedure's version number.
  - **getProcName** - Returns the name of the procedure with which the step is associated.
  - **getProcPath** - Provides the complete path to the outstanding EAI step. This can be used in the *WithdrawList* parameter in the **jumpTo** method.
  - **getStepName** - Returns the name of the EAI step.
- **SWEventStep** - This new object represents an outstanding Event step. A list of these objects is returned by the **getEventSteps** method on **SWCase**. This object contains the following methods:
  - **getArrived** - Returns the date and time the step became outstanding.
  - **getCaseNumber** - Identifies the case containing the outstanding step.
  - **getClassId** - Identifies the object class.

  - **getDeadline** - Returns the date and time of a deadline on the step, if set.
  - **getKey** - Returns the key for the step: ProcName|StepName|CaseNumber.
  - **getProcMajorVer** - Returns the *MajorVersion#* portion of the procedure's version number.
  - **getProcMinorVer** - Returns the *MinorVersion#* portion of the procedure's version number.
  - **getProcName** - Returns the name of the procedure with which the step is associated.
  - **getProcPath** - Provides the complete path to the outstanding Event step. This can be used in the *WithdrawList* parameter in the **jumpTo** method.
  - **getStepName** - Returns the name of the Event step.
- **SWExtProcess** - This new object represents an outstanding external process that was started by a graft step. A list of these objects is returned by the **getExtProcesses** method on **SWGraftStep**. This object contains the following methods:
  - **getClassId** - Identifies the object class.
  - **isOutstanding** - Identifies whether or not the external process has completed. This will return True until the application informs the engine that the external process has completed by calling **graftExtProcessComp**, at which point this flag is set to False.
  - **getKey** - Returns the key for the external process object: ExtProcessName
  - **getName** - Returns the name of the external process.
  - **getReturnStatus** - Returns a status that is specified by the user when the external process is initiated with the **startGraftTask** method, then again when the external process is flagged as complete with the **graftExtProcessComp** method.
  - **getStartIndex** - Returns a zero-based index that indicates the sequential order in which the engine started the external process that is represented by the **SWExtProcess** object.

The following new methods were added to the **SWOutstandingItem** object:

- **getArrived** - Returns the date and time the normal step became outstanding, i.e., the date and time the work item arrived in the work queue.
- **getDeadline** - Returns the date and time of a deadline on the step, if set.
- **getProcMajorVer** - Returns the *MajorVersion#* portion of the procedure's version number.
- **getProcMinorVer** - Returns the *MinorVersion#* portion of the procedure's version number.

The following changes were made to the **SWSubProcStep** object:

- **getArrived** - This new method returns the date and time the sub-procedure call step became outstanding.
- **getDeadline** - This new method returns the date and time of a deadline on the step, if set.
- **isOutstanding** - This new method functions as follows: If the **SWSubProcStep** represents a sub-procedure call step, this method will always return True. If the **SWSubProcStep** object represents a sub-procedure that was started by either a dynamic sub-procedure call step or graft step, this will return True if the sub-procedure is still outstanding (has not completed yet). (The engine will keep track of when sub-procedures complete, and will set this flag accordingly.)
- **getKey** - This method has been modified because the key returned was not sufficiently unique to identify a sub-procedure that was launched from a dynamic sub-procedure call step or graft step. It has been expanded to include the case number of the sub-case started by the dynamic sub-procedure call step or graft step, as follows:

  StepName|SubCaseNumber

- **getReturnStatus** - This new method returns a system-generated status that indicates the current status for the sub-procedure that is represented by the **SWSubProcStep** object. These are enumerated in **SWSubProcStatusType**.
- **getStartIndex** - This new method returns a zero-based index that indicates the sequential order in which the engine started the sub-procedure that is represented by the **SWSubProcStep** object. This index is only applicable to sub-procedures that are started by dynamic sub-procedure call steps and graft steps. If the **SWSubProcStep** object represents a sub-procedure call step, this method returns -1.
- **getSubProcMajorVer** - This new method returns the *MajorVersion#* portion of the sub-procedure's version number.
- **getSubProcMinorVer** - This new method returns the *MinorVersion#* portion of the sub-procedure's version number.
- **getSubProcPath** - This is a new method. If the **SWSubProcStep** object represents an outstanding sub-procedure call step, this method returns the path to the sub-procedure call step. If the **SWSubProcStep** object represents a sub-procedure that was started by a dynamic sub-procedure call step or graft step, this method returns the path to the sub-procedure itself.

The following changes have been made to the **SWGraftStep** object:

- **getArrived** - This new method returns the date and time the graft step became outstanding. Graft steps are considered outstanding when they are initiated by calling the **startGraftTask** or **setGraftTaskCnt** methods, or when the process flow reaches the graft step.
- **getExtProcessNames** - This method has been deprecated — the **getExtProcesses** method (see below) should be used instead. This method will continue to be supported to provide backward compatibility.
- **getExtProcesses** - This new method returns an **SWLocList** of **SWExtProcess** objects, one for each external process that has been initiated by the graft step. This returns *all* external processes initiated by the step, whether they have completed or not. You can determine which of the external processes are still outstanding (have not completed) by calling the **isOutstanding** method on the **SWExtProcess** object that represents the external process you are interested in.
- **getProcMajorVer** - This new method returns the *MajorVersion#* portion of the procedure's version number.
- **getProcMinorVer** - This new method returns the *MinorVersion#* portion of the procedure's version number.
- **getSubProcSteps** - This method returns an **SWLocList** of **SWSubProcStep** objects, one for each sub-procedure that was started by the graft step. This returns *all* sub-procedures that were started by the graft step, whether they have completed or not. You can determine which of the sub-procedures are still outstanding (have not completed) by calling the **isOutstanding** method on the **SWSubProcStep** object that represents the sub-procedure you are interested in.

3.4 Version 10.0(0.1)

3.4.1 New FieldType Argument on startCaseEx and simulateCase Methods (CR 12986)

Because of the addition of array fields (see section 3.4.3), a new *FieldType* argument needed to be added to some of the method signatures for **startCaseEx** and **simulateCase**. This results in an interface change. See the on-line help system for information about the specific signatures that now have a *FieldType* argument.

3.4.2 New Server Log Categories Added (CR 12608)

The following new log categories have been added to the **SWSrvLogCategoryType** enumeration.

Constant	Description	Hex Value
swCatSALTiming	SAL Timing	0x04000000
swCatDirector	Director Operations	0x08000000

3.4.3 Array Fields, Ad-Hoc Processing and Procedure Version Control Added (CR 12590)

The following new functionality has been added to the SPO Client:

- **Array Fields** - Array fields are defined using the SPD's Field Definition dialog in the same way as a standard single-instance field. An option on the Field Definition dialog allows you to designate the field as a single-instance field or an array field. If designated as an array field, the field can hold up to 99,999 data elements, each identified by an index (the field name followed by an index number in brackets "[ ]"). Array fields are used with ad-hoc processing (see below).
- **Ad-Hoc Processing** - This new functionality involves the addition of two new step types: **Dynamic Sub-Procedure Call Step** and **Graft Step**. These steps provide the ability to specify *at run-time* (rather than at procedure design-time) the number of sub-procedures and/or external processes that will be started by the step.
- **Procedure Version Control** - This new functionality provides the ability to create and track multiple versions of a TIBCO procedure.

The objects and methods that were added and/or modified in the SPO client to support this new functionality are listed below. For more detailed information about these subjects, see the *Ad-Hoc Processing* and *Procedure Version Control* topics in the SPO client on-line help.

Array Fields

The **SWField**, **SWMarking**, **SWFMarking** objects have the following new method to indicate whether the field is a single-instance field or array field:

- **isArrayField** - If set to True, the field/marking is an “array field”, which has up to 99,999 elements accessible by index. See the *Array Fields* topic in the SPO Client on-line help for information about how array fields are used in ad-hoc processing.

Ad-Hoc Processing

The following objects have been created/updated to support ad-hoc processing:

- **SWStepType** - The following constants have been added to this enumeration:

Constant	Description	Value
swDynamicSubProcCall	Dynamic Sub-Procedure Call Step	‘D’
swGraft	Graft Step	‘G’

- **SWGraftStep** - This new object represents an outstanding or initiated graft step in a live case. It contains the following methods:
  - **getCaseNumber** - Identifies the case to which the graft step belongs.
  - **getClassId** - Identifies the object class.

  - **getDeadline** - The date and time of deadline, if set.
  - **getExtProcessNames** - The names of the external processes that are outstanding on this graft step. Names are added to this list when external processes are started with the **startGraftTask** method. Names are removed from this list when the **graftExtProcessComp** and **deleteGraftTask** methods are called.
  - **getGraftId** - A unique ID that identifies the graft step.
  - **getKey** -The key for the graft step: ProcName|StepName|CaseNumber
  - **getProcName** - The name of the procedure containing the graft step.
  - **getProcPath** - The name of the graft step. If the graft step is in a sub-procedure, this provides the path to the step in the form: "sub1|stepname", where "sub1" is the name of the step that calls the sub-procedure in which the graft step resides.
  - **getStepName** - The name of the graft step.
  - **getTaskCnt** - The number of tasks that must be completed for the graft step to be released. This is set with the **setGraftTaskCnt** method on **SWCase**.
  - **isGraftOutstanding** - True if the graft step is currently the outstanding step.
  - **isGraftWithdrawn** - True if the graft step has been withdrawn.
  - **isTaskCntSet** - True if the task count has been set with the **setGraftTaskCnt** method.
- **SWCase** - The following methods have been added to this object:
  - **deleteGraftTask** - Decrements the task count by one.
  - **getGraftSteps** - Returns a list of **SWGraftStep** objects, one for each outstanding or initiated graft step in the case.
  - **graftExtProcessComp** - Informs the TIBCO Staffware iProcess Engine that an external process that was started with the **startGraftTask** method has completed.
  - **isRecurseGrafts** - If set to True, the list returned by **getGraftSteps** will include recursive graft steps.
  - **setGraftTaskCnt** - Increments the task count to inform the TIBCO Staffware iProcess Engine how many tasks must be completed before releasing the graft step.
  - **startGraftTask** - Initiates a task by specifying the sub-procedures and/or external processes to start.
- **SWStep** - The following methods have been added to this object:
  - **getSubProcStatus** - Returns a numeric array field that contains return statuses for each of the sub-procedures started from the dynamic sub-procedure call step or graft step. The return statuses are defined in the new **SWSubProcStatusType** enumeration.
  - **isHaltOnSubProc** - If set to True, process flow is halted if the step attempts to start a non-existent sub-procedure. (This flag is set in the step definition in the SPD.)
  - **isHaltOnTemplate** - If set to True, process flow is halted if different parameter templates are used when starting multiple sub-procedures. (This flag is set in the step definition in the SPD.)
  - **isHaltOnTemplateVer** - If set to True, process flow is halted if different versions of parameter templates are used when starting multiple sub-procedures. (This flag is set in the step definition in the SPD.)
  - **isKeepOnWithdraw** - If set to True, the work item is not deleted from the work queue on withdrawal (i.e., it is "kept" in the work queue). (This flag is set in the step definition in the SPD.)

- **SWWorkItem** - The following method has been added to this object:
  - **isKeepOnWithdraw** - If set to True, the work item is not deleted from the work queue on withdrawal (i.e., it is “kept” in the work queue). (This flag is set in the step definition in the SPD.)
- **SWSubProcStatusType** - This new enumeration, which identifies the sub-procedure return statuses, contains the following constants:

Constant	Description	Value
swAttempt	Sub-case start has not been attempted.	0
swStarted	Sub-case was started successfully.	1
swCompleted	Sub-case completed successfully.	2
swErrSubProc	Error starting the sub-case.	-1
swErrTemplate	Error starting the sub-case because different parameter templates were used.	-2
swErrInTemplateVer	Error starting sub-case because different versions of parameter template were used.	-3
swErrOutTemplateVer	Error completing sub-case because different versions of parameter template were used.	-4

- **SWAuditActionType** - The following constants have been added to this enumeration to identify the actions performed on dynamic sub-procedure call steps and graft steps:

Constant	Description	Value
swDynaGraftCaseStart	Case started for sub-case of dynamic or graft step.	25
swTaskCountSet	Task count received.	26
swTaskDeleted	Graft task deleted.	27
swSubCaseGrafted	Sub-case grafted.	28
swExtProcessGrafted	External process grafted.	29
swGraftInitiated	Graft task initiated.	30
swExtProcessReleased	External process released.	31
swGraftReleased	Graft step released.	32
swDynamicReleased	Dynamic sub-procedure call step released.	33
swGraftWithdrawn	Graft step withdrawn.	35
swDynaGraftDeadlineExp	Deadline Expired for Dynamic or Graft Step.	36
swDynamicWithdrawn	Dynamic sub-procedure.	37
swKeepOnWithdraw	Step withdrawn and kept in work queue.	38
swErrBadSubProc	Invalid sub-procedure error.	84
swErrDiffTemplate	Different parameter template error.	85
swErrDiffTemplateVer	Different parameter template version error.	86

- The following TIBCO SPO Server errors have been added.

Dec.	Hex.	Constant	Description
220	0x00DC	ER_INVALGRAFTID	Invalid Graft Id.
221	0x00DD	ER_EXTPROLEN	External process name exceeds 30 characters.

Dec.	Hex.	Constant	Description
222	0x00DE	ER_PROCCASE	The SAL returned ER_PROC. There is a problem with the procedure or the case may be invalid.

Procedure Version Control

The following objects have been created/updated to support procedure version control:

- **SWNode** - This object has the following new/modified methods:
  - **getProcGroups** - New method. Returns a list of **SWProcGroup** objects (see the next bullet item), one for each procedure defined on the node.
  - **makeProcByStatus** - New method. Allows you to make an **SWProc** object that has a specified status: **swReleased**, **swUnreleased**, or **swModel**.
  - **makeProc** - This method has been modified to allow you to specify a version number in the parameters so that an **SWProc** object can be created for a specific procedure version.
  - **makeStep** - This method has been modified to allow you to specify a version number in the parameters so that an **SWStep** object can be created for a specific procedure version.
- **SWProcGroup** - This new object provides access to all versions of a particular procedure. It has the following methods:
  - **getClassId** - Identifies the object class.
  - **getHostingNode** - The name of the node that hosts this procedure.
  - **getKey** - The key used to access a specific **SWProcGroup** object: HostingNode|Name
  - **getName** - The name of the procedure.
  - **getProcVersions** - This returns a list of **SWProc** objects, one for each version of the procedure defined on the node.
- **SWProcStatusType** - This enumeration has the following new constant.

Constant	Description	Value
swModel	Procedure that has been imported.	‘M’

- **SWProc** - This object has the following new/modified methods:
  - **getDateCreated** - New method. Date and time this version of the procedure was created.
  - **getDateModified** - New method. Date and time the procedure was last modified.
  - **getDateReleased** - New method. Date and time the procedure was released.
  - **getDateWithdrawn** - New method. Date and time the procedure was withdrawn.
  - **getLastUpdateUser** - New method. Name of the user who last updated the procedure.
  - **getProcAudits** - New method. This returns a list of **SWProcAudit** objects (see the next bullet item), one for each modification that has been made to the procedure.
  - **getProcMajorVer** - New method. The *MajorVersion#* portion of the procedure’s version number.
  - **getProcMinorVer** - New method. The *MinorVersion#* portion of the procedure’s version number.
  - **getVersionComment** - New method. Comment that was entered when this version of the procedure was created.

  - **isWithAuditData** - New method. If set to True, procedure audit data will be returned from the server with the procedure. The audit data is available with the **getProcAudits** method. This flag is set with **setWithAuditData**.
  - **setWithAuditData** - New method. Sets the **isWithAuditData** flag to specify whether or not procedure audit data is returned from the server.
  - **getTag** - The tag that is returned by this method has been modified to include the major and minor portions of the procedure's version number so that it can be used with the **makeProcByTag** method to make a specific version of the procedure. (Only applicable if using a TIBCO SPO Server that supports procedure version control.)
  - **getKey** - The value returned by this method is extended to include the major and minor portions of the procedure version number. This new key format must be used when invoking the **itemByKey** method to extract an **SWProc** object from the **getProcs**, **getAuditProcs**, **getStartProcs** and **getProcVersions** lists. (Only applicable if using a TIBCO SPO Server that supports procedure version control.)

    Note - SPO applications that have procedure keys "hard coded" must be modified to include the procedure version component (ProcMajorVer|ProcMinorVer) if both the client and TIBCO SPO Server support procedure version control. See "Procedure Key must include Version Number" on page 22 for more information.

- **SWProcAudit** - This new object represents a modification to a procedure. It contains the following methods:
  - **getAction** - The action that was performed on the procedure. These are defined in the new **SWProcAuditActionType** enumeration.
  - **getClassId** - Identifies the object class.
  - **getComment** - Describes the modification made to the procedure.
  - **getDate** - Date and time the modification was made.
  - **getKey** - The key to the **SWProcAudit** object. The key is an integer that is an index into the list of **SWProcAudit** objects returned by **getProcAudits**.
  - **getProcMajorVer** - The *MajorVersion#* portion of the procedure's version number.
  - **getProcMinorVer** - The *MinorVersion#* portion of the procedure's version number.
  - **getUser** - The user who made the modification to the procedure.
- **SWAuditActionType** - The following constant has been added to this enumeration to identify cases that are migrated to a new procedure version:

Constant	Description	Value
swCaseMigrated	Case migrated to new procedure version.	34

- **SWProcAuditActionType** - This new enumeration describes actions performed when modifying a procedure definition. It contains the following constants:

Constant	Description	Value
swProcCreated	Procedure created.	'C'
swProcComment	Procedure comment modified.	'M'
swProcImported	Procedure imported.	'I'
swProcUpdated	Procedure updated.	'U'
swProcReleased	Procedure released.	'R'
swProcWithdrawn	Procedure withdrawn.	'W'

- **SWCase** - This object has the following new methods, which allow you to determine the version of procedure the case is from.
    - **getProcMajorVer** - The *MajorVersion#* portion of the procedure's version number.
    - **getProcMinorVer** - The *MinorVersion#* portion of the procedure's version number.
- **SWAuditStep** - This object has the following new methods, which allow you to determine the version of procedure when the action in the audit step took place.
    - **getProcMajorVer** - The *MajorVersion#* portion of the procedure's version number.
    - **getProcMinorVer** - The *MinorVersion#* portion of the procedure's version number.
- **SWStep** - The following method has been modified on this object:
    - **getTag** - This method has been modified to include the major and minor portions of the procedure's version number so that is can be used with the **makeStepByTag** method to make a step from a specific version of the procedure. (Only applicable if using a TIBCO SPO Server that supports procedure version control.)
- The following SPO Client error has been added to support procedure version control (see section 5.3 for more information).

Constant	Dec.	Hex.	Description
E_InvalidKeyErr	2077	0x81D	"Invalid Key."

3.4.4 Method Added to get Script Contents (CR 12446)

The **getScript** method has been added to the **SWStep** object. This new method provides access to the script that is defined in a step of type **swScript**.

3.4.5 Methods Added to Discard Local Blocks of Items (CR 12427)

The **isKeepLocalItems** and **setKeepLocalItems** methods have been added to **SWXList** to allow you to control whether or not more than one block of items will be held locally after they have been retrieved from the TIBCO SPO Server. If this flag is set to True, multiple blocks can be held locally. If set to False (the default), when another block is sent from the server, the previous block is automatically cleared, thereby minimizing the use of local memory.

3.4.6 Method Added to get Audit Trail Message (CR 12411)

The **getMessage** method has been added to the **SWAuditStep** object. This new method provides access to the actual message that is written to the audit trail for that particular audit step.

3.4.7 New Counts for Work Items on an SWXList (CR 12410)

The following methods have been added to the **SWCriteriaWI** object to provide counts for work items on **SWXLists**:

- **getDeadlineCnt** - Returns the number of items with deadlines on the XList.
- **getUnopenedCnt** - Returns the number of new (i.e., unopened) work items on the XList.
- **getUrgentCnt** - Returns number of urgent work items on the work queue or in the XList.

3.5 New Features in Previous Versions

For a description of the new features that were added in each release of the TIBCO SPO C++ Client for HP-UX prior to Version 10.x, see the **ChangeHistory.pdf** file on the distribution CD.

4 Restrictions

Note - Where a Change Request (CR) has been raised in connection with a particular restriction, the CR number (CR nnnn) is shown in brackets.

None.

5 Known Issues

5.1 The Month Field in ‘struct tm’ is Incorrectly Represented

The month field (tm_mon) in a ‘**struct tm**’ is supposed to contain the values 0 (for January) through 11 (for December). The TIBCO SPO C++ Clients, however, are using the values 1 through 12 to represent months.

Starting with version 9.0(0.2) of the TIBCO SPO C++ Client, the month values will be correctly represented by the values 0 through 11, instead of the previously used values 1 through 12.

If you have externally stored tm_mon values from pre-version 9.0(0.2) TIBCO SPO C++ Clients, then later import them into a version 9.0(0.2) or later TIBCO SPO C++ Client, the month values must be adjusted before they are imported.

5.2 License Count may be a Negative Number

If the number of user licenses exceeds 32767, the **getLogonLicenseCnt** and **getLogonsAvailableCnt** methods on the **SWNodeInfoEx** object incorrectly return negative values. The workaround for this is to add 65536 to the negative value to obtain the correct license count value.

Note that the **getLogonLicenseCnt** and **getLogonsAvailableCnt** methods are not applicable if your TIBCO Staffware iProcess Engine includes CR 16592, which eliminates the requirement for user licenses. These methods always return -1 if your engine includes CR 16592.

5.3 Procedure Key must include Version Number

When procedure version control was added to the SPO client in version 10.0(0.0), the procedure key was extended to include the version number component, as follows:

SWProc->getKey = *HostingNode*|*Name*|***ProcMajorVer***|***ProcMinorVer***

If both your SPO client and TIBCO SPO Server support procedure version control (message interface version 3.0.0 or newer), this new key format must be used when invoking the **itemByKey** method to extract an **SWProc** object from the lists returned by these methods: **getProcs**, **getAuditProcs**, **getStartProcs**, and **getProcVersions**.

SPO applications that have procedure keys “hard coded” must be modified to include the procedure version component (*ProcMajorVer*|*ProcMinorVer*) if both the client and TIBCO SPO Server support procedure version control. If the **itemByKey** method is invoked without the version component in the key, an **E_InvalidKeyErr** error is thrown. (Another solution is to use the **makeProc** method (which doesn't require the version number — it will default to the "default" version) to get the **SWProc** object, then use **getKey** to get the full key for use where it is required.)

If you include the procedure version number component in the key, but your TIBCO SPO Server does not support procedure version control, the client will be aware that the server does not expect the version component and will not include it when the key is sent to the server.

6 Other Information

6.1 Other TIBCO SPO C++ Client Documentation

For detailed information about the TIBCO SPO C++ Client, see the on-line help and the *SPO Programmer's Guide* provided with the product.

6.2 Latest Product Information

For the latest TIBCO Staffware Process Suite product information, please refer to the TIBCO Support Services website at http://www.tibco.com/services/support.

6.3 Re-Branding of Staffware Software and Documentation

Staffware software and documentation is currently being re-branded. For example, the Staffware iProcess Engine is now called the TIBCO Staffware iProcess Engine. Until the re-branding is complete there may be some naming inconsistencies in the Staffware software and documentation.

7 Change History

The following Change Requests (CRs) have been implemented in each release of the TIBCO SPO C++ Client for HP-UX.

7.1 Version 10.2.2

CR Number	Description
CR 17198	*The **SWNodeInfo->getStatus** method always returns **swAvailable**.* Corrected.
CR 17150	*If the desired node does not respond to a directed UDP message (**addNode** method), an error is written to the log. Since UDP messages are not guaranteed, this should not be logged as an error.* Corrected. The log level for a non-response to a directed UDP message has been changed from **swLogError** to **swLogWarning**.
CR 17126	*The TIBCO SPO C++ Client needs to allow passwords up to 4096 characters.* Implemented. To use passwords up to 4096 (from the previously allowed 255 characters), you must also be using a TIBCO SPO Server that includes CR 17143 and a TIBCO Staffware iProcess Engine that includes CR 17117.
CR 16974	*Performing fast and frequent logins/disconnects may cause the client to run out of available ports.* Corrected.

CR Number	Description
CR 16970	*The TIBCO SPO C++ Client needs to support the TIBCO SPO Director.* Implemented. See "Support Added for the TIBCO SPO Director (CR 16970)" on page 4.

7.2 Version 10.2.0

CR Number	Description
CR 16907	*The **makeViewItemsByTag** method is returning an "Unauthorized to view workitems in Work Queue" error, even though the user has the proper authorization for the work queue.* Corrected.
CR 16814	*New constants need to be added to the **SWNodeInfoStatusType** enumeration to fully describe the status of a TIBCO SPO Server made available by a TIBCO SPO Director.* Implemented. See "New TIBCO SPO Server Status Types (CR 16814)" on page 4.
CR 16726	*The following extraneous message can appear in the Client Log: "swrtns.GetSockRef: Delete Socket Not Found. (ConnectId = 0x9)."* Corrected. This message will no longer appear.
CR 16667	*New error constants can be returned to the TIBCO SPO C++ Client from the TIBCO SPO Server.* Implemented. See "New Error Constants Added (CR 16667)" on page 4.
CR 16452	*When locking a previously locked work item, the **getLastError** method on **SWWorkItem** returns FF7A, which is not a valid error number.* Corrected.
CR 16252	*If connected to a TIBCO SPO Server with an interface version older than 1.3.1, the **getUrgentCnt**, **getDeadlineCnt**, and **getUnopenedCnt** methods return unpredictable results (whatever is in memory).* Corrected. These methods now return -1 if connected to a TIBCO SPO Server with an interface version older than 1.3.1.
CR 16225	*The **getLength** method on **SWField** and **SWFMarking** returns 0 (zero) for memo fields. It should return the actual length of the memo field so the appropriate amount of memory can be allocated.* Corrected. The **getLength** method now returns the actual length of memo fields.
CR 16207	*Accessing the **SWNodeInfoEx** object when there are a large number of clients can cause a stack overflow, resulting in a crash.* Corrected.
CR 16177	*An enhancement needs to be made to the TIBCO SPO C++ Client to allow you to specify which markings will be returned from the TIBCO SPO Server when the work item is locked, regardless of whether they are marked on the form or result from conditional statements.* Implemented. See "New Methods Added to Retrieve Markings When Locking Work Items (CR 16177)" on page 5.

CR Number	Description
CR 16084	*The **getWorkItemTag** method on **SWOutstandingItem** is returning an invalid tag. This only occurs if the client is connected to an older TIBCO SPO Server (message interface version 3.0.2 or older).* Corrected.
CR 16065	*The **SWStep** object needs a method to hold the extended description that can be entered when a normal-type step is defined in the SPD.* Implemented. See “Extended Description Method Added to SWStep (CR 16065)” on page 5.
CR 16045	*The TIBCO SPO C++ Client needs to support transaction control steps.* Implemented. See “Transaction Control Steps Added (CR 16045)” on page 5.
CR 15758	*The TIBCO SPO C++ Client needs to be modified to work with a TIBCO SPO Server that uses a data format of “YDM”. Currently, an “Invalid Date Format” error is being thrown when a method is called that performs a date operation if the TIBCO SPO Server is using a “YDM” date format.* Corrected.
CR 12885	*The key for the **SWFwdItem** object has a hard-coded “R”, indicating that work items can only be forwarded to released queues.* Corrected. The key for SWFwdItem has been modified to include test queues. See “Key for SWFwdItem Object Modified (CR 12885)” on page 7.
CR 12550	*An enhancement needs to be made to the TIBCO SPO C++ Client to allow you to specify which CDQP fields to return from the server with work items that reside in an **SWXList**.* Implemented. See “New Methods Allow you to Control CDQP Fields Returned from Server (CR 12550)” on page 7.
CR 10642	*An enhancement needs to be made to the TIBCO SPO C++ Client to allow you to specify which CDQP fields to return from the server when you create work items with the "make..." methods.* Implemented. See “New Methods Added to Create Work Items and Specify CDQPs and Case Fields (CR 10642)” on page 7.

7.3 Version 10.0(4.2)

CR Number	Description
CR 15469	*Methods that are passed NodeKey and NodeInfoKey parameters are not properly adding the default instance number to the keys.* Corrected. The keys for the **SWNode** and **SWNodeInfo** objects will now always include the *IsDirector* and *InstanceNumber* values. See “SWNode and SWNodeInfo Keys Modified (CR 15469)” on page 8.
CR 15392	*Enabling persistence on an **SWXList** that was returned by the **getXListPredict** method results in an erroneous “invalid socket” error.* Corrected.

CR Number	Description
CR 15247	*Enumeration constants need to be added to* ***SWAuditActionType*** *to describe steps that are released without an addressee or sub-procedures specified.* Implemented. See “Enumeration Constants Added to SWAuditActionType (CR 15247)” on page 8.
CR 15211	*Enabling persistence on an* ***SWXList*** *that was returned by the* ***getXList*** *method results in an erroneous “invalid socket” error.* Corrected.
CR 15209	*The* ***interfaceEqual*** *and* ***interfaceNewer*** *methods are not checking for valid parameters (specifically, missing brackets around the interface version number). Also, the* ***interfaceEqual*** *method is no longer supporting the use of the “*” wild card character.* Corrected.
CR 15193	*Using the RogueWave Wide strings class in the TIBCO SPO C++ Client prevents it from scaling well on systems with multiprocessors.* Corrected. Instead of using the RogueWave strings, TIBCO’s own string class, **SWWString**, is now used.
CR 15145	*Extraneous “item not found” messages are occuring when calling the* ***getFields*** *and* ***getMarkings*** *methods.* Corrected.
CR 15144	*The following exception can occur when retrieving items in an* ***SWList****: “Unrecoverable error detected - Call Staffware”. This also results in the following message being written to the client log: “MsgBase.FindDelim: No new buffers”. The problem is a timing issue with parsing the message returned from the SPO Server.* Corrected.
CR 15034	*Steps that have a deadline type of* ***swPeriod*** *return the time incorrectly.* Corrected.
CR 14985	*The TIBCO SPO C++ Client needs to be enhanced to allow logging into multiple instances of the SPO Server, as well as a means to limit the number of cases retrieved from the server when using SWXLists.* Implemented. See “Allow Logging Into Multiple Instances of the TIBCO SPO Server (CR 14985)” on page 9 and “Ability to Limit Number of Cases Retrieved from TIBCO SPO Server Added (CR 14985)” on page 9.
CR 14977	*Setting the filter expression (with the* ***setFilterExpression*** *method) for an* ***SWXList*** *to an invalid expression or too large of an expression, then rebuilding the* ***SWXList*** *results in the expected “data too big or truncated” error. However, calling methods on the* ***SWXList*** *object after this error causes an assert in MsgBase:GetStrFakeIt and a subsequent access violation crash of the client.* Corrected.
CR 14784	*Calling the* ***makeWorkItemByTag*** *method in a multi-node environment returns a “Queue not found” error.* Corrected.

CR Number	Description
CR 14770	*The ability to get the date and time a case was started needs to be added.* Implemented. See "Methods Added to Obtain Case Start Date and Time (CR 14770)" on page 10.
CR 14619	*Currently, the TIBCO SPO Server calls the **MOVESYSINFO** function after every administrative change. This can tie up the background and WIS/WQS processes for long periods of time if there are lots of users. A method needs to be added so the client can explicitly call the **MOVESYSINFO** function.* Corrected. The **moveSysInfo** method has been added to perform an explicit MOVESYSINFO. See "Method Added to Perform MOVESYSINFO Function (CR 14619)" on page 10.
CR 14585	*The **setDefCriteria** method on **SWWorkQ** sets the default filter and sort criteria based on the current criteria settings returned by the **getWorkItems** method on **SWView**. This is not practical when using **SWXLists**.* Corrected. A new method has been added to **SWWorkQ** that allows you to pass in the filter and sort criteria to be used as the default criteria. See "New Method Added to Set Default Criteria (CR 14585)" on page 10.
CR 14467	*Some error messages generated by the TIBCO SPO C++ Client and Server do not match the messages shown in the on-line help. There are also messages that include "SEO" instead of "SPO."* Corrected. Some TIBCO SPO C++ Client and Server error messages have been modified so that there is consistency between the on-line help and the actual message that is generated.
CR 14459	*External form data for EAI steps is being returned from the server when procedure definitions are retrieved. This data should be returned only when you call the **getExtForm** method.* Corrected. External form data on EAI steps is now retrieved from the server only when the **getExtForm** method is called.
CR 14436	*A value of "00" is always returned for the seconds portion of the date returned by **getTimeStarted** and **getTimeEnded** on **SWPredictedItem**.* Corrected.
CR 14382	*Procedure audit data is being returned from the server when the **isWithAuditData** flag on **SWProc** is set to False.* Corrected.
CR 14363	*The **makeAWorkQ** method on **SWAdmin** fails if it is called without specifying a node name. If both the node name and the work queue name are specified, but there are a large number of work queues, the method exits without returning an error.* Corrected.
CR 14321	*If the TIBCO SPO Server generates an error when the **isPersisted** flag on **SWCriteriaWI** is set, the TIBCO SPO C++ Client ignores the error.* Corrected. The TIBCO SPO C++ Client will now throw an error returned by the TIBCO SPO Server when setting the **isPersisted** flag.

CR Number	Description
CR 14223	*A large number of audit trail records can cause the client to abend with a stack overflow (the exact number of records that will cause a stack overflow is unknown, but it appears to be platform dependent).* Corrected. Temporary memory allocated to the stack upon creation of audit trail objects is now deallocated after the object creation, preventing the stack size from increasing.
CR 14204	*Calling the **makeCaseByTag** method results in an "Invalid Parameter" error.* Corrected.
CR 14199	*Passing a blank string to the **changePassword** method causes the TIBCO SPO C++ Client application to crash.* Corrected.
CR 14176	*Using a filter expression greater than 4K causes the TIBCO SPO C++ Client to crash. This is caused by memory being overwritten when the filter expression is written to the client log (only occurs if the log level is set to swLogDebug).* Corrected. Log entries greater than 4K are now truncated at 4K. It will no longer cause the client to crash.
CR 14165	*Accessing an **SWList** that contains more than 32K items causes the application to abend.* Corrected. An **SWList** can now contain more than 32K of items. The list's index size has been changed from a short to an unsigned long integer.
CR 14158	*If the definition of an external form for an EAI step (available with the **getExtForm** method on **SWStep**) exceeds the value set for the TIBCO SPO Server response buffer, the TIBCO SPO Server crashes. (The response buffer size is set with the TIBCO SPO Server configuration parameter, **TCPResponsePages** — it defaults to 4K bytes.)* Corrected. The TIBCO SPO C++ Client was modified to now send large external form definitions in multiple messages. A new TIBCO SPO Server error was also added, which is returned by the TIBCO SPO Server if an attempt is made to write a value to the response buffer that is larger than the size specified by the **TCPResponsePages** parameter. See "New TIBCO SPO Server Error for Buffer Overflow (CR 14158)" on page 10.
CR 14111	*Currently, object constructor and destructor information is written to the client log by default, which can result in a very large log file in a short period of time. The default client log categories need to be changed so that all object constructor and destructor information is not written to the log by default.* Corrected. Now when the swLogDebug level is used, object constructor and destructor information is not written to the log by default. See "Client Log Default Categories Changed (CR 14111)" on page 11.
CR 13902	*Attempting to rebuild an **SWWorkItem** object for a work item that has been released results in a "SWServer Error: Unknown" error. The error message that should be returned is "Queue item not found".* Corrected.
CR 13869	*Setting the **MessageWaitTimeout** environment variable to a very large value (e.g., 3600000 milliseconds) causes client logins to fail and returns error 2055.* Corrected.

CR Number	Description
CR 13836	*Setting the* ***isRebuildAll*** *flag to True, then rebuilding the work queue causes the TIBCO SPO C++ Client to crash.* Corrected.
CR 13821	*Attempting to access a work item on an* ***SWXList*** *causes an Access Violation error.* Corrected.
CR 13814	*Accessing a work item on an* ***SWXList*** *that has been locked may result in an empty list of markings for the work item. This occurs only on subsequent accesses when the* ***isKeepLocalItems*** *flag is set to False.* Corrected.
CR 13785	*A new method is needed so that "Admin" users can get a list of the work queues they are authorized to supervise. Currently, the* ***getAWorkQs*** *method on* ***SWAdmin*** *returns a list of all work queues when accessed by an Admin user.* Implemented. New methods have been added to **SWAdmin** and **SWUser** to allow this functionality. See "Methods Added to List Supervised Work Queues (CR 13785)" on page 11 for more information.
CR 13783	*Attempting to launch a sub-procedure is causing an error message in the* ***sw_warn*** *file. This is occurring because the TIBCO Staffware iProcess Engine is assuming a "precedence" of "released" for the sub-procedure, but no released version of the sub-procedure exists. SPO is not specifying a precedence for starting sub-procedures.* Corrected. A new parameter has been added to the **startCase** and **startCaseEx** methods to allow the user to specify sub-procedure precedence. A new enumeration type is also added to define each precedence that can be specified. See "Sub-Procedure Precedence Added to SPO (CR 13783)" on page 11 for more information.
CR 13770	*The* ***makeProc*** *and* ***makeStep*** *methods allow you to pass in only a ProcMinorVer number in the method call, without the ProcMajorVer number. The procedure/step is created and an error is not produced. These methods should require both ProcMajorVer and ProcMinorVer number if either is provided.* Corrected. The TIBCO SPO C++ Client now throws the "SWClient Error: Invalid Parameter" error if either the *ProcMajorVer* or *ProcMinorVer* number is provided, but not the other.
CR 13768	***SWProcStatusType*** *should contain an entry for withdrawn-incomplete procedures.* Corrected. See "New entry added to SWProcStatusType (CR 13768)" on page 12.
CR 13678	*The SPO object model needs to be expanded to allow runtime determination of all steps that are currently outstanding in the case. Previously, only steps that resulted in a work item being placed in a work queue were identifiable as outstanding.* Implemented. The user can now determine which normal, sub-procedure call, dynamic sub-procedure call, event, EAI, and graft steps are outstanding. See "Ability to Identify all Outstanding Steps Added (CR 13678)" on page 12.
CR 13597	*If the* ***SWList*** *returned by* ***SWStep.getPublicFields*** *is rebuilt, the list will contain incorrect information.* Corrected.

CR Number	Description
CR 13562	*Segmentation violations are occurring when running under heavy load conditions in a multi-threaded environment.* Corrected.
CR 13468	*Calling the **SWNode.makeProc** method without providing the NodeName parameter fails. The method call is failing to determine the default nodename.* Corrected.
CR 13408	*When calling the **rebuild** method on **SWXList** that contains work items, the status returned by the method call does not reflect the current status of the work items.* Corrected.
CR 13209	*The "swUnreleased" enumeration in **SWProcStatusType** is misnamed "swUnrelease".* Corrected. The enumeration has been renamed "swUnreleased".
CR 13156	*The value returned by the **isSuspended** method is invalid if that method is called on **SWCase** prior to calling the **getTimeTerminated** or **isActive** method.* Corrected.
CR 12946	*Duplicate **SWXLists** are created on the TIBCO SPO Server, increasing memory usage at the server. This is occurring when the **rebuild** method is called on the **SWXList**, but the reply is not immediately parsed (asynchronous communication). Then if the **isPersisted** flag is set before any other method on the **SWCriteriaWI** object is called, another message is sent to the server because the **PersistenceID** is empty. This causes another **SWXList** to be created at the server.* Corrected. Now when the **rebuild** method is called, the reply is immediately parsed (synchronous communication), so that the **PersistenceID** is available. Setting the **isPersisted** flag to inform the server to persist the **SWXList** does not cause another **SWXList** to be created.

7.4 Version 10.0(0.1)

CR Number	Description
CR 13099	*When a step is created with the **makeStepByTag** method, the **isPrediction** flag on the step returned may be incorrect. The **isPrediction** flag is always taking on the value to which the **isPublic** flag is set.* Corrected.
CR 13086	*A memory leak is occurring when the value of a marking is set to SW_NA (for "not assigned).* Corrected.
CR 13062	*If the TIBCO SPO Server is heavily loaded and there is a receive timeout, an assert may occur, resulting in a client crash.* Corrected.
CR 13059	*There is a memory leak when certain errors occur.* Corrected. Error handling for the **item** and **itemByKey** methods on **SWView** is now done the same way as on **SWList** and **SWXList**.

CR Number	Description
CR 12986	*The addition of array fields (see CR 12590) requires that a FieldType argument be added to some method signatures for* ***startCaseEx*** *and* ***simulateCase****.* Implemented. See “New FieldType Argument on startCaseEx and simulateCase Methods (CR 12986)” on page 15.
CR 12972	*The empty object that must be passed into “make” methods never gets initialized if the call fails with an exception. An attempt to delete the uninitialized object at a later time results in unexpected exceptions.* Corrected.
CR 12840	*Calling the* ***lockItemsEx*** *method on* ***SWWorkQ*** *causes a fatal assertion.* Corrected.
CR 12828	*Calling the* ***deleteEntUser(s)*** *method on* ***SWEnterprise*** *fails with an E_Inval_Method exception.* Corrected.
CR 12819	*Releasing a work item on an* ***SWXList*** *causes an error.* Corrected.
CR 12741	*If an error is returned from the server as a result of a* ***login****, an abend occurs.* Corrected.
CR 12608	*New TIBCO SPO Server log categories need to be added to* ***SWSrvLogCategoryType****.* Implemented. See “New Server Log Categories Added (CR 12608)” on page 16.
CR 12590	*The TIBCO SPO C++ Client needs to support ad-hoc processing and procedure version control.* Implemented. New objects and methods have been added to the object model to support this functionality. See “Array Fields, Ad-Hoc Processing and Procedure Version Control Added (CR 12590)” on page 16.
CR 12546	*The* ***getItemCount*** *method on* ***SWXList*** *does not return the correct value.* Corrected.
CR 12446	*The TIBCO SPO C++ Client cannot display the contents of the script on a Script step.* Corrected. The **getScript** method has been added to **SWStep** to provide access to the contents of the script. See “Method Added to get Script Contents (CR 12446)” on page 21.
CR 12427	*The ability to specify whether more than one block of items should be held locally after they are retrieved from the TIBCO SPO Server needs to be added. This would be used to prevent blocks of items from accumulating and consuming memory.* Corrected. The **isKeepLocalItems** and **setKeepLocalItems** methods were added to **SWXList**. See “Methods Added to Discard Local Blocks of Items (CR 12427)” on page 21.
CR 12414	*Customer configuration information needs to be included in the TIBCO SPO C++ Client Log.* Implemented. The log will now include general system information, such as the platform, number of processors, etc.

CR Number	Description
CR 12411	*The TIBCO SPO C++ Client cannot display the audit trail message.* Corrected. The **getMessage** method was added to **SWAuditStep** to provide access to the audit trail message. See "Method Added to get Audit Trail Message (CR 12411)" on page 21.
CR 12410	*Additional counts need to be available for work items on an* ***SWXList****.* Implemented. See "New Counts for Work Items on an SWXList (CR 12410)" on page 21.

7.5 Change History in Previous Versions

For a list of the change requests that were incorporated in each release of the TIBCO SPO C++ Client for HP-UX prior to Version 10.x, see the **ChangeHistory.pdf** file on the distribution CD.