RSA Authentication Agent 7.0 for Web for Apache Web Server on Red Hat Linux Developer's Guide

The Security Division of EMC

Contact Information

Go to the RSA corporate web site for regional Customer Support telephone and fax numbers**:** **www.rsa.com**

Trademarks

RSA and the RSA logo are registered trademarks of RSA Security Inc. in the United States and/or other countries. For the most up-to-date listing of RSA trademarks, go to **www.rsa.com/legal/trademarks_list.pdf**. EMC is a registered trademark of EMC Corporation. All other goods and/or services mentioned are trademarks of their respective companies.

License agreement

This software and the associated documentation are proprietary and confidential to RSA, are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice below. This software and the documentation, and any copies thereof, may not be provided or otherwise made available to any other person.

No title to or ownership of the software or documentation or any intellectual property rights thereto is hereby transferred. Any unauthorized use or reproduction of this software and the documentation may be subject to civil and/or criminal liability.

This software is subject to change without notice and should not be construed as a commitment by RSA.

Note on encryption technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

Distribution

Limit distribution of this document to trusted personnel.

RSA notice

The RC5™ Block Encryption Algorithm With Data-Dependent Rotations is protected by U.S. Patent #5,724,428 and #5,835,600.

March 2009

Contents

Preface 5
About This Guide 5
Developer Profile 5
RSA Authentication Agent 7.0 for Web for Apache Web Server Documentation 5
Related Documentation 5
Getting Support and Service 6
Before You Call Customer Support 6

Chapter 1: Using the Web Authentication API 7
Sample Source Code 7
Configuring Cookie Expiration Times 7

Chapter 2: Installation and Setup 9
Introduction 9
Setting Up the Environment for the Java Program 9
Setting Up the Environment for the C/C++ Program 10
Setting Up the Environment for the Perl Program 11

Chapter 3: API Information for the Java Environment 13
Constructor 13
RSACookieAPI 13
Methods 14
RSAGetLastError 14
RSAGetShellField 14
RSAGetTagField 16
RSAGetUserName 17
RSAGetCSRFToken 18
RSAGetWebIDURL 19
RSASetTagField 20
RSADeleteTagField 23

Chapter 4: API Functions for the C/C++ Environment 25
RSAGetLastError 25
RSAGetShellField 26
RSAGetTagField 27
RSAGetUserName 29
RSAGetCSRFToken 31
RSAGetWebIDURL 32
RSASetTagField 34
RSADeleteTagField 36
RSAFreeMemory 37

Chapter 5: API Application for the Perl Script Environment 39

Chapter 6: Troubleshooting C/C++ and Perl Programs 41

Getting Third-Party Tag Data From the Cookie 41

Setting Third-Party Tag Data in the Cookie 42

Parameter Settings 43

Agent Parameter Settings 44

BrowserIP Parameter Settings 44

Cookie Parameter Settings 45

szInstance Parameter Settings 45

User Parameter Settings 46

Preface

About This Guide

This guide describes how to develop custom programs using the application programming interfaces (APIs) of RSA Authentication Agent 7.0 for Web for Apache Web Server on Red Hat Linux 4.0, 5.0, and 5.1. It is intended for web developers, system engineers, and other trusted personnel. Do not make this guide available to the general user population.

Developer Profile

Developers must understand the Common Gateway Interface (CGI) environment as defined for use in the C, Java, Perl, or ASP web application development environments.

RSA Authentication Agent 7.0 for Web for Apache Web Server Documentation

For more information about RSA Authentication Agent 7.0 for Web for Apache Web Server (Web Agent), see the following documentation:

***Release Notes*.** Provides information about what is new and changed in this release, as well as workarounds for known issues. The latest version of the *Release Notes* is available from RSA SecurCare Online at **https://knowledge.rsasecurity.com**.

***Installation and Configuration Guide*.** Describes detailed procedures on how to install and configure the Web Agent.

***Developer's Guide*.** Provides information about developing custom programs using the Web Agent application programming interfaces (APIs). Includes an overview of the APIs.

Related Documentation

For more information about products related to RSA Authentication Agent 7.0 for Web for Apache Web Server, see the following:

RSA Authentication Manager documentation set. The full documentation set for RSA Authentication Manager 6.1 is included in the ***InstallPath*\RSA Security\RSA Authentication Manager\doc** directory. The documentation set for RSA Authentication Manager 7.1 is included in the ***InstallPath*\doc** directory.

Getting Support and Service

RSA SecurCare Online	**https://knowledge.rsasecurity.com**
Customer Support Information	**www.rsa.com/support**
RSA Secured Partner Solutions Directory	**www.rsasecured.com**

RSA SecurCare Online offers a knowledgebase that contains answers to common questions and solutions to known problems. It also offers information on new releases, important technical news, and software downloads.

The RSA Secured Partner Solutions Directory provides information about third-party hardware and software products that have been certified to work with RSA products. The directory includes Implementation Guides with step-by-step instructions and other information about interoperation of RSA products with these third-party products.

Before You Call Customer Support

Make sure you have direct access to the computer running the Web Agent software.

Please have the following information available when you call:

- Your RSA Customer/License ID.
- RSA Authentication Agent 7.0 for Web for Apache Web Server software version number.
- The make and model of the machine on which the problem occurs.
- The name and version of the operating system under which the problem occurs.

1 Using the Web Authentication API

- Sample Source Code
- Configuring Cookie Expiration Times

This chapter describes the Web Authentication application programming interface (API). Use this API to add, modify, and delete data in a custom section of the web access authentication browser cookie.

The data is signed by the API as part of the cookie and can be guaranteed against tampering. For privacy, the API also provides a facility to encrypt custom data using the RC5 encryption algorithm.

All API functions described in this document are thread-safe, which means that they can safely be called from multithreaded applications without program failure or data corruption.

Important: United States export regulations impose a limit of six different encrypted custom fields, each of which consists of a tag and its data string. Duplicate tags with the same or different data string do not add to the field count. A maximum of 30 bytes of data can be encrypted in a field. The system returns an error if you exceed these limits.

Sample Source Code

The RSA Authentication Agent 7.0 for Web for Apache Web Server installer includes sample source code that demonstrates the basic calling sequence and usage in the development environments.

The Web Authentication API sample source code is included on your RSA Authentication Agent 7.0 for Web for Apache Web Server installer.

Note: When running CGI scripts, POST data must be 6036 bytes or less.

Configuring Cookie Expiration Times

If you replace a cookie before a customized web access authentication browser cookie expires, the replacement cookie supersedes the customized cookie. As a result, you lose any third-party data that you are setting using the Web Authentication API.

To prevent the loss of third-party data, use the following guidelines to configure your Web Agent cookie expiration times:

- If the expiration time for idle cookies is greater than the overall cookie expiration time, the idle cookie feature becomes invalid, and the cookie is not replaced.

- If the expiration time for idle cookies is less than three minutes and less than the overall cookie expiration time, the cookie is replaced every 30 seconds.
- If the expiration time for idle cookies is greater than three minutes but less than the overall cookie expiration time, the cookie is replaced every 60 seconds.

2 Installation and Setup

- Introduction
- Setting Up the Environment for the Java Program
- Setting Up the Environment for the C/C++ Program
- Setting Up the Environment for the Perl Program

Introduction

You must install and run the Web Authentication API software on the same server that is running the Authentication Agent.

Note: If the web server and the application that runs the Web Authentication API are not run under the same user account, you must add the application user account to the web server's group and change the protection on the **rsawebagent** directory to 710.

Setting Up the Environment for the Java Program

Use the following procedure to set up the environment for the Java program.

To set up the environment for the Java program:

1. Change to the directory you created when you downloaded the software. Untar the **RSACookieAPI** distribution archive.

   ```
   tar -xvf rsacookieapi.tar
   ```

2. Copy the shared library to **/lib**.

   ```
   cd rsacookieapi
   cp librsacookieapi.so /lib
   chmod 755 /lib/librsacookieapi.so.
   ```

   Note: If you are setting up the environment on a 64-bit platform, the shared library directory is **/lib64**.

3. Set the environment variable to **LD_LIBRARY_PATH**. If you use:
   - C-Shell variants (**csh**, **tcsh**), type:

     ```
     setenv LD_LIBRARY_PATH /lib
     ```

   - Bourne shell or a Bourne-compatible shell (such as **sh**, **bash**, **ksh**), type:

     ```
     export LD_LIBRARY_PATH=/lib
     ```

4. Copy the jar file into a directory that is within the classpath.

```
cp RSACookieAPI.jar
serverlet_engine_directory/webapps/jsp-examples
/WEB-INF/lib
```

5. Copy **sample.jsp** to the directory where the servlet engine is installed.

```
cd samples

cp sample.jsp
serverlet_engine_directory/webapps/jsp-examples/
```

Setting Up the Environment for the C/C++ Program

Use the following procedure to set up the environment for the C/C++ program.

Important: These procedures use the Apache web server as an example. Directory locations will vary depending on the web server you are using.

To set up the environment for the C/C++ program:

1. Change to the directory you created when you downloaded the software. Untar the **RSACookieAPI** distribution archive.

```
tar -xvf rsacookieapi.tar
```

2. Copy the shared library to **/lib**.

```
cd rsacookieapi

cp librsacookieapi.so /lib

chmod 755 /lib/librsacookieapi.so.
```

Note: If you are setting up the environment on a 64-bit platform, the shared library directory is **/lib64** .

3. Copy the **cdtexample.cgi** C script to your web server **cgi** directory.

```
cd samples/c

cp cdtexample.cgi /usr/local/apache/cgi-bin

chmod 755 /usr/local/apache/cgi-bin/cdtexample.cgi
```

4. If necessary, create an **images** directory in your web server document root directory.

```
mkdir /usr/local/apache/htdocs/images
```

Note: Some web servers create an **images** directory by default, during installation.

5. Copy the .gif files to the images directory.

```
cd files
```

```
cp *.gif /usr/local/apache/htdocs/images
```

6. Copy **cgd4.htm** to your web server document root directory.

```
cp cgd4.htm /usr/local/apache/htdocs
```

Setting Up the Environment for the Perl Program

Use the following procedure to set up the environment for the Perl program.

Important: These procedures use the Apache web server as an example. Directory locations will vary depending on the web server you are using.

To set up the environment for the Perl program:

1. Change to the directory you created when you downloaded the software. Untar the **RSACookieAPI** distribution archive.

```
tar -xvf rsacookieapi.tar
```

2. Copy the shared library to **/lib**.

```
cd rsacookieapi
cp librsacookieapi.so /lib
chmod 755 /lib/librsacookieapi.so.
```

Note: If you are setting up the environment on a 64-bit platform, the shared library directory is **/lib64**.

3. Copy the Perl scripts to your web server **cgi** directory.

```
cd samples/perl
cp *.pl /usr/local/apache/cgi-bin
chmod 755 /usr/local/apache/cgi-bin/*.pl
```

4. Copy the **rsacookie** executable to your web server **cgi** directory.

```
cp rsacookie /usr/local/apache/cgi-bin
```

5. Copy the **sample.htm** file to your web server document root directory.

```
cp sample.htm /usr/local/apache/htdocs
```

3 API Information for the Java Environment

- RSACookieAPI
- RSAGetLastError
- RSAGetShellField
- RSAGetTagField
- RSAGetUserName
- RSAGetCSRFToken
- RSAGetWebIDURL
- RSASetTagField
- RSADeleteTagField

This chapter describes the functions you can use in a Java development environment.

To use the methods listed in this chapter, you must have set the environment variable **LD_LIBRARY_PATH** to **/lib**.

Note: If you migrate code from your Web Agent 5.3 installation to your Web Agent 7.0 installation, you must reference the **com.rsa.cookieapi** package.

Constructor

RSACookieAPI

Description

```
public RSACookieAPI(HttpServletRequest request)
```

The RSACookieAPI constructor defines an RSACookieAPI object.

You must create an instance of RSACookieAPI. Use one of the following statements:

```
RSACookieAPI rsacookieapi = new RSACookieAPI(request);
```

or

```
RSACookieAPI rsacookieapi;
rsacookieapi = new RSACookieAPI(request);
```

This class can be instantiated only in a Java Server Pages (JSP) web server page or a servlet.

Input Arguments

Argument	Description
Request	HTTPServletRequest object created by the servlet container.

Methods

This section describes the different methods of RSACookieAPI.

Note: The methods of RSACookieAPI expect to obtain the RSA web access authentication cookie from the HttpServletRequest request variable.

RSAGetLastError

Description

```
int RSAGetLastError()
```

The RSAGetLastError method returns the last error code value.

Architecture

This method returns the last error code value. The "Output and Post Conditions" section of each reference page describes the conditions under which the method sets the last error code.

Input Arguments

None required.

Calling or Command Sequence

For examples on how to use this method, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use this method immediately when a method returns to check for error conditions. Subsequent methods overwrite older error codes.

RSAGetShellField

Description

```
String RSAGetShellField()
```

The RSAGetShellField method returns the Default Shell field stored in the web access authentication cookie. The value in this field is the same as the Default Shell field value stored in the RSA Authentication Manager database for the user.

Architecture

This method returns the Default Shell field as a String object. If the returned value is an empty string, the caller can use the RSAGetLastError method of the RSACookie object to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 15.

Input Arguments

None required.

Output and Post Conditions

If the value that is returned by this method is an empty string, the last error code contains one of the following values. Use RSAGetLastError to return the value.

For more information, see "Error Handling" on page 15.

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
101	RSACOOKIE_ERROR_VALID_ COOKIE_NOT_FOUND	The Request argument does not contain a valid RSA cookie.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_ INVALID_ARGUMENT	One of the input arguments is invalid.

Calling or Command Sequence

For examples on how to use this method, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this method at a decision point in the code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError method and take appropriate action.

RSAGetTagField

Description

```
String RSAGetTagField(String tag, int encrypted)
```

The RSAGetTagField method returns a developer-defined field identified by name. The name of the tag is identified by the Tag argument. The field is assumed to have been stored in the web access authentication cookie by a previous call to the RSASetTagField method.

Architecture

This method returns the field data as a String object. If the returned value is an empty string, the caller can use the RSAGetLastError method of the RSACookie object to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 16.

Input Arguments

Argument	Description
Tag	Name of the field to set or replace.
Encrypted	A Boolean flag indicating that the content of the field was encrypted when it was created by the call to the RSASetTagField function. **Note:** This flag is included for backward compatibility with the Web Authentication API used in RSA Web Agent 4.4 for Windows NT, which did not include the embedded 'Encrypted' flag. The RSAGetTagField function ignores this flag.

Output and Post Conditions

If the value that is returned by this method is an empty string, the last error code contains one of the following values. Use RSAGetLastError to return the value. For more information, see "Error Handling" on page 17.

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
101	RSACOOKIE_ERROR_ VALID_COOKIE_NOT_FOUND	The request argument does not contain a valid RSA cookie.
102	RSACOOKIE_ERROR_ DATA_TAG_NOT_FOUND	The cookie is valid but the required tag is not present.

Error	Constant	Description
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_ INVALID_ARGUMENT	One of the input arguments is invalid.
105	RSACOOKIE_ERROR_ CIPHERSUITE_ERROR	The decryption routine failed to decrypt the tag in the cookie.

Calling or Command Sequence

For examples on how to use this method, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this method at a decision point in the code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError method and take appropriate action.

RSAGetUserName

Description

```
String RSAGetUserName()
```

The RSAGetUserName method returns the user name stored in the web access authentication cookie.

Architecture

This method returns the user name as a String object. If the returned value is an empty string, the caller can use the RSAGetLastError method to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 17.

Input Arguments

None required.

Output and Post Conditions

If the value that is returned by this method is an empty string, the last error code contains one of the following values. Use RSAGetLastError to return the value. For more information, see "Error Handling" on page 18.

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.

Error	Constant	Description
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
101	RSACOOKIE_ERROR_ VALID_COOKIE_NOT_FOUND	The request argument does not contain a valid RSA cookie.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_ INVALID_ARGUMENT	One of the input arguments is invalid.

Calling or Command Sequence

For examples on how to use this method, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this method at a decision point in the code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError method and take appropriate action.

RSAGetCSRFToken

Description

```
String RSAGetCSRFToken()
```

The RSAGetCSRFToken method returns the CSRF Token in the Web access authentication cookie.

Architecture

This method returns the CSRF Token as a String object. If the returned value is an empty string, the caller can use the RSAGetLastError method to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 18.

Input Arguments

None required.

Output and Post Conditions

If the value that is returned by this method is an empty string, the last error code contains one of the following values. Use RSAGetLastError to return the value.

For more information, see "Error Handling" on page 19.

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
101	RSACOOKIE_ERROR_VALID_ COOKIE_NOT_FOUND	The Request argument does not contain a valid RSA cookie.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_INVALID_ ARGUMENT	One of the input arguments is invalid.

Calling or Command Sequence

For examples on how to use this method, refer to the sample code included with the Web Agent installer.

The following is an example for Logoff URL with RSArand:

```
<ahref=<%rsacookieapi.RSAGetWebIDURL()%>?logoff?RSArand=<%rsacookieapi.RSAGetCSRFToken()%>>Logoff</a>
```

A hidden field called RSArand is added to the Logoff submission URL. The value of RSArand is checked by the web agent to make sure that the Logoff request is not a result of a CSRF attack, rather, the request is coming from the user who authenticated with the web agent.

Error Handling

To handle errors appropriately, use the value returned by this method at a decision point in the code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError method and take appropriate action.

RSAGetWebIDURL

Description

```
String RSAGetWebIDURL()
```

The RSAGetWebIDURL method returns the *RSA Web Agent URL* for currently accessed RSA protected resource. This URL is required for constructing the correct Logoff URL.

Architecture

This method returns the WebID URL for currently accessed RSA protected resource as a String object. If the returned value is an empty string, the caller can use the RSAGetLastError method of the RSACookie object to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 20.

Input Arguments

None required.

Output and Post Conditions

If the value that is returned by this method is an empty string, the last error code contains one of the following values. Use RSAGetLastError to return the value.

For more information, see "Error Handling" on page 20.

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.

Calling or Command Sequence

For examples on how to use this method, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this method at a decision point in the code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError method and take appropriate action.

RSASetTagField

Description

To set one tag in a cookie or to set the first of multiple tags in the same cookie:

```
String RSASetTagField(String tag, String data, int
encrypted)
```

To set multiple tags in the same cookie, after the initial tag has been set:

```
String RSASetTagField(String newcookie, String tag, String
data, int encrypted)
```

The RSASetTagField method stores the String object passed as the Data argument in the web access authentication cookie. If the name identified by the Tag argument already exists, it will be replaced.

Architecture

This method returns a new cookie string suitable for an HTTP Set-Cookie: header as a String object. You must then set the cookie header in the response object.

Note: U.S. export regulations impose a limit of six different encrypted custom fields (a field consists of a tag and its data string). Duplicate tags with the same or different data string do not add to the count of fields. A maximum of 30 bytes of data can be encrypted in a field. The system returns an error if you exceed these limits.

If the returned value is an empty string, the caller can use the RSAGetLastError method of the RSACookie object to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 22.

Input Arguments

Use the following input arguments when setting one tag in a cookie or the first of multiple tags in the same cookie:

Argument	Description
Tag	Name of the field to set or replace.
Data	The data to set in the field. If binary data must be stored, use any suitable ASCII encoding method to convert the data to a NULL-terminated ANSI string. If the Encrypted flag is set, the maximum is 30 bytes of data. If the Encrypted flag is not set, the only size limitations are those imposed by the browser and available memory.
Encrypted	A Boolean flag indicating that the contents of the field are to be encrypted.

Use the following input arguments when setting subsequent tags in the same cookie.

Argument	Description
Newcookie	Takes the cookie created by the initial RSASetTagField.
Tag	Name of the field to set or replace.
Data	Data to set in the field. If binary data must be stored, use any suitable ASCII encoding method to convert the data to a NULL-terminated ANSI string. If the Encrypted flag is set, the maximum is 30 bytes of data. If the Encrypted flag is not set, the only size limitations are those imposed by the browser and available memory.

Argument	Description
Encrypted	Boolean flag indicating that the contents of the field are to be encrypted.

Output and Post Conditions

If the value that is returned by this method is an empty string, the last error code contains one of the following values. Use RSAGetLastError to return the value. For more information, see "Error Handling" on page 22.

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
101	RSACOOKIE_ERROR_ VALID_COOKIE_NOT_FOUND	The request argument does not contain a valid RSA cookie.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_ INVALID_ARGUMENT	One of the input arguments is invalid.
105	RSACOOKIE_ERROR_ CIPHERSUITE_ERROR	The decryption routine failed to decrypt the tag in the cookie.
106	RSACOOKIE_ERROR_ LONGDATALEN_ENCRYPTION	Attempted to encrypt more than 30 bytes of data in a field.
107	RSACOOKIE_ERROR_ TOOMANY_ENCRYPTED_FIELDS	Attempted to encrypt more than six fields of custom data.

Calling or Command Sequence

For examples on how to use this method, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this method at a decision point in the code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError method and take appropriate action.

RSADeleteTagField

Description

To delete one tag in a cookie or to delete the first of multiple tags in the same cookie:

```
String RSADeleteTagField(String tag)
```

To delete multiple tags in the same cookie, after the initial tag has been deleted:

```
String RSADeleteTagField(String newcookie, String tag)
```

The RSADeleteTagField method deletes a developer-defined field identified by name. The name of the field is given by the Tag argument. The field is assumed to be stored in the web access authentication cookie by a previous call to the RSASetTagField method.

Architecture

This method returns a new cookie string suitable for an HTTP Set-Cookie: header as a String object. You must then set the cookie header in the response object.

If the returned value is an empty string, the caller can use the RSAGetLastError method of the RSACookie object to retrieve one of the defined error codes.

4 API Functions for the C/C++ Environment

- RSAGetLastError
- RSAGetShellField
- RSAGetTagField
- RSAGetUserName
- RSAGetCSRFToken
- RSAGetWebIDURL
- RSASetTagField
- RSADeleteTagField
- RSAFreeMemory

This chapter describes the functions you can use in a C/C++ development environment. If you use:

- C-Shell variants (**csh**, **tcsh**), type:

  ```
  setenv CPPC path_to_C++_compiler make
  ```

- Bourne shell or a Bourne-compatible shell (such as **sh**, **bash**, or **ksh)**, type:

  ```
  CPPC = path_to_C++_compiler make
  ```

To use the API functions, you must include header file **rsacookieapi.h** when you compile and link your CGI C executable with the library file **librsacookieapi.so**, which you previously copied to the **/lib** directory.

To compile a new C CGI API, execute the **build.sh** script in the **/sample** directory.

RSAGetLastError

Description

```
unsigned int RSACOOKIEAPI_API RSAGetLastError(void);
```

The RSAGetLastError function returns the last error code value.

Architecture

This function returns the last error code value. The "Output and Post Conditions" section of each reference page describes the conditions under which the function sets the last error code.

Input Arguments

None.

Calling or Command Sequence

For examples on how to use this function, refer to the sample code included with your Web Agent installer.

Error Handling

To handle errors appropriately, use this function immediately when a function returns to check for error conditions. Subsequent functions overwrite older error codes.

RSAGetShellField

Description

```
LPCSTR RSACOOKIEAPI_API RSAGetShellField(
    LPCSTR szInstance, LPCSTR Cookie,
    LPCSTR User, LPCSTR BrowserIP, LPCSTR Agent);
```

The RSAGetShellField function returns the Default Shell field value stored in the web access authentication cookie. The value in the cookie is the same as the Default Shell field value stored in the RSA Authentication Manager database for the user.

Architecture

This function returns the Default Shell field value as a NULL-terminated string. To contain the string, the function allocates a buffer that must be freed by the caller when the buffer is no longer useful. To free the buffer, your code must pass the buffer to the RSAFreeMemory function.

If the RSAGetShellField function returns a NULL pointer, the caller can use the RSAGetLastError method of the RSACookie object to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 26.

Input Arguments

Argument	Description
szInstance	Value of the CGI variable **SERVER_NAME** unmodified.
Cookie	Value of the CGI variable **HTTP_COOKIE** unmodified.
User	Value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	Value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	Value of the CGI variable **HTTP_USER_AGENT** unmodified.

Output and Post Conditions

This function returns the Default Shell field as a NULL-terminated string and sets the last error code to one of the values in the following table. Use RSAGetLastError to return the value. For more information, see "Error Handling" on page 27.

For more information on field and parameter settings that result in particular error codes, see Chapter 6, "Troubleshooting C/C++ and Perl Programs."

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
101	RSACOOKIE_ERROR_ VALID_COOKIE_NOT_FOUND	The cookie argument does not contain a valid RSA cookie.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_ INVALID_ARGUMENT	One of the input arguments is invalid.

Calling or Command Sequence

For examples on how to use this function, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this function at a decision point in your code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError function and take appropriate action.

RSAGetTagField

Description

```
LPCSTR RSACOOKIEAPI_API RSAGetTagField(
    LPCSTR szInstance, LPCSTR Cookie,
    LPCSTR User, LPCSTR BrowserIP, LPCSTR Agent,
    LPCSTR Tag, BOOL Encrypted);
```

The RSAGetTagField function returns a developer-defined field identified by name. The name of the field is given by the Tag argument. The field is assumed to have been stored in the web access authentication cookie by a previous call to the RSASetTagField function.

Architecture

This function returns the field as a NULL-terminated string. To contain the string, the function allocates a buffer that must be freed by the caller when the buffer is no longer useful. To free the buffer, your code must pass the buffer to the RSAFreeMemory function.

If the RSAGetTagField function returns a NULL pointer, the caller can use the RSAGetLastError method of the RSACookie object to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 28.

Input Arguments

Argument	Description
szInstance	Value of the CGI variable **SERVER_NAME** unmodified.
Cookie	Value of the CGI variable **HTTP_COOKIE** unmodified.
User	Value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	Value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	Value of the CGI variable **HTTP_USER_AGENT** unmodified.
Tag	Name of the field to retrieve.
Encrypted	Boolean flag indicating that the content of the field was encrypted when it was created by the call to the RSASetTagField function. **Note:** This flag is included for backward compatibility with the Web Authentication API used in RSA Web Agent 4.4 for Windows NT, which did not include the embedded 'Encrypted' flag. The RSAGetTagField function ignores this flag.

Output and Post Conditions

This function returns the field as a NULL-terminated string and sets the last error code to one of the values in the following table. Use RSAGetLastError to return the value. For more information, see "Error Handling" on page 29.

For more information on field and parameter settings that result in particular error codes, see Chapter 6, "Troubleshooting C/C++ and Perl Programs."

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
101	RSACOOKIE_ERROR_ VALID_COOKIE_NOT_FOUND	The cookie argument does not contain a valid RSA cookie.
102	RSACOOKIE_ERROR_ DATA_TAG_NOT_FOUND	The cookie is valid, but the required tag is not present.

Error	Constant	Description
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_ INVALID_ARGUMENT	One of the input arguments is invalid.
105	RSACOOKIE_ERROR_ CIPHERSUITE_ERROR	The decryption routine failed to decrypt the tag in the cookie.

Calling or Command Sequence

For examples on how to use this function, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this function at a decision point in your code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError function and take appropriate action.

RSAGetUserName

Description

```
LPCSTR RSACOOKIEAPI_API RSAGetUserName(
    LPCSTR szInstance, LPCSTR Cookie,
    LPCSTR User, LPCSTR BrowserIP, LPCSTR Agent);
```

The RSAGetUserName function returns the user name stored in the web access authentication cookie.

Architecture

This function returns the user name as a NULL-terminated string. To contain the string, the function allocates a buffer that must be freed by the caller when the buffer is no longer useful. To free the buffer, your code must pass the buffer to the RSAFreeMemory function.

If the RSAGetUserName function returns a NULL pointer, the caller can use the RSAGetLastError function to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 30.

Input Arguments

Argument	Description
szInstance	Value of the CGI variable **SERVER_NAME** unmodified.
Cookie	Value of the CGI variable **HTTP_COOKIE** unmodified.

Argument	Description
User	Value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	Value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	Value of the CGI variable **HTTP_USER_AGENT** unmodified.

Output and Post Conditions

This function returns the user name as a NULL-terminated string and sets the last error code to one of the values in the following table. Use RSAGetLastError to return the value. For more information, see "Error Handling" on page 30.

For more information on field and parameter settings that result in particular error codes, see Chapter 6, "Troubleshooting C/C++ and Perl Programs."

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
101	RSACOOKIE_ERROR_ VALID_COOKIE_NOT_FOUND	The cookie argument does not contain a valid RSA cookie.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_ INVALID_ARGUMENT	One of the input arguments is invalid.

Calling or Command Sequence

For examples on how to use this function, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the error value returned by this function at a decision point in your code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError function and take appropriate action.

RSAGetCSRFToken

Description

```
LPCSTR RSACOOKIEAPI_API RSAGetCSRFToken (
     LPCSTR szInstance, LPCSTR Cookie,
     LPCSTR User, LPCSTR BrowserIP, LPCSTR Agent);
```

The RSAGetCSRFToken function returns the CSRF token stored in the web access authentication cookie. This token is required to protect logoff against CSRF attacks.

Architecture

This function returns the CSRF token as a NULL-terminated string. To contain the string, the function allocates a buffer that must be freed by the caller when the buffer is no longer useful. To free the buffer, your code must pass the buffer to the RSAFreeMemory function.

If the RSAGetUserName function returns a NULL pointer, the caller can use the RSAGetLastError method of the RSACookie object to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 31.

Input Arguments

Argument	Description
szInstance	Value of the CGI variable **SERVER_NAME** unmodified.
Cookie	Value of the CGI variable **HTTP_COOKIE** unmodified.
User	Value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	Value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	Value of the CGI variable **HTTP_USER_AGENT** unmodified.

Output and Post Conditions

This function returns the CSRF token as a NULL-terminated string and sets the last error code to one of the values in the following table. Use RSAGetLastError to return the value. For more information, see "Error Handling" on page 32.

For more information on field and parameter settings that result in particular error codes, see Chapter 6, "Troubleshooting C/C++ and Perl Programs."

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.

Error	Constant	Description
101	RSACOOKIE_ERROR_ VALID_COOKIE_NOT_FOUND	The cookie argument does not contain a valid RSA cookie.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_INVALID_ ARGUMENT	One of the input arguments is invalid.

Calling or Command Sequence

For examples on how to use this function, refer to the sample code included with the Web Agent installer.

The following is an example for Logoff URL with RSArand:

```
CSRFToken = RSAGetCSRFToken(ID,
                            Cookie,
                            User,
                            BrowserIP,
                            Agent);
printf("<a href=\"%s?logoff?RSArand=%s\">Logoff</a>",
pWebIDURL, CSRFToken);
```

A hidden field called RSArand is added to the Logoff submission URL. The value of RSArand is checked by the web agent to make sure that the Logoff request is not a result of a CSRF attack, rather the request is coming from the user who authenticated with the web agent.

Error Handling

To handle errors appropriately, use the value returned by this function at a decision point in your code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError function and take appropriate action.

RSAGetWebIDURL

Description

```
LPCSTR RSACOOKIEAPI_API RSAGetWebIDURL(
      LPCSTR szInstance);
```

The RSAGetWebIDURL function returns the *RSA Web Agent URL* for currently accessed RSA protected resource. This URL is required for constructing the correct Logoff URL.

Architecture

This function returns the WebID URL for currently accessed RSA protected resource as a NULL-terminated string. To contain the string, the function allocates a buffer that must be freed by the caller when the buffer is no longer useful. To free the buffer, your code must pass the buffer to the RSAFreeMemory function.

If the RSAGetWebIDURL function returns a NULL pointer, the caller can use the call RSAGetLastError to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 33.

Input Argument

Argument	Description
szInstance	Value of the CGI variable **SERVER_NAME** unmodified.

Output and Post Conditions

This function returns the WebID URL as a NULL-terminated string and sets the last error code to one of the values in the following table. Use RSAGetLastError to return the value. For more information, see "Error Handling" on page 33.

For more information on field and parameter settings that result in particular error codes, see Chapter 6, "Troubleshooting C/C++ and Perl Programs."

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_INVALID_ ARGUMENT	One of the input arguments is invalid.

Calling or Command Sequence

For examples on how to use this function, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this function at a decision point in your code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError function and take appropriate action.

RSASetTagField

Description

```
LPCSTR RSACOOKIEAPI_API RSASetTagField(
    LPCSTR szInstance, LPCSTR Cookie,
    LPCSTR User, LPCSTR BrowserIP, LPCSTR Agent,
    LPCSTR Tag, LPCSTR Data, BOOL Encrypted);
```

The RSASetTagField function stores the NULL-terminated string passed as the Data argument in the web access authentication cookie. If the tag identified by the Tag argument already exists, it is replaced.

Note: If more than one field is to be set, this function can accept the result of a previous call in the Cookie argument.

Architecture

This function returns a new cookie string suitable for an HTTP Set-Cookie: header as a NULL-terminated string. To contain the string, the function allocates a buffer that must be freed by the caller when the buffer is no longer useful. To free the buffer, your code must pass the buffer to the RSAFreeMemory function.

Note: U.S. export regulations impose a limit of six different encrypted custom fields (a field consists of a tag and its data string). Duplicate tags with the same or different data string do not add to the count of fields. A maximum of 30 bytes of data can be encrypted in a field. The system returns an error if you exceed these limits.

If the RSASetTagField function returns a NULL pointer, the caller can use the RSAGetLastError function to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 35.

Input Arguments

Argument	Description
szInstance	Value of the CGI variable **SERVER_NAME** unmodified.
Cookie	Value of the CGI variable **HTTP_COOKIE** unmodified.
User	Value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	Value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	Value of the CGI variable **HTTP_USER_AGENT** unmodified.
Tag	Name of the field to set or replace.

Argument	Description
Data	Data to set in the field. If binary data must be stored, use any suitable ASCII encoding function to convert the data to a NULL-terminated ANSI string. If the Encrypted flag is set, the maximum is 30 bytes of data. If the Encrypted flag is not set, the only size limitations are those imposed by the browser and available memory.
Encrypted	A Boolean flag indicating that the content of the field is to be encrypted.

Output and Post Conditions

This function returns the new cookie string as a NULL-terminated string and sets the last error code to one of the values in the following table. Use RSAGetLastError to return the value. For more information, see "Error Handling" on page 36.

For more information on field and parameter settings that result in particular error codes, see Chapter 6, "Troubleshooting C/C++ and Perl Programs."

Errors	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
101	RSACOOKIE_ERROR_VALID_ COOKIE_NOT_FOUND	The cookie argument does not contain a valid RSA cookie.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_ INVALID_ARGUMENT	One of the input arguments is invalid.
105	RSACOOKIE_ERROR_ CIPHERSUITE_ERROR	The decryption routine failed to decrypt the tag in the cookie.
106	RSACOOKIE_ERROR_ LONGDATALEN_ENCRYPTION	Attempted to encrypt more than 30 bytes of data in a field.
107	RSACOOKIE_ERROR_ TOOMANY_ENCRYPTED_FIELDS	Attempted to encrypt more than six fields of custom data.

Calling or Command Sequence

For examples on how to use this function, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this function at a decision point in your code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError function and take appropriate action.

RSADeleteTagField

Description

```
LPCSTR RSACOOKIEAPI_API RSADeleteTagField(
    LPCSTR szInstance, LPCSTR Cookie,
    LPCSTR User, LPCSTR BrowserIP, LPCSTR Agent,
    LPCSTR Tag);
```

The RSADeleteTagField function deletes a developer-defined field identified by name. The name of the field is given by the Tag argument. The field is assumed to have been stored in the web access authentication cookie by a previous call to the RSASetTagField function.

Note: This function can accept the result of a previous call in the Cookie argument, if more than one field is to be deleted.

Architecture

This function returns a new cookie string suitable for an HTTP Set-Cookie: header as a NULL-terminated string. To contain the string, the function allocates a buffer that must be freed by the caller when the buffer is no longer useful. To free the buffer, your code must pass the buffer to the RSAFreeMemory function.

If the RSADeleteTagField function returns a NULL pointer, use the RSAGetLastError function to retrieve one of the defined error codes. For more information, see "Output and Post Conditions" on page 37.

Input Arguments

Argument	Description
szInstance	Value of the CGI variable **SERVER_NAME** unmodified.
Cookie	Value of the CGI variable **HTTP_COOKIE** unmodified.
User	Value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	Value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	Value of the CGI variable **HTTP_USER_AGENT** unmodified.
Tag	Name of the field to delete.

Output and Post Conditions

This function returns a NULL pointer and sets the last error code to one of the values in the following table. Use RSAGetLastError to return the value. For more information, see "Error Handling" on page 37.

For more information on field and parameter settings that result in particular error codes, see Chapter 6, "Troubleshooting C/C++ and Perl Programs."

Error	Constant	Description
0	RSACOOKIE_ERROR_ NO_ERROR	The operation was successful.
100	RSACOOKIE_ERROR_ CANNOT_ACCESS_SETTINGS	The API library cannot communicate with the web server process to retrieve the necessary information.
101	RSACOOKIE_ERROR_ VALID_COOKIE_NOT_FOUND	The cookie argument does not contain a valid RSA cookie.
102	RSACOOKIE_ERROR_ DATA_TAG_NOT_FOUND	The cookie is valid but the required tag is not present.
103	RSACOOKIE_ERROR_ NOT_ENOUGH_MEMORY	There is not enough memory to perform the requested operation.
104	RSACOOKIE_ERROR_ INVALID_ARGUMENT	One of the input arguments is invalid.

Calling or Command Sequence

For examples on how to use this function, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this function at a decision point in your code. A successful return allows processing to continue. To handle a failure, the code must call the RSAGetLastError function and take appropriate action.

RSAFreeMemory

Description

```
VOID RSACOOKIEAPI_API RSAFreeMemory(LPCSTR Buffer);
```

The RSAFreeMemory function releases memory buffers returned by any of the other C API functions.

Architecture

This function releases the memory returned by the rsacookieapi library. If you use the function with any other type of memory buffer, the program does not run to completion.

Input Arguments

Argument	Description
Buffer	Address of the buffer returned by the previous call.

Calling or Command Sequence

For examples on how to use this function, refer to the sample code included with the Web Agent installer.

Output and Post Conditions

The buffer referenced by the Buffer input argument is no longer valid.

5 API Application for the Perl Script Environment

This chapter describes the application in the Authentication Agent's Web Authentication API that is suitable for use in a Perl script development environment.

The Perl API consists of an applet named rsacookie. You can manually copy this applet from the **samples/web/bin** directory on the RSA Authentication Agent 7.0 for Web for Apache Web Server installer to your web server's **cgi** directory. You can call the applet only in the context of a Perl script running as a CGI application on the web server. Attempting to call this applet directly results in error code 2, indicating that the required CGI variables are missing.

Description of the API Application

```
rsacookie -g –shell
          get contents of shell field
rsacookie -g -tag tag_name
          get data field contents from tag named tag_name
rsacookie -g -e -tag tag_name
          get encrypted data field from tag tag_name
rsacookie -g –user
          get contents of user field
rsacookie -g -CSRFToken
          get CSRF Token for the current session. This token
          is required to protect logoff against CSRF attacks.
rsacookie -g -webIdURL
          get RSA Web Agent URL for currently accessed
          RSA protected resource
rsacookie -s -tag tag_name -data ASCII_data
          set data field contents into tag named tag_name
rsacookie -s -e -tag tag_name -data ASCII_data
          set encrypted data field into tag named tag_name
rsacookie -d -tag tag_name
          delete tag named tag_name
```

Architecture

This application returns the requested data as standard output. If there are any errors, the output is emitted as stderr and the error code sets to indicate the type of failure. To set or delete multiple tags, use setenv to set the HTTP_COOKIE variable to the return value of rsacookie for each call.

Note: U.S. export regulations limit encryption of custom data to six different fields (a field consists of a tag and its data string). Duplicate tags with the same or different data string do not add to the count of fields. A maximum of 30 bytes of data can be encrypted in a field. The system returns an error if you exceed these limits.

Input Arguments

The data supplied as input for the -s switch must follow the -data switch and must be a contiguous ASCII string. To include spaces and non-alphanumeric characters in the string, enclose the string in double quotation marks (“ ”). To include a quotation mark in the string, you must place a backslash directly before the quotation mark (\”).

Output and Post-Conditions

For more information on field and parameter settings that result in particular error codes, see Chapter 6, “Troubleshooting C/C++ and Perl Programs.”

Code	Description
0	Success. The output is the requested data.
1	The supplied arguments are invalid or incorrect.
2	The required CGI variables are missing.
3	The cookie operation failed. The first line of output in stderr contains the detailed error.
4	Attempted to encrypt more than six custom data fields.
5	Attempted to encrypt more than 30 bytes of data in a custom data field.
6	The encryption routine failed to encrypt the tag in the cookie.

Calling or Command Sequence

For examples on how to use this API application, refer to the sample code included with the Web Agent installer.

Error Handling

To handle errors appropriately, use the value returned by this function at a decision point in your code. A successful return allows processing to continue. To handle a failure, your code must examine the value of the error property and take appropriate action.

6 Troubleshooting C/C++ and Perl Programs

- Getting Third-Party Tag Data From the Cookie
- Setting Third-Party Tag Data in the Cookie
- Parameter Settings

This chapter provides information on the error codes returned by Web Authentication API calls depending on field and parameter settings.

Getting Third-Party Tag Data From the Cookie

You can use the following C/C++ API function to get third-party data from the cookie:

```
RSAGetTagField(const char* szInstance,
   const char* Cookie,
   const char* User,
   const char* BrowserIP,
   const char* Agent,
   const char* Tag,
   int Encrypted)
```

You can use the Perl API application with the following argument to get third-party data from the cookie:

```
rsacookie -g -tag tag_name
```

Action	Result (C)	Result (Perl)
Field Tag set to a nonexistent flag.	Error Code 102: RSACOOKIE_ERROR_DATA_TAG_NOT_FOUND	Error Code 3
Field Tag set to an empty string.	Error Code 104: RSACOOKIE_ERROR_INVALID_ARGUMENT	Error Code 3
Field Tag parameter set to the Tag of an unencrypted field. Encrypted parameter set to TRUE.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
Field Tag parameter set to the Tag of an encrypted field. Encrypted parameter set to FALSE.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0

Action	Result (C)	Result (Perl)
Field Tag set to a NULL pointer.	Error Code 104: RSACOOKIE_ERROR_INVALID_ARGUMENT	Error Code 1

Setting Third-Party Tag Data in the Cookie

You can use the following C/C++ API function to set third-party data in the cookie:

```
RSASetTagField(const char* szInstance,
      const char* Cookie,
   const char* User,
   const char* BrowserIP,
   const char* Agent,
   const char* Tag,
   const char* Data,
      int Encrypted)
```

You can use the Perl API function with the following argument to set third-party data in the cookie:

```
rsacookie -s -tag tag_name -data ASCII_data
```

Action	Result (C)	Result (Perl)
Field Tag set to a very long string.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
Field Tag set to an empty string.	Error Code 104: RSACOOKIE_ERROR_INVALID_ARGUMENT	Error Code 3
Field Data set to a very long string.	If process does not run out of memory, Error Code 0: RSACOOKIE_ERROR_NO_ERROR If process runs out of memory, Error Code 103: RSACOOKIE_ERROR_NOT_ENOUGH_MEMORY	Error Code 0 Error Code 3
31-byte (or longer) encrypted Data field added. (Maximum length is 30 bytes.)	Error Code 106: RSACOOKIE_ERROR_LONGDATALEN_ ENCRYPTION	Error Code 5
Seven or more encrypted Data fields added. (Maximum is six encrypted fields.)	Error Code 107: RSACOOKIE_ERROR_TOOMANY_ENCRYPTED_ FIELDS	Error Code 4

Action	Result (C)	Result (Perl)
Field Data set to an empty string.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
Field Data set to a NULL pointer.	Error Code 104: RSACOOKIE_ERROR_INVALID_ARGUMENT	Error Code 1

Parameter Settings

You can use the following C/C++ API functions to set various parameters:

```
RSAGetShellField(const char* szInstance,
    const char* Cookie,
    const char* User,
    const char* BrowserIP,
    const char* Agent)

RSAGetTagField(const char* szInstance,
    const char* Cookie,
    const char* User,
    const char* BrowserIP,
    const char* Agent,
    const char* Tag,
    int Encrypted)

RSAGetUserName(const char* szInstance,
    const char* Cookie,
    const char* User,
    const char* BrowserIP,
    const char* Agent)

RSASetTagField(const char* szInstance,
    const char* Cookie,
    const char* User,
    const char* BrowserIP,
    const char* Agent,
    const char* Tag,
    const char* Data,
    int Encrypted)

RSADeleteTagField(const char* szInstance,
    const char* Cookie,
    const char* User,
    const char* BrowserIP,
    const char* Agent,
    const char* Tag)
```

You can use the Perl API application with the following arguments to set various parameters:

```
rsacookie -g –shell
```

```
rsacookie -g -tag tag_name
rsacookie -g –user
rsacookie -s -tag tag_name -data ASCII_data
rsacookie -d -tag tag_name
```

Agent Parameter Settings

The following table lists the action and results in C and Perl for Agent Parameters.

Action	Result (C)	Result (Perl)
Agent parameter set to a value other than the value of the *HTTP_USER_AGENT* environment variable.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
Agent parameter set to a very long string.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
Agent parameter set to an empty string.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
Agent parameter set to a *NULL* pointer.	Error Code 104: RSACOOKIE_ERROR_INVALID_ARGUMENT	Error Code 1

BrowserIP Parameter Settings

The following table lists the action and results in C and Perl for BrowserIP Parameters.

Action	Result (C)	Result (Perl)
BrowserIP parameter set to a value other than the value of the *REMOTE_ADDR* environment variable.	Error Code 101: RSACOOKIE_ERROR_VALID_COOKIE_NOT_FOUND	Error Code 3
BrowserIP parameter set to a very long string.	Error Code 101: RSACOOKIE_ERROR_VALID_COOKIE_NOT_FOUND	Error Code 3
BrowserIP parameter set to an empty string.	Error Code 101: RSACOOKIE_ERROR_VALID_COOKIE_NOT_FOUND	Error Code 3
BrowserIP parameter set to a *NULL* pointer.	Error Code 104: RSACOOKIE_ERROR_INVALID_ARGUMENT	Error Code 1

Cookie Parameter Settings

The following table lists the action and results in C and Perl for Cookie Parameters.

Action	Result (C)	Result (Perl)
Cookie parameter set to a value other than the value of the *HTTP_COOKIE* environment variable.	Error Code 101: RSACOOKIE_ERROR_VALID_COOKIE_NOT_FOUND	Error Code 3
Cookie parameter set to a very long string.	Error Code 101: RSACOOKIE_ERROR_VALID_COOKIE_NOT_FOUND	Error Code 3
Cookie parameter set to an empty string.	Error Code 101: RSACOOKIE_ERROR_VALID_COOKIE_NOT_FOUND	Error Code 3
Cookie parameter set to a NULL pointer.	Error Code 104: RSACOOKIE_ERROR_INVALID_ARGUMENT	Error Code 1

szInstance Parameter Settings

The following table lists the action and results in C and Perl for szInstance Parameters.

Action	Result (C)	Result (Perl)
szInstance parameter set to a value other than the value of the *SERVER_NAME* environment variable.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
szInstance parameter set to a very long string.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
szInstance parameter set to an empty string.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
szInstance parameter set to a NULL pointer.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0

User Parameter Settings

The following table lists the action and results in C and Perl for User Parameters.

Action	Result (C)	Result (Perl)
User parameter set to a very long string.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
User parameter set to an empty string.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
User parameter set to a value other than the value of the *REMOTE_USER* environment variable.	Error Code 0: RSACOOKIE_ERROR_NO_ERROR	Error Code 0
User parameter set to a NULL pointer.	Error Code 104: RSACOOKIE_ERROR_ INVALID_ARGUMENT	Error Code 1