Sonork Instant Messaging

Visual Basic API Reference Manual

February 2003

Product Version: Sonork Server V 1.7

This manual has the Microsoft Visual Basic References used to Sonork IM Server to Version 1.7.

Visual Basic API Reference Manual
Copyright Notice

© Copyright GTV Solutions, Inc. **1996, 2003**. All rights reserved. May only be used pursuant to a Sonork Messaging Systems Software License Agreement. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without prior written permission of GTV Solutions, Inc. GTV Solutions, Inc. grants you limited permission to make hardcopy or other reproductions of any machine-readable documentation for your own use, provided that each such reproduction shall carry the GTV Solutions, Inc. notice. No other rights under copyright are granted without prior written permission of GTV Solutions, Inc. The document is not intended for production and is furnished "as is" without warranty of any kind. **All warranties on this document are hereby disclaimed, including the warranties of merchantability and fitness for a particular purpose.**
U.S. Government Users Restricted Rights—Use, duplication or disclosure restricted, contact GTV Solutions. Inc.

Trademarks

Sonork Messaging System, and the Sonork logo are trademarks or registered trademarks of GTV Solutions, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Other company, product, or service names may be trademarks or service marks of others.

Notices

References in this publication to Sonork systems, products, programs, or services do not imply that they will be available in all countries in which GTV Solutions, Inc. operates. Any reference to these products, programs, or services is not intended to imply that only Sonork systems, products, programs, or services can be used.
Subject to valid intellectual property or other legally protectable right of Sonork, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by GTV Solutions, Inc., are the responsibility of the user. GTV Solutions, Inc. may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the Sonork Director of Licensing, GTV Solutions, Inc., 49E. 21st. Street, 9th Floor, New York, NY 10010, United States of America.

© Copyright GTV Solutions, Inc. 2003. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted, contact GTV Solutions, Inc.

About This Manual

This Manual describes the different installation procedures that can be used to install or upgrade Sonork Instant Messaging System to Version 1.7

I. Audience

The target audience for this guide is programmers responsible for the programming over Sonork products. Users of this guide should have knowledge of the following:

- PC operating Systems
- Database architecture and concepts
- Networking
- Graphical User Interface
- Microsoft Visual Basic Programming

II. Organization

This manual is organized as follows:

- Chapter 1: Overview
- Chapter 2: Instance life-span
- Chapter 3: Event-processing
- Chapter 4: API Calls
- Chapter 5: Special Values/Enums
- Chapter 6: Data Types

III. Related Documents

Sonork provides the following related documentation:

- Sonork Release Notes: may contain important undocumented information about anything related to Sonork Server that may affect your installation.

You can also view the Sonork Version 1.7 Technical Update for any additional information not included in this manual. You can access the Technical Update from the following URL: http://www.sonork.com/docs/pub/

IV. Reader's Comments

Sonork welcomes any comments and suggestions you have on this and other Sonork manuals.
You can send your comments in the following ways:

- Fax: 1-212-228-9600 Attn: SLD Publications
- Internet electronic mail: readers_comment@sonork.com
- Postal Address: **GTV Solutions**
  Attn: SLD Publications
  49 East 21st Street
  New York, NY 10010
  USA

A Reader's Comment form is located on your system in the following location:
$SONORK/doc/readers_comment.txt
Please include the following information along with your comments:

- The section numbers and page numbers of the information on which you are commenting.
- The version of Sonork that you are using.
- If known, the type of processor that is running the Sonork Messaging System software.

The Sonork Library Department (SLD) cannot respond to system problems or technical support inquiries. Please address technical questions to your local system vendor or to the appropriate Sonork technical support office. Information provided with the software media explains how to send problem reports to Sonork.

V. Conventions

This guide uses several typeface conventions for special terms and actions. These conventions have the following meaning:

- **BOLD** Commands, keywords, file names, authorization roles URLs, names of windows and dialogs, other controls, or other information that you must use literally is in bold.
- Italics Variables and values that you must provide, new terms, and words and phrases that are emphasized are in italics.
-

This Guide uses the UNIX convention for specifying environment variables and directory notation:

- When using the windows command line, replace $variable with %variable% for environment variables and replace each forward slash (/) with a backslash (\) in directory paths.

VI. Publications On-line

When GTV published an update version of one or more online or hardcopy publications, they are posted to the Sonork Information Center. You can access updated publications in the Sonork Information Center from the following Sonork Customer Support Web Site: http://www.sonork.com/docs/pub/
The Sonork Information Center contains the most recent version of the books in the product library in PDF or HTML formats, or both. Translated documents are also available for some publications.

VII. Ordering Publications

You can order many Sonork publications online at the following Web site.
http://www.sonork.com/store/
From this Web page, select Publications and follow the instructions.
In other countries, for a list of telephone numbers, see the following Web site:
http://www.sonork.com/docs/pub/contatcs/

VIII. Customer support

If you have a problem with any Sonork product, you can contact Sonork Customer Support. See the Sonork Customer Support Handbook at the following Web Site: http://www.sonork.com/support/handbook/

Or visit

http://www.sonork.com

Server support

server-support@sonork.com

Client support

client-support@sonork.com

Developer support

developer@sonork.com

Overview

The Sonork VB API is a wrapper to use the Sonork engine from Microsoft's Visual Basic™. It is composed of 2 items:

- **sonorkvblib.tlb:** library type definition that must referenced from visual basic (from Project/References) so that all data types, enumerations and methods are available.
- **sonorkvblib.dll:** DLL containing the API defined by sonorkvblib.tlb

The VB application (The HOST) creates one or more instances of the sonork engine (refer to **Instance life-span**) and obtains a handle for each. To keep the engine running, the host calls the TimeSlot (refer to **SrkTimeSlot**) method which will execute all pending tasks and invoke the application-defined callback (refer to **The callback**).

Instance life-span

Initializing

The host calls **SrkCreateInstance** to create an instance and register its callback (refer to **The callback**) along with an arbitrary tag. The handle returned is required for most API calls. The engine is kept running by invoking **SrkTimeSlot** at least twice per second for each instance. Every time the host's callback is invoked, both handle and tag are included along with the event's data so that the host may use the same procedure as callback for several instances.

Connecting and processing events

The host sets the target server with **SrkSetSonorkServer** and initiates the connection sequence with **SrkConnect**. Once connected, the application executes post-connection initialization tasks such as downloading user list with **SrkRefreshUserList** and/or obtain list of server-side services (e.g. file transfer services) with **SrkRefreshConfig**.
After post-connection initialization tasks are complete, the application set its online presence indicator with **SrkSetSidMode**.
Once presence indicator is set, engine will advise of events by invoking the callback (refer to **The callback**) with an event messages (refer to **SONORK_EVENT_CLIENT_MSG**) when an instant message arrives and with **SONORK_EVENT_CLIENT_USER_SID** when other users change their online status. The application sends messages by with **SrkSendMsg**.

Exiting

Before exiting, the hosts disconnects with **SrkDisconnect** and destroys each instance with **SrkDestroyHandle**.

Event-processing

The callback

Events are received by the host through a callback that must have the following prototype:

```
Sub MyCallback(
     ByVal handle As Long_
   , ByRef SonorkEvent As vbSonorkApiEvent)
```

IMPORTANT: The host must never display modal dialogs nor take "long" (>0.5 seconds) when processing events because it will halt the engine and possibly the application. If the application needs more time, it should store the event's data and display the dialog outside the callback procedure.

Events and parameters

This section describes the meaning of SONORK_EVENT values and interpretation of the **vbSonorkApiEvent** structure passed as parameter to the callback.

The `event` member of the **vbSonorkApiEvent** structure is the event id that defines what the instance is reporting. Events are divided in 3 types: *Client*, *Task* and *File*. To differentiate each, the host application can compare the event's id against the xxx_FIRST and xxx_LAST values of the SONORK_EVENT enum:

CLIENT EVENTS

Notify about changes in the network status, online presence of other users, instant messages, etc. In general, these events are related to the Client-Server link or to internal tasks and may occur at any time during the life of the API instance.
Range: SONORK_EVENT_CLIENT_FIRST~SONORK_EVENT_CLIENT_LAST

TASK EVENTS

These events are in response to tasks initiated by the host, they report status and availability of data. Each task has 2 attributes: 1) A instance-wide unique task id assigned by the API and 2) A task tag which is an arbitrary value defined by the Host when creating the task.
Whenever the API reports a task-related events, it will include these 2 attributes.
Range: SONORK_EVENT_TASK_FIRST~SONORK_EVENT_TASK_LAST

FILE EVENTS

Notify about progress and status of file transfer sessions. As with tasks, file sessions have an instance-wide unique id assigned by the API and an arbitrary host-defined tag. These 2 attributes are included whenever the API reports a file-related event.
Range: SONORK_EVENT_TASK_FIRST~SONORK_EVENT_TASK_LAST

The `appTag` is the tag provided by the host when creating the instance. If the event is a task or file event, the `taskId` and `taskTag` members will be loaded with the task's/file session's id and tag provided by the host when creating the task/file session.
The meaning of the other members of the structure depend on the event's id.

API calls

SrkCreateInstance

Description

Creates an instance of the engine. (Replaces *SrkInit*)

Prototype

```
Function SrkCreateInstance( _
    callbackProc As Long _
  , callbackTag As Long _
  , useUnicode As Long) As Long
```

Parameters

- *callbackProc*: The address of the host's callback (refer to The callback). Use the VB keyword "**AddressOf**" to obtain the address of a procedure.
- *callbackTag*: An arbitrary value defined by the application. This tag is included when reporting events for the instance.
- *useUnicode*: The API will perform unicode translations. Normally, this should be true.

Return values
The instance handle. Zero on failure

Example

```
Const MyTag = 1000
Dim Instance As Long
Instance = SrkCreateInstance(AddressOf MyCallback, MyTag, True)
If Instance = 0 Then
        ' If zero, something is very wrong (the address is invalid)
        ' normally this will never happen: No other action than exiting is possible
        MsgBox "Sonork DLL initialization failed!!"
        Exit Sub
End If
```

SrkDestroyHandle

Description

Destroys an instance, file session or server session.

Prototype

```
Sub SrkDestroyHandle(objectHandle As Long)
```

Parameters

- *objectHandle:* And instance handle, a file session or a server session

Remarks

The callback may be invoked before this call returns.
Replaces **SrkExit** and SrkDestroyFileSession

SrkTimeSlot

Description

Runs a time slot for the object and invokes the callback if there are events pending. The function returns immediately because all long-running tasks are fragmented.

Prototype

```
Function SrkTimeSlot( objectHandle As Long ) As Long
```

Parameters

- *objectHandle:* An instance, file session or server session handle.

Return values
If the object is idle, return value is zero. If there are fragmented tasks still running, it returns non-zero. Host may use the return value to increase the frequency of calls to the method while it returns non-zero.

SrkSetSonorkServer

Description

Specifies the Sonork server to which **SrkConnect** will link. This address is also used by any server sessions.

Prototype

```
Function SrkSetSonorkServer( _
    instance As Long _
  , addressType As SONORK_PHYS_ADDR_TYPE _
  , host As String _
  , port As Long _
  , loginMode As SONORK_LOGIN_MODE) As SONORK_RESULT
```

Parameters

- *instance:* The instance handle.
- *addressType:* The protocol the engine must use. Must be either TCP_1 or UDP_1.
- *host:* The host of the Sonork server. (See remarks section)
- *port:* The port used by the Sonork server for the specified addressType (default: 1503 for UDP, 1504 for TCP)
- *loginMode:* Should be `SONORK_LOGIN_MODE_INTERNET` to connect to the public Internet server (host: server.sonork.com) or when connecting to a private server (Regardless of whether the connection is done over the Internet or not). Trying to connect to the public server with `SONORK_LOGIN_MODE_INTRANET` mode or vice-versa will always fail.

Return value

- On success, the return value is `SONORK_RESULT_OK`.
- On failure the result will be non-zero. Use **SrkGetLastError** to obtain the error information.

Remarks
The host may specify a dotted IP address or a DNS name. However, if a DNS name is supplied the engine will resolve the address synchronously and the application will be blocked while resolution takes place. Solid applications should use asynchronous name resolutions and pass the resolved IP address to the engine to avoid "freezing" until resolution is complete.

SrkConnect

Description

Initiates connection sequence. **SrkSetSonorkServer** must be invoked at least once before calling this function.

Prototype

```
Function SrkConnect( _
    instance As Long
  , userId As vbSonorkDw2
  , password As String) As SONORK_RESULT
```

Parameters

- *instance:* The instance handle.
- *userId:* User account id.
- *password:* User account password.

Return value

- On success, the return value is SONORK_RESULT_OK. The callback will be invoked with event **SONORK_EVENT_CLIENT_STATUS_CHANGE** and status SONORK_NETIO_STATUS_CONNECTING before the function returns. The engine will report progress from within **SrkTimeSlot** with more **SONORK_EVENT_CLIENT_STATUS_CHANGE** events until SONORK_NETIO_STATUS_CONNECTED is reached. Once that status is reached, connection phase has ended and online functions are enabled. The host should then invoke **SrkSetSidMode** so that server notifies other online users about the connection.
- On Failure the callback will not be invoked and the result will be non-zero. Use **SrkGetLastError** to obtain the error information.

SrkDisconnect

Description

Disconnects or aborts a connection sequence.

Prototype

```
Sub SrkDisconnect( objectHandle As Long _
  , invokeHandlers as boolean)
```

Parameters

- *objectHandle:* And instance handle, a file session or a server session
- *invokeHandlers:* Allows invocation of handlers while disconnecting

Remarks
The callback may be invoked before this call returns.
Information set by **SrkSetSonorkServer** is preserved.

SrkSetSidMode

Description

Sets online status (refer to **SONORK_SID_MODE**) (BUSY, AVAILABLE, AWAY, etc.) and message. If the engine is currently connected a global task is initiated to synchronize the mode with server and notify other users about the change. If engine is not connected, the function has no effect.

Prototype

```
Function SrkSetSidMode( _
    instance As Long _
  , sidMode As SONORK_SID_MODE
  , sidText As String) As SONORK_RESULT
```

Parameters

- *instance:* The instance handle.
- *sidMode:* New sid mode.
- *sidText:* Session notes (Explain status. e.g. "I'm in a meeting" or "Out for lunch")

Return value

- On success, the return value is `SONORK_RESULT_OK`. If engine is currently connected a global task will also be started and the callback will be invoked with event **`SONORK_EVENT_CLIENT_GLOBAL_TASK_START`** and task type `SONORK_GLOBAL_TASK_SET_SID` before the function returns. The engine will report task completion (and result) by sending a **`SONORK_EVENT_CLIENT_GLOBAL_TASK_END`** event with the same task type.
- On failure, the callback will not be invoked and the result will be different from `SONORK_RESULT_OK`. Use **`SrkGetLastError`** to obtain the error information.

Remarks

When connected, a `SONORK_RESULT_OK` return value does not guarantee that the sid mode has changed: It only indicates that the global task was successfully started. The host application must process the **`SONORK_EVENT_CLIENT_GLOBAL_TASK_END`** result's to check if the engine was able to synchronize the sid mode with the server.

SrkNormalizeSidMode

Description

Translates a **SONORK_SID_MODE** to one of the primary types. To ensure compatibility with future client/server versions, the host application should always translate sid modes before testing them against one of the primary types.

Prototype

```
Function SrkNormalizeSidMode( _
    sidMode As SONORK_SID_MODE) As SONORK_SID_MODE
```

Parameters

- *sidMode:* Sid mode to translate.

Return value

- The equivalent primary sid mode. If the parameter is already a primary type, the function returns that same value.

SrkRefreshUserList

Description

Downloads the user list from the server. This is a global task.

Prototype

```
Function SrkRefreshUserList(
    instance as Long) As SONORK_RESULT
```

Parameters

- *instance:* The instance handle.

Return value

- On success, the return value is SONORK_RESULT_OK. A global task will also be started and the callback will be invoked with event **SONORK_EVENT_CLIENT_GLOBAL_TASK_START** and task type SONORK_GLOBAL_TASK_REFRESH_USER_LIST before the function returns. The engine will send one **SONORK_EVENT_CLIENT_USER_ADD** event for each user retrieved and completion (with result) will be indicated with a **SONORK_EVENT_CLIENT_GLOBAL_TASK_END** event.

- On failure to initiate the task, the callback will not be invoked and the result will be different from SONORK_RESULT_OK. Use **SrkGetLastError** to obtain the error information.

Remarks

When connected, a SONORK_RESULT_OK return value does not guarantee that the user list was refreshed: It only indicates that the global task was successfully started. The host application must process the **SONORK_EVENT_CLIENT_GLOBAL_TASK_END** result's to check if the engine was able to complete the operation.

SrkRefreshConfig

Description

Downloads configuration items from server. This is a global task.

Prototype

```
Function SrkRefreshConfig(
    instance As Long
  , flags As long) As SONORK_RESULT
```

Parameters

- *instance:* The instance handle.
- flags: A combination of **SONORK_REFRESH_CONFIG_FLAGS**

Return value

- On success, the return value is SONORK_RESULT_OK. A global task will also be started and the callback will be invoked with event **SONORK_EVENT_CLIENT_GLOBAL_TASK_START** and task type SONORK_GLOBAL_TASK_REFRESH_CONFIG before the function returns. Depending on the task type, the engine might send other events before the completion (with result) is indicated with a **SONORK_EVENT_CLIENT_GLOBAL_TASK_END** event.
- On failure to initiate the task, the callback will not be invoked and the result will be different from SONORK_RESULT_OK. Use **SrkGetLastError** to obtain the error information.

Remarks

When reporting **SONORK_EVENT_CLIENT_GLOBAL_TASK_START**, **SONORK_EVENT_CLIENT_GLOBAL_TASK_END** events, data(1) of **vbSonorkApiEvent** (vbSonorkApiEvent.data(1)) contains the flags specified when creating the task.
Most configuration items are valid only while connected and discarded after the engine disconnects (refer to **SONORK_EVENT_CLIENT_STATUS_CHANGE**). Applications that support file transfer should refresh the configuration immediately after connecting with flags SONORK_REFRESH_CONFIG_MAPPINGS and SONORK_REFRESH_CONFIG_SERVICES.

SrkRefreshUser

Description

Refreshes a remote user's data in background. Similar to **SrkGetUserById**.

Prototype

```
Function SrkRefreshUser( _
    instance As Long _
  , userId As vbSonorkDw2 _
  , userInfoLevel As SONORK_USER_INFO_LEVEL
  , getUserFlags As Long
  , authPin As Long) As SONORK_RESULT
```

Parameters

- *instance:* The instance handle.
- *userId:* The user's id.
- *userInfoLevel:* The requested information level (should normally be SONORK_USER_INFO_LEVEL_1)
- getUserFlags: Data requested and method (normally zero), a combination of **SONORK_GET_USER_DATA_FLAGS**
- *authPin:* Authorization pin other user has granted (normally zero)

Return value

- ! On success, the return value is SONORK_RESULT_OK. If the engine is able to retrieve the user's data, a **SONORK_EVENT_CLIENT_USER_REFRESH_DATA** event will be sent before the task end is reported with a **SONORK_EVENT_CLIENT_USER_REFRESH_END** event.
- ! On failure to initiate the task, the callback will not be invoked and the result will be different from SONORK_RESULT_OK. Use **SrkGetLastError** to obtain the error information.

SrkSendMsg

Description

Sends an instant message.

Prototype

```
Function SrkSendMsg( _
    instance As Long _
  , vbLocus As vbSonorkLocus1 _
  , msg As VbSonorkMsg _
  , taskTag As Long) As Long
```

Parameters

- *instance:* The instance handle.
- *vbLocus:* The user location (user id and sid), see remarks section.
- *msg:* The instant message.
- *taskTag:* An arbitrary value associated to the task.

Return value

- On success, the return value is a non-zero task id. The completion (and result) of the task will be reported to the callback by sending a **SONORK_EVENT_TASK_END** event with the proper values loaded in the `taskId` and `taskTag` members of the **vbSonorkApiEvent**.
- On failure to initiate the task, the callback will not be invoked and the function returns zero (0). Use **SrkGetLastError** to obtain the error information.

Remarks
The host application must process the **SONORK_EVENT_TASK_END** result's to check if the engine was able to complete the operation.
Depending on the configuration, the server may allow all zeros in the **sid** member of **vbLocus**. Setting **sid** to zero forces the server to resolve it and/or store the message if user is offline, however this method is not recommended because it is much slower and might not work if the server's administrator has disabled delayed resolution or offline messaging. The recommended way is to store the user's sid value when a **SONORK_EVENT_CLIENT_USER_SID** event is received by the callback (Sid information is included in the event) and then use that value when invoking this function.

SrkGetUserById

Description

Retrieve an user or list of users. `SrkGetUserById` is similar to **`SrkRefreshUser`**: Both return always 0 or 1 records, but the later is done in background. `SrkGetUserByInfo` and **SrkGetUserByCallSign** are used to search the accounts database.

Prototypes

```
Function SrkGetUserById( _
    instance As long _
  , userId As vbSonorkDw2 _
  , userInfoLevel As SONORK_USER_INFO_LEVEL _
  , getUserFlags As long _
  , authPin as long _
  , taskTag as long) As long
```

Parameters

- *instance:* The instance handle.
- *userId:* The user's id.
- *userInfoLevel:* The requested information level (should normally be `SONORK_USER_INFO_LEVEL_1`)
- getUserFlags: Data requested and method (normally zero), a combination of **SONORK_GET_USER_DATA_FLAGS**
- *authPin:* Authorization pin other user has granted (not supported by API, set to zero)
- *taskTag:* An arbitrary value associated to the task.

Return value

- On success, the return value is a non-zero task id. A **SONORK_EVENT_TASK_USER_DATA** event will be sent for each user retrieved, the completion (and result) of the task will be reported to the callback by sending a **SONORK_EVENT_TASK_END** event. For both events, proper values will be loaded in the `taskId` and `taskTag` members of the **`vbSonorkApiEvent`**.
- On failure to initiate the task, the callback will not be invoked and the function returns zero (0). Use **`SrkGetLastError`** to obtain the error information.

Remarks

The authorized list members are notified by the server when the list's owner changes attributes such as online mode (**SONORK_EVENT_CLIENT_USER_SID**) and user data

(**SONORK_EVENT_CLIENT_USER_SET**). A private authorization record is also created; this record allows the host to modify permissions such as status, e-mail and network address visibility on a per-user basis. See **SrkGetUserById** and **SrkRefreshUser**.
After the server accepts and processes the request, it'll notify both source and target entities with a **SONORK_EVENT_CLIENT_USER_AUTH_REQ** event. The request will remain in "pending" state until it is cancelled (by either side) with **SrkDelAuth** or accepted by the target with **SrkAcceptAuth**.
If the user was already in the authorized list, the server silently removes it before sending the notifications (iow: Receiving a **SONORK_EVENT_CLIENT_USER_AUTH_REQ** event implies the user is no longer in the authorized list).

Sonork user lists are symmetric: If user A is in B's authorized list, then B is also in A's authorized list

SrkGetUserByInfo

Description

Retrieve an user or list of users. **SrkGetUserById** is similar to **SrkRefreshUser**: Both return always 0 or 1 records, but the later is done in background. `SrkGetUserByInfo` and **SrkGetUserByCallSign** are used to search the accounts database.

Prototypes

```
Function SrkGetUserByInfo( _
    instance As long _
  , alias As string _
  , name As string _
  , email As String _
  , notes As String _
  , searchFlags As Long _
  , sex As SONORK_SEX _
  , region As vbSonorkDw2 _
  , maxResults As Long _
  , taskTag As Long) As long
```

Parameters

- *instance:* The instance handle.
- *alias, name, email, notes*: Information that the returned list of users must match. Case insensitive. Zero length strings match all records; at least one must be non-empty.
- searchFlags: Combination of **SONORK_SEARCH_USER_FLAGS**
- *sex:* Use `SEX_NA` to include all.
- *maxResults:* Maximum records the server will return. Depending on the server's configuration, this parameter may be adjusted to the maximum allowed for the user requesting the list.
- *taskTag:* An arbitrary value associated to the task.

Return value

- On success, the return value is a non-zero task id. A **SONORK_EVENT_TASK_USER_DATA** event will be sent for each user retrieved, the completion (and result) of the task will be reported to the callback by sending a **SONORK_EVENT_TASK_END** event. For both events, proper values will be loaded in the `taskId` and `taskTag` members of the **vbSonorkApiEvent**.
- On failure to initiate the task, the callback will not be invoked and the function returns zero (0). Use **SrkGetLastError** to obtain the error information.

Remarks

The authorized list members are notified by the server when the list's owner changes attributes such as online mode (**SONORK_EVENT_CLIENT_USER_SID**) and user data (**SONORK_EVENT_CLIENT_USER_SET**). A private authorization record is also created; this record allows the host to modify permissions such as status, e-mail and network address visibility on a per-user basis. See **SrkGetUserById** and **SrkRefreshUser**.

After the server accepts and processes the request, it'll notify both source and target entities with a **SONORK_EVENT_CLIENT_USER_AUTH_REQ** event. The request will remain in "pending" state until it is cancelled (by either side) with **SrkDelAuth** or accepted by the target with **SrkAcceptAuth**.

If the user was already in the authorized list, the server silently removes it before sending the notifications (iow: Receiving a **SONORK_EVENT_CLIENT_USER_AUTH_REQ** event implies the user is no longer in the authorized list).

Sonork user lists are symmetric: If user A is in B's authorized list, then B is also in A's authorized list.

SrkGetUserByCallSign

Description

Retrieve an user or list of users. **SrkGetUserById** is similar to **SrkRefreshUser**: Both return always 0 or 1 records, but the later is done in background. `SrkGetUserByInfo` and `SrkGetUserByCallSign` are used to search the accounts database.

Prototypes

```
Function SrkGetUserByCallSign( _
    instance As long _
  , callSign As string _
  , searchFlags As Long _
  , maxResults As Long _
  , taskTag As Long) As long
```

Parameters

- *instance:* The instance handle.
- *callSign:* Exact name of call sign. Cannot be zero-length. Case insensitive.
- searchFlags: Combination of **SONORK_SEARCH_USER_FLAGS**
- *maxResults:* Maximum records the server will return. Depending on the server's configuration, this parameter may be adjusted to the maximum allowed for the user requesting the list.
- *taskTag:* An arbitrary value associated to the task.

Return value

- On success, the return value is a non-zero task id. A **SONORK_EVENT_TASK_USER_DATA** event will be sent for each user retrieved, the completion (and result) of the task will be reported to the callback by sending a **SONORK_EVENT_TASK_END** event. For both events, proper values will be loaded in the `taskId` and `taskTag` members of the **vbSonorkApiEvent**.
- On failure to initiate the task, the callback will not be invoked and the function returns zero (0). Use **SrkGetLastError** to obtain the error information.

Remarks

The authorized list members are notified by the server when the list's owner changes attributes such as online mode (**SONORK_EVENT_CLIENT_USER_SID**) and user data (**SONORK_EVENT_CLIENT_USER_SET**). A private authorization record is also created; this record allows the host to modify permissions such as status, e-mail and network address visibility on a per-user basis. See **SrkGetUserById** and **SrkRefreshUser**.

After the server accepts and processes the request, it'll notify both source and target entities with a **SONORK_EVENT_CLIENT_USER_AUTH_REQ** event. The request will remain in "pending" state until it is cancelled (by either side) with **SrkDelAuth** or accepted by the target with **SrkAcceptAuth**.
If the user was already in the authorized list, the server silently removes it before sending the notifications (iow: Receiving a **SONORK_EVENT_CLIENT_USER_AUTH_REQ** event implies the user is no longer in the authorized list).
Sonork user lists are symmetric: If user A is in B's authorized list, then B is also in A's authorized list.

SrkRequestAuth

Description

Issues a request to add a member to the local "authorized" user list.

Prototype

```
Function SrkRequestAuth( _
    instance As long _
  , userId As vbSonorkDw2 _
  , userAuth As vbSonorkUserAuth _
  , authNotes As string _
  , taskTag As Long) As long
```

Parameters

- *instance:* The instance handle.
- *userId:* The remote user's id.
- *userAuth:* The authorization record
- *authNotes:* Arbitrary text attached to the request.
- *taskTag*: An arbitrary value associated to the task.

Return value

- On success, the return value is a non-zero task id. The completion (and result) of the task will be reported to the callback by sending a **SONORK_EVENT_TASK_END** event with the proper values loaded in the taskId and taskTag members of the **vbSonorkApiEvent**.
- On failure to initiate the task, the callback will not be invoked and the function returns zero (0). Use **SrkGetLastError** to obtain the error information.

Remarks

The authorized list members are notified by the server when the list's owner changes attributes such as online mode (**SONORK_EVENT_CLIENT_USER_SID**) and user data (**SONORK_EVENT_CLIENT_USER_SET**). A private authorization record is also created; this record allows the host to modify permissions such as status, e-mail and network address visibility on a per-user basis. See **SrkGetUserById** and **SrkRefreshUser**.
After the server accepts and processes the request, it'll notify both source and target entities with a **SONORK_EVENT_CLIENT_USER_AUTH_REQ** event. The request will remain in "pending" state until it is cancelled (by either side) with **SrkDelAuth** or accepted by the target with **SrkAcceptAuth**.
If the user was already in the authorized list, the server silently removes it before sending the notifications (iow: Receiving a **SONORK_EVENT_CLIENT_USER_AUTH_REQ** event implies the user is no longer in the authorized list).

Sonork user lists are simmetric: If user A is in B's authorized list, then B is also in A's authorized list.