

UNIOTP AGENT MANUAL

VERSION 1.1

SecuTech

www.eSecuTech.com

The data and information contained in this document cannot be altered without the express written permission of SecuTech Solution Inc. No part of this document can be reproduced or transmitted for any purpose whatsoever, either by electronic or mechanical means.

The general terms of trade of SecuTech Solution Inc. apply. Diverging agreements must be made in writing.

Copyright © SecuTech Solution Inc. All rights reserved.

WINDOWS is a registered trademark of Microsoft Corporation.

The WINDOWS-logo is a registered trademark ^(TM) of Microsoft Corporation.

Software License

The software and the enclosed documentation are copyright-protected. By installing the software, you agree to the conditions of the licensing agreement.

Licensing Agreement

SecuTech Solution Inc. (SecuTech for short) gives the buyer the simple, exclusive and non-transferable licensing right to use the software on one individual computer or networked computer system (LAN). Copying and any other form of reproduction of the software in full or in part as well as mixing and linking it with others is prohibited. The buyer is authorized to make one single copy of the software as backup. SecuTech reserves the right to change or improve the software without notice or to replace it with a new development. SecuTech is not obliged to inform the buyer of changes, improvements or new developments or to make these available to him. A legally binding promise of certain qualities is not given. SecuTech is not responsible for damage unless it is the result of deliberate action or negligence on the part of SecuTech or its aids and assistants. SecuTech accepts no responsibility of any kind for indirect, accompanying or subsequent damage.

Contact Information

HTTP: www.eSecuTech.com

E-Mail: Sales@eSecuTech.com

Please Email any comments, suggestions or questions regarding this document or our products to us at: Sales@eSecuTech.com

Version	Date
1.0	2011.5.6
1.1	2012.4.4

CE Attestation of Conformity



UniOTP is in conformity with the protection requirements of CE Directives 89/336/EEC Amending Directive 92/31/EEC. UniOTP satisfies the limits and verifying methods: EN55022/CISPR 22 Class B, EN55024: 1998.

FCC Standard



This device is in conformance with Part 15 of the FCC Rules and Regulation for Information Technology Equipment.

Operation of this product is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Conformity to ISO 9001:2000



The Quality System of SecuTech Solution Inc., including its implementation, meets the requirements of the standard ISO 9001:2000

ROHS



All UniOTP products are environmental friendly with ROHS certificates.

Table of Contents

ABOUT THIS GUIDE	1
CHAPTER 1: C/C++	2
1.1 Installation and usage	2
1.2 SECUAUTH()	2
CHAPTER 2: C#	5
2.1 Installation and usage	5
2.2 SECURADCLTS_CLIENT()	5
CHAPTER 3: JAVA	7
3.1 Installation and usage	7
3.2 RADIUSCLIENT()	7
3.3 AUTHENTICATE()	9
3.4 GETCHALLENGEMESSAGE()	10
CHAPTER 4: PHP	11
4.1 Installation and usage	11
4.2 RADIUS()	11
4.3 ACCESSREQUEST()	12
4.4 GETCHALLENGEMSG()	13
4.5 GETLASTERROR()	13
4.6 CLEARLASTERROR()	14

About this guide

The UniOTP Agent SDK manual is intended for use by Developers and Maintenance Engineers who wish to use Agent SDK for developing or building some systems with UniOTP dynamic password authentication. This includes some explanations about how to use Agent SDK and key points.

This guide is divided into sections for each programming language, currently supporting C/C++, C#, Java and PHP.

Each section is separated into:

- Usage
- Function modulation/dependencies

Descriptions of how functions interface/operate is also given.

.

Chapter 1: C/C++

1.1 Installation and usage

1. Please copy the Securepass_auth.h header file and the radclt.lib library file into your project.
2. Please perform authentication through the SecuAuth interface. For detailed instructions, please see the demo project.

1.1.2 How to integrate with the C++ environment

1. Please add the RadiusClt1.h header file and radclt.lib library file to the project.
2. When you need to perform authentication, instantiate the RadiusClt class.
3. Use the initrad member function inside the RadiusClt class to initialize authentication information.
4. Use the “parase” member function inside the RadiusClt class in order to set the username and OTP PIN for authentication. Use the “auth” member function inside the RadiusClt class to terminate the authentication process.

Please refer to the sample for detailed instructions.

1.2 SECUAUTH()

1.2.1 Prototype

```
int secuauth (  
    char *phost,  
    char *pshare,  
    int nsharelen,  
    char *pszusername,  
    int nusernameLen,  
    char *pszpassword,  
    int npasswordlen,  
    int *pnchallenge,  
    int *pbvalidresponse,  
    char *presponse/* = NULL*/,  
    int nmaxbuflen/* = 0*/,  
    unsigned short uport/* = 1812*/,  
    int nwaittime/* = 3*/  
);
```

1.2.2 Parameters

Parameters	Description
<i>Char *phost</i>	[IN] authentication server main address (IP address)
<i>Char *pshare</i>	[IN] shared key used by authentication server and dynamic password token
<i>Int nsharelen</i>	[IN] shared key length
<i>char *pszusername</i>	[IN] The username waiting for Authentication
<i>int nusernameLen</i>	[IN] username length
<i>char *pszpassword</i>	[IN] Password for this time (OTP[PIN] PIN is optional, but it is compulsory to choose a PIN for challenge/response type.)
<i>int npasswordlen</i>	[IN] password length
<i>int *pnchallenge</i>	[OUT] The challenge code returned from the server. If the function return value is RD_ERROR_CHALLENGE_NEEDED, pnchallenge will be used as the challenge code.
<i>int *pbvalidresponse</i>	[OUT] whether Challenge information returned or not (please don't try to set this parameter to NULL)
<i>char *presponse</i>	[OUT] Buffer used for receiving challenge information. If the buffer length is smaller than the challenge information length, challenge information will not be returned. Challenge information length cannot exceed 256 bytes.
<i>int nmaxbuflen</i>	[IN] response buffer maximum size (byte)
<i>unsigned short uport</i>	[IN] Authentication server port number, the default port for standard Radius is 1812.
<i>int nwaittime</i>	[IN] Maximum waiting time (second), if waiting time exceeds this value, the client will stop waiting for server response.

1.2.3 Return Value

Return Value	Description
<i>RD_ERROR_SUCCESS</i>	Authentication succeeded
<i>RD_ERROR_CHALLENGE_NEEDED</i>	The function received successfully the challenge information returned from the server
<i>RD_ERROR_INVALID_USERNAME</i> <i>RD_ERROR_INVALID</i>	Cannot use this username
<i>RD_ERROR_INVALID_PWD</i>	Cannot use this password (wrong format etc.)
<i>RD_ERROR_INVALID_HOST</i>	Cannot use this authentication server (this error is also returned when the IP address field is empty)
<i>RD_ERROR_INVALID_PCNAME</i>	Cannot read Computer name
<i>RD_ERROR_GENRA_FAILED</i>	Failed to generate authenticator request, an error might have occurred when requesting certification package.
<i>RD_ERROR_GENPWD_FAILED</i>	Failed to encrypt the password submitted by the user, this error might have occurred when organizing the packets
<i>RD_ERROR_CREATE_PKT_FAILED</i>	Failed to generate the authentication packets
<i>RD_ERROR_CREATE_SOCKET_FAILED</i>	Socket initialization failed
<i>RD_ERROR_SEND_DATA_FAILED</i>	Failed to send packets
<i>RD_ERROR_RECV_DATA_FAILED</i>	Failed to receive data
<i>RD_ERROR_INVALID_PKT</i>	Data received are invalid
<i>RD_ERROR_NO_RESPONSE</i>	The server doesn't respond
<i>RD_ERROR_AUTH_FAILED</i>	Authentication failed (wrong password, etc.)
<i>RD_ERROR_GET_CHALLENGE_FAILED</i>	The server sent challenge information but an error occurred when receiving data.
<i>RD_ERROR_REMOTE_SOCKET_CLOSED</i>	Server socket closed
<i>RD_ERROR_INIT_SOCKET_FAILED</i>	Under Windows, error when executing WSASocket function

Chapter 2: C#

2.1 Installation and usage

1. Please add the UniOTP_Clt.dll dynamic library to your project.
2. Instantiate the SecuRadClts_Client constructor class in your source code where you want to authenticate the user.
3. Use the “Authenticate” method inside the SecuRadClts_Client class to perform the authentication.

For further details about the source code, please see the sample folder.

SecuRadClts_Client class performs Radius client functions. By calling the member class function interfaces below, you can perform standard Radius authentication.

2.2 SECURADCLTS_CLIENT()

2.2.1 Prototype

```
SecuRadClts_Client (
    string Server,
    string SharedSecret,
    string Username,
    string Password,
    int nport
);
```

2.2.2 Description

Class object instantiation, completes the initialization process

2.2.3 Parameters

Parameters	Description
<i>String Server</i>	Authentication server address
<i>String SharedSecret</i>	Shared key
<i>String Username</i>	Username for the authentication
<i>String Password</i>	Password for the authentication
<i>Int nport</i>	Authentication server port

2.2.4 Return value

Return Value	Description
0	Authentication Succeeded
1	Authentication request failed (impossible to connect to the server or wrong shared key)
2	authentication server response time out
3	returned packets are invalid (length shorter than the packet shortest length)
4	Authentication failed (password error)
5	Invalid packets (Identification code error)
6	packet length error
7	invalid challenge information
8	invalid packets
100	The server sent a challenge

2.2.5 Properties

Property	Type	Meaning	Performing method
Shared Secret	<i>String</i>	Shared Key	<i>Set get</i>
UesrName	<i>String</i>	Username waiting to be authenticated	<i>Set get</i>
Password	<i>String</i>	Authentication password	<i>Set get</i>
Server	<i>String</i>	Authentication server address	<i>Set get</i>
Port	<i>String</i>	Authentication server port (Standard Radius authentication protocol users 1812 for authentication)	<i>Set get</i>
UDPTimeout	<i>String</i>	Server longest waiting time	<i>Set get</i>
pChallenge	<i>String</i>	Server Challenge message	<i>Set get</i>

Chapter 3: Java

3.1 Installation and usage

1. Add the secuotp_jradius.jar file inside your project.
2. When you need to authenticate, Use RadientClient to create a client instance.
3. Use “authenticate” to complete the authentication.

For further details about the source code, please see the sample folder.

3.2 RADIUSCLIENT()

3.2.1 Prototype

```
RadiusClient (  
    String hostname,  
    String sharedSecret,  
    String username  
);
```

3.2.2 Description

Create an object from the RadiusClient class

3.2.3 Parameters

Parameters	Description
<i>String hostname</i>	Authentication server address
<i>String shareSecret</i>	Shared key
<i>String username</i>	Username for user waiting to be authenticated

3.2.4 Prototype 2

```
RadiusClient(  
    String hostname,  
    int authPort,  
    String sharedSecret,  
    String username  
);
```

3.2.5 Parameters

Parameters	Description
<i>String hostname</i>	Authentication server address
<i>Int authPort</i>	Authentication service port (standard Radius authentication port is 1812)
<i>String shareSecret</i>	Shared key
<i>String username</i>	Username for user waiting to be authenticated

3.2.6 Prototype 3

```
RadiusClient(  
    String hostname,  
    int authPort,  
    String sharedSecret,  
    String userName,  
    int sockTimeout  
);
```

3.2.7 Description

Create an object from the RadiusClient class

3.2.8 Parameters

Parameters	Description
<i>String hostname</i>	Authentication server address
<i>Int authPort</i>	Authentication service port (standard Radius authentication port is 1812)
<i>String shareSecret</i>	Shared key
<i>String username</i>	Username for user waiting to be authenticated
<i>Int sockTimeout</i>	Longest waiting time for the server response

3.3 AUTHENTICATE()

3.3.1 Prototype

```
int authenticate(  
    String userPass  
);
```

3.3.2 Description

Perform authentication

3.3.3 Parameters

Parameters	Description
<i>String userPass</i>	User authentication password for this time (OTP only or OTP + PIN)

3.3.4 Return value

Return Value	Description
<i>ACCESS_ACCEPT</i>	Authentication succeeded
<i>ACCESS_REJECT</i>	Authentication failed
<i>ACCESS_CHALLENGE</i>	The server sent challenge information

3.4 GETCHALLENGEMESSAGE()

3.4.1 Prototype

String getChallengeMessage();

3.4.2 Description

When the authentication server sends a challenge, this function is used to retrieve the challenge information returned by the authentication service.

3.4.3 Parameters

N/A

3.4.4 Return value

N/A Challenge information returned by the server

3.4.5 Static data definition

Static data	Value	Meaning
<i>ACCESS_ACCEPT</i>	2	Radius packet type, this packet shows that the authentication succeeded
<i>ACCESS_REJECT</i>	3	Radius packet type, this packet shows that the authentication failed
<i>ACCESS_CHALLENGE</i>	11	Radius packet type, this packet shows that the authentication emitted a challenge

Chapter 4: PHP

4.1 Installation and usage

1. Copy the 2 files, radius.class.php and radius_config.php into the project directory.
2. Change the radius_config.php file contents so that it matches your authentication server parameters.
3. When you need to authenticate, create an instance of Radius class
4. Use the AccessRequest member function to complete the authentication procedure.

For further details about the source code, please see the sample folder.

Class Radius is used for Radius on the client side, by using the following functions of the class, you can get the configuration data from the authentication server, the authentication information configuration data of the authentication user, user ID authentication, the server additional information, etc.

4.2 RADIUS()

4.2.1 *Prototype*

```
Radius(  
    $ip_radius_server = '127.0.0.1',  
    $shared_secret = "",  
    $udp_timeout = 5,  
    $authentication_port = 1812  
);
```

4.2.2 *Description*

Function to create an object from Radius class. Call this function to instantiate Radius class and in the same time configure server address, shared key and longest waiting time for server response,

4.2.3 Parameters

Parameters	Description
<i>\$ip_radius_server</i>	Authentication server address or server domain name, if empty, local computer will be chosen as default.
<i>\$shared_secret</i>	shared key, if empty, no shared key will be used
<i>\$udp_timeout</i>	Server response waiting time, if empty, time out will be 5
<i>\$authentication_port</i>	Authentication server port, if empty, the value will be 1812

4.2.4 Return value

N/A

4.3 ACCESSREQUEST()

4.3.1 Prototype

```
AccessRequest(  
    $username = "",  
    $password = "",  
    $udp_timeout = 0  
);
```

4.3.2 Description

Perform user authentication

4.3.3 Parameters

Parameters	Description
<i>\$username</i>	Username for the user waiting to be authenticated
<i>\$password</i>	Authentication password for this time's authentication, depending on the authentication method, this can either be OTP or OTP+PIN
<i>\$udp_timeout</i>	Longest server response waiting time. If you don't set this parameter, the value will be the default value defined in the class when instantiating.

4.3.4 Return value

Return Value	Description
<i>ACCESS_ACCEPT</i>	Authentication succeeded
<i>ACCESS_REJECT</i>	Authentication failed
<i>ACCESS_CHALLENGE</i>	Challenge sent by the authentication server for this time's authentication

4.4 GETCHALLENGMSG()

4.4.1 Prototype

GetChallengeMsg();

4.4.2 Description

When the AccessRequest function returns ACCESS_CHALLENGE, call this function to get the challenge code returned by the authentication server, which is used for the current ID authentication.

4.4.3 Parameters

N/A

4.4.4 Return Value

Challenge information returned by the authentication server for this time's authentication

4.5 GETLASTERROR()

4.5.1 Prototype

GetLastError()

4.5.2 Description

Get information about last error

4.5.3 Parameters

N/A

4.5.4 Return value

Character string, last abnormal error message

4.6 CLEARLASTERROR()

4.6.1 Prototype

ClearLastError()

4.6.2 Description

Clears error information

4.6.3 Parameters

N/A

4.6.4 Return value

N/A

4.6.5 Static data definition

Static data	Value	Meaning
<i>ACCESS_ACCEPT</i>	2	Radius packet type, this packet shows that the authentication succeeded
<i>ACCESS_REJECT</i>	3	Radius packet type, this packet shows that the authentication failed
<i>ACCESS_CHALLENGE</i>	11	Radius packet type, this packet shows that the authentication emitted a challenge

Follow us!



[Twitter](#)



[Facebook](#)



[Youtube](#)



[Linked in](#)



About SecuTech

SecuTech Solution Inc. is a company specializing in data protection and strong authentication, providing total customer satisfaction in security systems & services for banks, financial institutions & other industries. Having extensive and in-depth experience within the information security market, SecuTech has drawn upon this experience to utilize today's cutting-edge technologies, enables enterprises, financial institutions, and government to safely adopt the economic benefits of mobile and cloud computing that are effective against increasingly sophisticated cyber attacks.



www.eSecuTech.com SecuTech Solution Inc.

North America

1250 Boulevard René-
Lévesque Ouest, #2200,
Montreal, QC, H3B 4W8,
Canada
T: +1 -888-259-5825
F: +1 -888-259-5825 ext.0
E: INFO@eSecuTech.com

China

Level 12, #67 Bei Si Huan
Xi Lu,
Beijing, China, 100080
T: +8610-8288 8834
F: +8610-8288 8834
E: CN@eSecuTech.com

APAC

Suite 5.14, 32 Delhi Rd,
North Ryde,
NSW, 2113, Australia
T: 00612-9888 6185
F: 00612-9888 6185
E: AUS@eSecuTech.com

EMEA

4 Cours Bayard 69002
Lyon, France
T: +33-042-600-2810
F: +33-042-600-2810
M: +33-060-939 6463
E: Europe@eSecuTech.com

©Copyright 2012 SecuTech Solution Inc. All rights reserved. Reproduction in whole or in part without written permission from SecuTech is prohibited. SecuTech UniOTP and the SecuTech logo are trademarks of SecuTech Inc. Windows and all other trademarks are properties of their respective owners. Features and specifications are subject to change without notice.