TIBCO iProcess® User Validation API

User's Guide

Software Release 11.4.1
April 2014

Two-Second Advantage®

Important Information

Contents

Preface

This guide describes how to use the TIBCO iProcess User Validation API (UVAPI) to create your own user validation system, which you can use with TIBCO iProcess Engine. The API can be used to create a shared library for UNIX or a DLL for Windows.

You should read this guide in conjunction with the source code and sample User Validation package provided as part of the TIBCO iProcess Engine installation.

This guide is aimed at application developers with a knowledge of C programming.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO iProcess Engine Documentation

The following documents form the TIBCO iProcess Engine documentation set:

- *TIBCO iProcess Engine Installation* Read this manual for instructions on site preparation and installation.
- *TIBCO iProcess Engine Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.
- TIBCO iProcess Suite Documentation This documentation set contains all the manuals for TIBCO iProcess Engine and other TIBCO products in TIBCO iProcess® Suite. The manuals for TIBCO iProcess Engine are as follows:
  - *TIBCO iProcess Engine Architecture Guide*
  - TIBCO iProcess Engine Administrator's Guides:

    TIBCO iProcess Engine Administrator's Guide

    TIBCO iProcess Objects Director Administrator's Guide

    TIBCO iProcess Objects Server Administrator's Guide
  - TIBCO iProcess Engine Database Administrator's Guides:

    TIBCO iProcess Engine (DB2) Administrator's Guide

    TIBCO iProcess Engine (Oracle) Administrator's Guide

    TIBCO iProcess Engine (SQL) Administrator's Guide
  - *TIBCO iProcess swutil and swbatch Reference Guide*
  - *TIBCO iProcess Engine System Messages Guide*
  - *TIBCO iProcess User Validation API User's Guide*

Other TIBCO Product Documentation

You may find it useful to read the documentation for the following TIBCO products:

- TIBCO ActiveMatrix BusinessWorks™
- TIBCO Business Studio™

- TIBCO Enterprise Message Service™
- TIBCO Hawk®
- TIBCO Rendezvous®

Typographical Conventions

TIBCO iProcess Engine can be run on both Microsoft Windows and UNIX/Linux platforms. In this manual, the Windows convention of a backslash (\) is used. The equivalent pathname on a UNIX or Linux system is the same, but using the forward slash (/) as a separator character.

UNIX or Linux pathnames are occasionally shown explicitly, using forward slashes as separators, where a UNIX or Linux-specific example or syntax is required.

Any references to UNIX in this manual also apply to Linux unless explicitly stated otherwise.

The following typographical conventions are used in this manual

Table 1 General Typographical Conventions

Convention	Use
SWDIR	TIBCO iProcess Engine installs into a directory. This directory is referenced in documentation as *SWDIR*. The value of *SWDIR* depends on the operating system. For example, • on a Windows server (on the `C:` drive) if *SWDIR* is set to the `C:\swserver\staffw_nod1` directory, then the full path to the `swutil` command is in the `C:\swserver\staffw_nod1\bin\swutil` directory. • on a UNIX or Linux server if *SWDIR* is set to the `/swserver/staffw_nod1` directory, then the full path to the `swutil` command is in the `/swserver/staffw_nod1/bin/swutil` directory or the *$SWDIR*`/bin/swutil` directory. **Note**: On a UNIX or Linux system, the environment variable $*SWDIR* should be set to point to the iProcess system directory for the *root* and *swadmin* users.
IPEADMIN	Indicates the operating system account that is used to administer the iProcess Engine node.
IPESERVICE	Indicates the Windows account that is used to run the iProcess Engine node. (*Not used on UNIX*.)
IPEBAKGROUND	Indicates the UNIX user account that owns most iProcess Engine files and is used to run the iProcess Engine background processes. (*Not used on Windows*.)

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand [`**`enable`**` | disable]`
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*. • To introduce new terms. For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical `OR` that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand param1 | param2 | param3`

`{ }`	A logical group of items in a command. Other syntax notations may appear within each logical group.
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	`MyCommand {param1 param2} | {param3 param4}`
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	`MyCommand {param1 | param2} {param3 | param4}`
	In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.
	`MyCommand param1 [param2] {param3 | param4}`

Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access TIBCO Documentation

You can access TIBCO documentation here:

http://docs.tibco.com

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support as follows:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

Chapter 1 Introduction

The software development kit (SDK) for the TIBCO iProcess User Validation API contains the following components:

File Description	On Windows	On UNIX
Makefile	uvapiw32.mak	uvapiunx.mak
Header file	swuvapi.h	swuvapi.h
Encryption object	uvapienc.obj	uvapienc.o
Sample user validation application code	swuvamod.c	swuvamod.c
Test application	tstuvapi.c, tstuvapi.exe	tstuvapi.c, tstuvapi.exe

Topics

What is iProcess User Validation?

The default method of user validation in the TIBCO iProcess Engine requires you to create operating system accounts for each registered iProcess user. Also, the login passwords are maintained by the operating system.

If you have different security validation requirements, you can use the User Validation API to create your own method of user validation to match your business needs. For example, you might want to create operating system users but have a different form of password validation. Alternatively, you might want to completely separate the users and their passwords from the operating system by maintaining them in a database of your choice.

The User Validation API provides interfaces that you can use to maintain your list of iProcess users and their passwords. It essentially isolates iProcess from validating users against the operating system so that you can provide your own user identity and validation system.

The following core iProcess users are still required as operating system accounts.

O/S Account	Windows (SQL Server or Oracle)	UNIX/Oracle	UNIX/DB2
IPEBAKGROUND	n/a	Required	Required
IPEADMIN	Required	Required	Required
IPESERVICE	Required	n/a	n/a
iProcess Engine DB Schema Owner	n/a	n/a	Required
iProcess Engine DB user	n/a	n/a	Required

System Requirements

To use the TIBCO iProcess User Validation API, you need a computer with the following hardware and software:

- Windows 2003/XP or UNIX platform
- C compiler such as Microsoft Visual Studio.

Installation

The TIBCO iProcess User Validation API is installed as part of the TIBCO iProcess Engine installation. The files are placed the directory *SWDIR***sdks****uvapisdk**.

UNIX Platform

The following files are installed.

Filename	Description
uvapiunx.mak	Makefile for example UVAPI package & test utility
uvapienc.o	TIBCO supplied encryption/API object
swuvamod.c	Source code for implementation of example UVAPI package
swuvamod.o	Source definition object file
tstuvapi.c	Source code for test utility
tstuvapi	Generated test utility executable
tstuvapi.o	Test definition object file
uvapi.so	Generated example UVAPI package as a Shared Library

Windows Platform

The following files are installed:

Filename	Description
uvapiw32.mak	Makefile for example UVAPI package & test utility
uvapienc.obj	TIBCO supplied encryption/API object
swuvamod.c	Source code for implementation of example UVAPI package
swuvamod.def	Source definition file
swuvapi.h	Source header file

Filename	Description
tstuvapi.c	Source for test utility
uvapi.dll	Generated example UVAPI package as a Dynamic Link Library
tstuvapi.exe	Generated test utility executable

Using the User Validation API

A generic User Validation interface has been created to enable the TIBCO iProcess Engine to retrieve a list of iProcess users and to check and change their passwords. The details of this interface are published in the User Validation API.

The following is a list of functions provided by the API:

- UVAPI package initialization
- List the possible iProcess users
- Validate a user name as a possible iProcess user
- Validate the password for a possible iProcess user, the maximum length of TIBCO iProcess Engine password is 32768 bytes
- Change the password for a possible iProcess user
- Provide a user identity for the current execution context
- Set a user identity for the current execution context
- UVAPI package termination

The concept of an operating system home directory for iProcess users no longer exists. The user's home directory from a iProcess perspective will be SWDIR**queues***username*.

IMPORTANT!

You must ensure that any user validation package you create using the User Validation API is threadsafe. This is because within each TIBCO iProcess Engine process, multiple threads may call the User Validation API interfaces during normal TIBCO iProcess Engine operation.

To ensure that your user validation package is threadsafe, make sure that you adhere to the following guidelines:

- Make sure that any modules in your user validation package that use User Validation API interfaces use threadsafe code.
- Use mutual exclusion locks (mutexes) to prevent multiple threads from simultaneously executing any critical sections of code that are *not* threadsafe, but that access shared data.
- When you build the user validation package, make sure that you use the appropriate flags (for your chosen operating system and compiler) to link the application using threadsafe libraries.

Deploying a non-threadsafe user validation package can cause TIBCO iProcess Engine processes to fail.

The iProcess Encryption Layer Object

The following encryption objects are provided in the SDK so that you can create replacement UVAPI packages:

- **uvapienc.obj** - for Windows 2003/XP
- **uvapienc.o** - for UNIX

Text strings passed across the UVAPI are encrypted using a proprietary TIBCO mechanism. You need to use this object when implementing a new UVAPI package to provide the encryption/decryption of parameter strings.

The Header File

The header file called **swuvapi.h** is provided for inclusion by applications using the UVAPI package. It contains:

- Type definitions
- Literal constants (return codes and flag values)
- Function prototypes (API functions)

Compiling Your UVAPI Package

Any user validation package you create using the User Validation API must be compiled using the GNU gcc/g++ set of compilers.

The TIBCO iProcess User Validation API Sample Application

The SDK for the TIBCO iProcess User Validation API provides a sample application that supports all of the UVAPI interfaces. The package is supplied as a single module (**swuvamod.c**) with the **swuvapi.h** header file and a makefile.

The user database for the example is a text file (**exuvapi.dat**) which defines the iProcess users, their descriptions and passwords. There is one entry per line with fields separated by the \ character. The example text file needs to be located in your SWDIR**util** directory.

IMPORTANT!

The sample application is intended only as a simple example that demonstrates the use of the TIBCO iProcess User Validation API interrfaces. It is **not** a fully-developed, threadsafe application that is suitable for deployment.

You must ensure that any user validation package you create using the User Validation API is threadsafe - see page 7 for more information.

Deploying a non-threadsafe user validation package can cause TIBCO iProcess Engine processes to fail.

Build Instructions

The following sections describe how to build the UVAPI example application and test application.

UNIX Platforms

To build the example UVAPI package and test utility, enter the following command in the directory where your files are located:

```
make -f uvapiunx.mak
```

This produces the following files:

- **uvapi.so** (the UVAPI package as a shared library)
- **tstuvapi** (the test utility executable).

The build environment requires modifications for different UNIX platforms. The **uvapiunx.mak** makefile contains a section that sets up the environment for the target platform. Edit this section as appropriate for your target platform. Examples for Solaris and AIX are given as comments in the makefile.

To run the test utility, the UVAPI package must be locatable as a shared library (LD_LIBRARY_PATH environment variable on Solaris, LIBPATH environment variable on AIX). For the example UVAPI package, the SWDIR environment variable must also be set, and the file *SWDIR*/**util**/**exuvapi.dat** must exist (see the UVAPI developers documentation for details of the example UVAPI package).

The test utility only calls the Initialisation and Termination interfaces in the UVAPI package as all other interfaces require iProcess encrypted strings to be passed in or returned.

Windows Platforms

To build the example UVAPI package and test utility, set up the environment for the Microsoft Visual Studio V6 C compiler and enter the following command:

```
nmake -f uvapiw32.mak
```

This produces the following files:

- **uvapi.dll** (the UVAPI package as a Dynamic Link Library)
- **tstuvapi.exe** (the test utility executable).

To run the test utility, the UVAPI package must be locatable as a DLL (on the PATH or in the current directory). For the example UVAPI package, the **tstuvapi.exe** utility and **uvapi.dll** components must be located in the **util** sub-directory of an iProcess installation. The file *SWDIR***util****exuvapi.dat** must exist.

The test utility only calls the Initialisation and Termination interfaces in the UVAPI package as all other interfaces require iProcess encrypted strings to be passed in or returned.

Chapter 2 The TIBCO iProcess User Validation API

The TIBCO iProcess User Validation API is a series of interfaces that are called by iProcess and are required in the User Validation package. The U nser Validation API is a C interface and therefore must provide interfaces that can be called from C.

All functions return an integer variable to indicate the completion status of the call (with the exception of uva_initialise). The return values can be found in the **swuvapi.h** header file and they are also described in Return Values on page 40.

Topics

Developing a Replacement User Validation Package

The following sections provide important guidelines for developers creating a new user validation package that uses the TIBCO iProcess User Validation API interfaces.

Thread Safety

IMPORTANT!

You must ensure that any user validation package you create using the User Validation API is threadsafe - see page 14 for more information.

To ensure that your user validation package is threadsafe, make sure that you adhere to the following guidelines:

- Make sure that any modules in your user validation package that use User Validation API interfaces use threadsafe code.
- Use mutual exclusion locks (mutexes) to prevent multiple threads from simultaneously executing any critical sections of code that are *not* threadsafe, but that access shared data.
- When you build the user validation package, make sure that you use the appropriate flags (for your chosen operating system and compiler) to link the application using threadsafe libraries.

Deploying a non-threadsafe user validation package can cause TIBCO iProcess Engine processes to fail.

Internal Function Names

When developing a new UVAPI package, you must provide internal functions that support the external interfaces detailed in this chapter. The internal function names are based on the external interfaces, but prefixed with “int_”. Therefore, the internal function to support the uva_initialise interface is called int_uva_initialise. The return and argument types are the same as the external interfaces. The only difference is that the external interfaces pass all strings in an encrypted form but the internal functions receive and return all strings as plain text.

The supplied iProcess encryption object provides all of the external interface functions, decrypts/encrypts string parameters and then calls the internal function supplied by you. System security is enhanced by only passing encrypted strings between iProcess and the UVAPI package.

Interface Support

If you do not want to support a particular interface (and the default iProcess action is appropriate) the internal function can return the value ER_NOTIMPLEMENTED. This causes iProcess to perform its default action using the internal functionality for that interface. If you do not want iProcess to perform the default action, you can return the value SW_OK (depending on the interface).

Password Validation on Windows Systems

The information in this section is only relevant to the Windows variant of the iProcess Engine.

If the iProcess Engine is running on a machine that is a member of a domain or a domain controller, it uses the search path provided by the Windows **LookupAccountName** function to find the location it should use to validate a user's password when they try to log in.

However, there are two ways in which you can override this behavior and directly specify the location where password validation is to be performed, either on a per-user basis, or globally for an installation:

1. the SW_DOMAIN user attribute specifies a single valid machine name or domain name that should be used to validate a particular user's password when they attempt to log in to the iProcess Engine. See *TIBCO iProcess Windows (Workspace) Manager's Guide* for more information about this attribute and how to set it.
2. the LOGON_OS_LOCATION process attribute defines the default location where passwords should be validated when any user attempts to log in to the iProcess Engine. See the "Administering Process Attributes" chapter of the *TIBCO iProcess Engine Administrator's Guide* for more information about this attribute and how to set it.

Note that:

- If both attributes are set, the SW_DOMAIN value takes precedence over the LOGON_OS_LOCATION value.
- If the iProcess Engine is running on a standalone machine, passwords are always validated against local machine accounts. The SW_DOMAIN and LOGON_OS_LOCATION attributes are ignored even if they are set.

If you use the SW_DOMAIN or LOGON_OS_LOCATION attributes, your UVAPI package must be able to receive and return the additional information about a user's location, to ensure that their password is checked in that location.

To facilitate this, the UVAPI includes extended (_ex) versions of the following interfaces:

- uva_next_user_ex
- uva_user_info_ex
- uva_change_password_ex
- uva_check_password_ex
- uva_set_user_identity_ex

These interfaces can accept (and, in the case of uva_next_user_ex, return) an iProcess user name in either of the following formats:

Format	Description
name	*name* is the iProcess user name. This format is also supported by the equivalent non-extended interfaces.
name@location	*name* is the iProcess user name. *location* is the value (machine or domain name) provided by either the user's SW_DOMAIN user attribute (if defined), or the value of the LOGON_OS_LOCATION process attribute. This format is **not** supported by the equivalent non-extended interfaces.

If your UVAPI package supports these extended interfaces, they are called instead of the non-extended interfaces. If these interfaces do not exist or return ER_NOT_SUPPORTED (see Interface Support on page 15), the non-extended interfaces are called instead.

You should ensure that you use these extended interfaces if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes.

Creating a Session Handle

The UVAPI package works on a session which is allocated by the uva_initialise function. This function must return a session identifier (handle). You need to be aware that several threads in a iProcess process can be using the UVAPI interfaces so you must make sure that the session allocation and management is performed in a thread-safe manner.

Design Issues

You should be aware that several different iProcess processes will call the UVAPI package while the TIBCO iProcess Engine is running. This can cause problems if the UVAPI package is not designed correctly. The example application provides a good example of this.

The user information is stored in the text file and each iProcess process that uses the UVAPI package loads the contents of the text file into a memory cache. However, these caches are specific to each session and to each iProcess process. Therefore, when an iProcess user causes its iProcess process to perform a change password action, that process updates the main text file and the processes' cache.

This means that other iProcess processes (including the one that validates iProcess passwords) will still be using the original cached copy of the data in the text file. Therefore, the example UVAPI does not reflect a changed password until the iProcess system is shutdown and restarted. This is the only way that all the processes can re-cache the user information.

To avoid this problem, you need to design the UVAPI package as a set of interfaces that communicate with a single server process that maintains the user information, ensuring that any changes to the user information is made available to all the iProcess processes using the UVAPI package.

API Interfaces

The following sections summarize each UVAPI interface. Refer to the source code and sample application for more information about how they are used.

The available interfaces are:

- uva_initialise
- uva_terminate
- uva_next_user
- uva_next_user_ex
- uva_user_info
- uva_user_info_ex
- uva_change_password
- uva_change_password_ex
- uva_check_password
- uva_check_password_ex
- uva_set_user_identity
- uva_set_user_identity_ex
- uva_get_user_identity

uva_initialise

Purpose Initializes the user validation package and creates the session handle.

Prototype

```
UV_SH uva_initialise (void)
```

Return Values

Value	Description
>0	Valid session handle
ER_SYSTEM	Generic (undefined) error

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks This interface can be used to establish and store connection details about a connection to the database that is used for subsequent calls before being terminated.

uva_terminate

Purpose

Stops the user validation package and discards the supplied session.

Prototype

```
UV_RCODE uva_terminate (
   UV_SH uvsh
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle

Return Values

Value	Description
SW_OK	Success
ER_HANDLE	Invalid session handle
ER_SYSTEM	Generic (undefined) error

Refer to Return Values on page 40 for a complete list of possible return values.

uva_next_user

Purpose

Returns the encrypted user and encrypted description buffer for the name and description of the first or subsequent **Valid Possible iProcess User (VPIU)**.

You must use the uva_next_user_ex interface instead of this interface if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes to specify the location where a user's password should be validated. See Password Validation on Windows Systems on page 15 for more information.

Prototype

```
UV_RCODE uva_next_user (
   UV_SH uvsh,
   UV_FLAG fFirstUser
   UV_PSTR pEncrNameBuf,
   UV_SIZE iNameBufSize,
   UV_PSTR pEncrDescBuf,
   UV_SIZE iDescBufSize
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
fFirstUser	IN	First/next VPIU flag
pEncrNameBuf	OUT	Pointer to buffer to receive encrypted VPIU name
iNameBufSize	IN	Maximum length of encrypted VPIU name
pEncrDescBuf	OUT	Pointer to buffer to receive encrypted VPIU description
iDescBufSize	IN	Maximum length of encrypted VPIU description

Return Values

Value	Description
SW_OK	Success
SW_EOF	No more users available
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error

Value	Description
ER_TOOBIG	Value is too large for supplied buffer

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks

This interface is used by iProcess to obtain a list of possible iProcess users (currently used in the operating system user list of the TIBCO iProcess Administrator).

On the initial call to this interface the **fFirstUser** parameter should be set to TRUE (to return the first VPIU), subsequent calls should set fFirstUser to FALSE. The order in which VPIUs are returned by this interface is defined by the implementation of the underlying UVAPI package so the caller should assume no specific order.

uva_next_user_ex

Purpose

Returns the encrypted user and encrypted description buffer for the name and description of the first or subsequent VPIU.

You must use this interface instead of the uva_next_user interface if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes to specify the location where a user's password should be validated. See Password Validation on Windows Systems on page 15 for more information.

Prototype

```
UV_RCODE uva_next_user_ex (
   UV_SH uvsh,
   UV_FLAG fFirstUser
   UV_PSTR pOSUserLocations,
   UV_PSTR pEncrNameBuf,
   UV_SIZE iNameBufSize,
   UV_PSTR pEncrDescBuf,
   UV_SIZE iDescBufSize
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
fFirstUser	IN	First/next VPIU flag
pOSUserLocations	IN	Value of the OS_USER_LOCATIONS process attribute.
pEncrNameBuf	OUT	Pointer to buffer to receive encrypted VPIU name
iNameBufSize	IN	Maximum length of encrypted VPIU name
pEncrDescBuf	OUT	Pointer to buffer to receive encrypted VPIU description
iDescBufSize	IN	Maximum length of encrypted VPIU description

Return Values

Value	Description
SW_OK	Success
SW_EOF	No more users available

Value	Description
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error
ER_TOOBIG	Value is too large for supplied buffer

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks

This interface is an extended version of the uva_next_user interface. It differs from that interface only in the following ways:

- It supports the passing in and out of user location information from the SW_DOMAIN user attribute and/or LOGON_OS_LOCATION process attribute. See Password Validation on Windows Systems on page 15 for more information.
- It has an additional **pOSUserLocations** parameter, which allows the value of the OS_USER_LOCATIONS process attribute to be passed in (for example, if you want to limit the users that are to be returned by the UVAPI layer). See the "Administering Process Attributes" chapter of *TIBCO iProcess Engine Administrator's Guide* for more information about this attribute and how to set it.

uva_user_info

Purpose

Decrypts the supplied encrypted user and returns the encrypted description and bit-encoded flags for the supplied user.

You must use the uva_user_info_ex interface instead of this interface if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes to specify the location where a user's password should be validated. See Password Validation on Windows Systems on page 15 for more information.

Prototype

```
UV_RCODE uva_user_info (
   UV_SH uvsh,
   UV_PSTR pEncrUserName,
   UV_PSTR pEncrDescBuf,
   UV_SIZE iDescBufSize,
   UV_PFLAGS pUserFlags
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
pEncrUserName	IN	Pointer to encrypted VPIU name
pEncrDescBuf	OUT	Pointer to buffer to receive VPIU description
iDescBufSize	IN	Maximum length of VPIU description
pUserFlags	OUT	Pointer to returned user information flags value

Return Values

Value	Description
SW_OK	Success
ER_NOTFOUND	Unknown user
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error
ER_TOOBIG	Value is too large for supplied buffer

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks This interface is used by iProcess to validate a user name as a VPIU, and to determine system related attributes of the user. The TIBCO iProcess Engine currently uses a similar validation when adding users during a Restore operation.

The returned UserFlags will be a bit-encoded value, currently the only defined bits are:

```
#define SWUV_FLAG_OSUSER 1
```

If this bit is set for a VPIU, it means there is a corresponding operating system account for that user.

uva_user_info_ex

Purpose

Decrypts the supplied encrypted user and returns the encrypted description and bit-encoded flags for the supplied user.

You must use this interface instead of the uva_user_info interface if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes to specify the location where a user's password should be validated. See Password Validation on Windows Systems on page 15 for more information.

Prototype

```
UV_RCODE uva_user_info_ex (
   UV_SH uvsh,
   UV_PSTR pEncrUserName,
   UV_PSTR pEncrDescBuf,
   UV_SIZE iDescBufSize,
   UV_PFLAGS pUserFlags
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
pEncrUserName	IN	Pointer to encrypted VPIU name
pEncrDescBuf	OUT	Pointer to buffer to receive VPIU description
iDescBufSize	IN	Maximum length of VPIU description
pUserFlags	OUT	Pointer to returned user information flags value

Return Values

Value	Description
SW_OK	Success
ER_NOTFOUND	Unknown user
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error
ER_TOOBIG	Value is too large for supplied buffer

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks This interface is an extended version of the uva_user_info interface. It is identical to that interface except that it supports the passing in and out of user location information from the SW_DOMAIN user attribute and/or LOGON_OS_LOCATION process attribute. See Password Validation on Windows Systems on page 15 for more information.

uva_change_password

Purpose

Decrypts the supplied encrypted user name and passwords and then change the password for the supplied user to the supplied password.

You must use the uva_change_password_ex interface instead of this interface if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes to specify the location where a user's password should be validated. See Password Validation on Windows Systems on page 15 for more information.

Prototype

```
UV_RCODE uva_change_password (
   UV_SH uvsh,
   UV_PSTR pEncrUserName,
   UV_PSTR pEncrOldPassword,
   UV_PSTR pEncrNewPassword
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
pEncrUserName	IN	Pointer to encrypted VPIU name
pEncrOldPassword	IN	Pointer to encrypted VPIU current password
pEncrNewPassword	IN	Pointer to encrypted VPIU new password

Return Values

Value	Description
SW_OK	Success
ER_NOTFOUND	Unknown user
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error
ER_SECCANTCHNG	Cannot change password for this user
ER_SECBADNEWPSWD	New password is invalid
ER_SECBADUSER	Unknown user

Value	Description
ER_SECUNKNOWN	Generic (undefined) security error
ER_SECBADPSWD	Password is invalid
ER_SECBADPERMS	Permissions are incorrect for this operation

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks This interface is currently used in the iProcess Work Queue Manager.

uva_change_password_ex

Purpose

Decrypts the supplied encrypted user name and passwords and then changes the password for the supplied user to the supplied password.

You must use this interface instead of the uva_change_password interface if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes to specify the location where a user's password should be validated. See Password Validation on Windows Systems on page 15 for more information.

Prototype

```
UV_RCODE uva_change_password_ex (
   UV_SH uvsh,
   UV_PSTR pEncrUserName,
   UV_PSTR pEncrOldPassword,
   UV_PSTR pEncrNewPassword
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
pEncrUserName	IN	Pointer to encrypted VPIU name
pEncrOldPassword	IN	Pointer to encrypted VPIU current password
pEncrNewPassword	IN	Pointer to encrypted VPIU new password

Return Values

Value	Description
SW_OK	Success
ER_NOTFOUND	Unknown user
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error
ER_SECCANTCHNG	Cannot change password for this user
ER_SECBADNEWPSWD	New password is invalid
ER_SECBADUSER	Unknown user

Value	Description
ER_SECUNKNOWN	Generic (undefined) security error
ER_SECBADPSWD	Password is invalid
ER_SECBADPERMS	Permissions are incorrect for this operation

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks

This interface is an extended version of the uva_change_password interface. It is identical to that interface except that it supports the passing in and out of user location information from the SW_DOMAIN user attribute and/or LOGON_OS_LOCATION process attribute. See Password Validation on Windows Systems on page 15 for more information.

uva_check_password

Purpose

Decrypt the supplied encrypted user and password and then return a value indicating whether the password for the supplied user is valid.

You must use the uva_check_password_ex interface instead of this interface if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes to specify the location where a user's password should be validated. See Password Validation on Windows Systems on page 15 for more information.

Prototype

```
UV_RCODE uva_check_password (
   UV_SH uvsh,
   UV_PSTR pEncrUserName,
   UV_PSTR pEncrPassword
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
pEncrUserName	IN	Pointer to encrypted VPIU name
pEncrOldPassword	IN	Pointer to encrypted VPIU password

Return Values

Value	Description
SW_OK	Success
ER_NOTFOUND	Unknown user
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error
ER_SECEXPIRED	Password has expired
ER_SECDISABLED	This user account is disabled
ER_SECBADUSER	Unknown user
ER_SECUNKNOWN	Generic (undefined) security error
ER_SECBADPSWD	Password is invalid

Value	Description
ER_SECBADPERMS	Permissions are incorrect for this operation
ER_USERDISABLED	The account has been disabled
ER_SECBADWKSTN	Login/Validation is not allowed from this location
ER_SECBADHOURS	Login/Validation is not allowed at this time
ER_SECLOCKOUT	The account has been locked

Refer to Return Values on page 40 for a complete list of possible return values.

uva_check_password_ex

Purpose

Decrypt the supplied encrypted user and password and then return a value indicating whether the password for the supplied user is valid.

You must use this interface instead of the uva_check_password interface if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes to specify the location where a user's password should be validated. See Password Validation on Windows Systems on page 15 for more information.

Prototype

```
UV_RCODE uva_check_password_ex (
   UV_SH uvsh,
   UV_PSTR pEncrUserName,
   UV_PSTR pEncrPassword
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
pEncrUserName	IN	Pointer to encrypted VPIU name
pEncrOldPassword	IN	Pointer to encrypted VPIU password

Return Values

Value	Description
SW_OK	Success
ER_NOTFOUND	Unknown user
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error
ER_SECEXPIRED	Password has expired
ER_SECDISABLED	This user account is disabled
ER_SECBADUSER	Unknown user
ER_SECUNKNOWN	Generic (undefined) security error
ER_SECBADPSWD	Password is invalid

Value	Description
ER_SECBADPERMS	Permissions are incorrect for this operation
ER_USERDISABLED	The account has been disabled
ER_SECBADWKSTN	Login/Validation is not allowed from this location
ER_SECBADHOURS	Login/Validation is not allowed at this time
ER_SECLOCKOUT	The account has been locked

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks This interface is an extended version of the uva_check_password interface. It is identical to that interface except that it supports the passing in and out of user location information from the SW_DOMAIN user attribute and/or LOGON_OS_LOCATION process attribute. See Password Validation on Windows Systems on page 15 for more information.

uva_set_user_identity

Purpose

Sets the execution context of the current process to that of the user (or that user's operating system proxy) whose encrypted name is passed in. On UNIX this involves setting the UID and GID.

You must use the uva_set_user_identity_ex interface instead of this interface if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes to specify the location where a user's password should be validated. See Password Validation on Windows Systems on page 15 for more information.

Prototype

```
UV_RCODE uva_set_user_identity (
   UV_SH uvsh,
   UV_PSTR pEncrUserName
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
pEncrUserName	IN	Pointer to encrypted VPIU name

Return Values

Value	Description
SW_OK	Success
ER_NOTFOUND	Unknown user
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks

If the user does not map directly to an operating system identity, the UVAPI package must set the identity of a compatible proxy user. iProcess calls this interface before running Automatic Step programs or ServerRun programs.

uva_set_user_identity_ex

Purpose

Sets the execution context of the current process to that of the user (or that user's operating system proxy) whose encrypted name is passed in. On UNIX this involves setting the UID and GID.

You must use this interface instead of the uva_set_user_identity interface if you use the SW_DOMAIN or LOGON_OS_LOCATION attributes to specify the location where a user's password should be validated. See Password Validation on Windows Systems on page 15 for more information.

Prototype

```
UV_RCODE uva_set_user_identity_ex (
   UV_SH uvsh,
   UV_PSTR pEncrUserName
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
pEncrUserName	IN	Pointer to encrypted VPIU name

Return Values

Value	Description
SW_OK	Success
ER_NOTFOUND	Unknown user
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks

This interface is an extended version of the uva_set_user_identity interface. It is identical to that interface except that it supports the passing in and out of user location information from the SW_DOMAIN user attribute and/or LOGON_OS_LOCATION process attribute. See Password Validation on Windows Systems on page 15 for more information.

uva_get_user_identity

Purpose

Returns the encrypted VPIU name that relates to the current processes execution context.

Prototype

```
UV_RCODE uva_get_user_identity (
   UV_SH uvsh,
   UV_PSTR pEncrUserNameBuf,
   UV_SIZE iNameBufSize
);
```

Parameters

Parameter	Type	Description
uvsh	IN	Session handle
pEncrUserNameBuf	OUT	Pointer to buffer to receive encrypted VPIU name
iNameBufSize	IN	Maximum length of encrypted VPIU name

Return Values

Value	Description
SW_OK	Success
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error
ER_TOOBIG	Value is too large for supplied buffer

Refer to Return Values on page 40 for a complete list of possible return values.

Remarks

If there is no direct mapping between the identity of the execution context and a VPIU, the UVAPI package must supply the name of a compatible VPIU. iProcess will call this interface to determine if the user executing certain iProcess Engine hosted utilities has permissions to perform a requested action.

Return Values

All user validation interfaces (apart from the uva_initialise function) return an integer return code with the following type:

```
typedef int UV_RCODE
```

Return codes are classified as follows:

Value	Description
> 0	Success
0	Failure (but not an error)
<0	Failure (an error condition)

The following values can be returned. Refer to each interface description to see which values each interface returns.

Return Value	Description
SW_OK	Success
SW_EOF	No more users available
ER_HANDLE	Invalid session handle
ER_PARAM	Invalid parameter(s)
ER_SYSTEM	Generic (undefined) error
ER_TOOBIG	Value is too large for supplied buffer
ER_NOTFOUND	Unknown user
ER_SECCANTCHNG	Cannot change password for this user
ER_SECBADNEWPSWD	New password is invalid
ER_SECBADUSER	Unknown user
ER_SECUNKNOWN	Generic (undefined) security error

Return Value	Description
ER_SECBADPSWD	Password is invalid
ER_SECBADPERMS	Permissions are incorrect for this operation
ER_SECEXPIRED	Password has expired
ER_SECDISABLED	This user account is disabled
ER_SECBADWKSTN	Login/Validation is not allowed from this location
ER_SECBADHOURS	Login/Validation is not allowed at this time
ER_SECLOCKOUT	The account has been locked
ER_USERDISABLED	The account has been disabled