



# **TIBCO iProcess® User Validation API**

## **User Guide**

Version 11.9.1 | March 2024

# Contents

---

<b>Contents</b>	<b>2</b>
<b>Introduction</b>	<b>4</b>
What is iProcess User Validation?	4
System Requirements	5
Installation	6
UNIX Platform	6
Windows Platform	7
Using the User Validation API	7
The iProcess Encryption Layer Object	9
The Header File	9
Compiling Your UVAPI Package	9
The TIBCO iProcess User Validation API Sample Application	9
Build Instructions	10
<b>The TIBCO iProcess User Validation API</b>	<b>12</b>
Developing a Replacement User Validation Package	12
Thread Safety	12
Internal Function Names	13
Interface Support	13
Password Validation on Windows Systems	13
Creating a Session Handle	15
Design Issues	15
API Interfaces	16
uva_initialise	17
uva_terminate	17
uva_next_user	18
uva_next_user_ex	20

uva_user_info .....	22
uva_user_info_ex .....	24
uva_change_password .....	26
uva_change_password_ex .....	28
uva_check_password .....	30
uva_check_password_ex .....	32
uva_set_user_identity .....	34
uva_set_user_identity_ex .....	35
uva_get_user_identity .....	37
Return Values .....	38
<b>TIBCO Documentation and Support Services .....</b>	<b>40</b>
<b>Legal and Third-Party Notices .....</b>	<b>42</b>

# Introduction

---

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

File Description	On Windows	On UNIX
Makefile	uvapiw64.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

## 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.

**Note**

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

O/S Account	Windows (SQL Server or Oracle)	UNIX/Oracle	UNIX/DB2
<i>IPEBAKGROUND</i> , where, <i>IPEBAKGROUND</i> 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.	n/a	Required	Required
<i>IPEADMIN</i> , where, <i>IPEADMIN</i> indicates the operating system account that is used to administer the iProcess Engine node.	Required	Required	Required
<i>IPESERVICE</i> , where, indicates the Windows account that is used to run the iProcess Engine node (Not used on UNIX.).	Required	n/a	n/a
<i>iProcess Engine DB Schema Owner</i>	n/a	n/a	Required
<i>iProcess Engine DB user</i>	n/a	n/a	Required

## System Requirements

To use the TIBCO iProcess User Validation API, you require the following hardware and software:

- Windows 10 - Microsoft Visual Studio 2017
- Linux 6.6

**Note:**

If you already have a User Validation API module built for Unix/Linux, then it will need to be rebuilt as a 64 bit library before upgrading from a release prior to versions 11.6.0.

If you already have a User Validation API module built for Windows, then it will need to be rebuilt as a 64 bit library before upgrading from a release prior to versions 11.8.0.

## 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
uvapiunix.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
uvapiw64.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
tstuvapi.c	Source for test utility
uvapi.dll	Generated example UVAPI package as a Dynamic Link Library
tstuvapi.exe	Generated test utility executable

**Note**

TIBCO iProcess Engine 11.8 User Validation API no longer supports a 32 bit library. The TIBCO iProcess Engine 11.8 User Validation API must now be built as a 64 bit library.

## 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

**Note**

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`.

---

**Warning****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
- **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 Microsoft Visual Studio compilers for Windows and GNU gcc/g++ compilers for UNIX platforms.

## 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.

**Warning****IMPORTANT!**

The sample application is intended only as a simple example that demonstrates the use of the TIBCO iProcess User Validation API interfaces. 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 [IMPORTANT!](#) 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 uvapiunix.mak
```

This command produces the following files:

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



**Note:** TIBCO iProcess Engine 11.8.0 is built as a 64-bit application; therefore, the UVAPI package must be built as a 64-bit library as well as the test utility executable.

**i Note:** The build environment requires modifications for different UNIX platforms. The **uvapiunix.mak** makefile contains a section that sets up the environment for the target platform. Edit this section as appropriate for your target platform.

To run the test utility, the UVAPI package must be locatable as a shared library. 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 2017 compiler and enter the following command:

```
nmake -f uvapiw64.mak
```

This command 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.

# 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 User 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 (except [uva\\_initialise](#)). The return values can be found in the **swuvapi.h** header file and they are also described in [Return Values](#).

## 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

---



### Warning

### IMPORTANT!

You must ensure that any user validation package you create using the User Validation API is threadsafe - see [IMPORTANT!](#) 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 section. 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



### Note

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" section of the *TIBCO iProcess Engine Administrator's Guide* for more information about this attribute and how to set it.



Note that:

**Note**

- 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
<i>name</i>	<i>name</i> is the iProcess user name.  This format is also supported by the equivalent non-extended interfaces.
<i>name</i> <i>@location</i>	<i>name</i> is the iProcess user name. <i>location</i> 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 <b>not</b> 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](#)), 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. See 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

See [Return Values](#) 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_RC CODE 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

See [Return Values](#) 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)**.



#### Warning

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](#) for more information.

## Prototype

```
UV_RC CODE 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

Value	Description
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

See [Return Values](#) 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.



#### Warning

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](#) 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

See [Return Values](#) 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](#) 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" section 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.



#### Warning

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](#) 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

Value	Description
ER_TOOBIG	Value is too large for supplied buffer

See [Return Values](#) 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.



#### Warning

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](#) for more information.

### Prototype

```
UV_RC CODE 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

See [Return Values](#) 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](#) 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.



### Warning

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](#) for more information.

## Prototype

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

## Parameters

Parameter	Type	Description
uvsh	IN	Session handle

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

See [Return Values](#) 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.



### Warning

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](#) for more information.

## Prototype

```
UV_RC CODE 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
ER_SECUNKNOWN	Generic (undefined) security error
ER_SECBADPSWD	Password is invalid
ER_SECBADPERMS	Permissions are incorrect for this operation

See [Return Values](#) 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](#) 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.



### Warning

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](#) for more information.

## Prototype

```
UV_RC CODE 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
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

See [Return Values](#) 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.

**Warning**

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](#) for more information.

## Prototype

```
UV_RC CODE 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
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

See [Return Values](#) 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](#) 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.



#### Warning

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](#) 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

See [Return Values](#) 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.



### Warning

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](#) for more information.

## Prototype

```
UV_RCCode 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

See [Return Values](#) 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](#) for more information.

# uva\_get\_user\_identity

## Purpose

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

## Prototype

```
UV_RC CODE 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

Value	Description
ER_TOOBIG	Value is too large for supplied buffer

See [Return Values](#) 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. See 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
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

# TIBCO Documentation and Support Services

---

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

## How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [Product Documentation website](#), mainly in HTML and PDF formats.

The [Product Documentation website](#) is updated frequently and is more current than any other documentation included with the product.

## Product-Specific Documentation

The documentation for this product is available on the [TIBCO iProcess® Engine Product Documentation](#) page.

## How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

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

## How to Join TIBCO Community

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



requests from within the [TIBCO Ideas Portal](#). For a free registration, go to [TIBCO Community](#).

# Legal and Third-Party Notices

---

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, ActiveMatrix BusinessWorks, Business Studio, Enterprise Message Service, Hawk, iProcess, and Rendezvous are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (<https://www.cloud.com/legal>) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: <https://scripts.sil.org/OFL>

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

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

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

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

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

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

Copyright © 1994-2024. Cloud Software Group, Inc. All Rights Reserved.