RSA Authentication Agent 5.3 for Web Authentication Developer’s Guide for Apache Web Server on Red Hat Linux 4.0

RSA®

The Security Division of EMC

Contents

Web Authentication API Overview

The Web Authentication Application Programming Interface (API) lets developers add, modify, and delete data within a custom section of the Web access authentication browser cookie.

The data is signed by the API as part of the cookie, and therefore 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, meaning they can safely be called from multithreaded applications without program failure or data corruption.

Important: U.S. 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 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.

Developer Profile

Previous experience in RSA Authentication Agent development is helpful but not necessary. However, you must understand the CGI environment as defined for use in the C, Java, or Perl web application development environments.

Sample Source Code

The RSA Authentication Agent 5.3 for Web product includes sample source code that demonstrates the basic calling sequence and usage in the development environments. You can run the sample code on Red Hat Linux platforms.

The Web Authentication API sample source code is included in the **rsacookieapi.tar** file included with your Web Agent. After you install the Web Authentication API, you can find the sample code in the **rsacookieapi/samples** directory.

Before You Begin

If a cookie is replaced 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 you are setting using the RSA Authentication Agent Web Authentication API.

To prevent the loss of third-party data, use the following guidelines to configure your Web Agent cookie expiration times so that you have an appropriate window for setting third-party data:

- 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.

Installation and Set Up

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

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

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
   ```

3. Set the environment variable **LD_LIBRARY_PATH**.

   If you use one of the C-Shell variants (**csh**, **tcsh**), type

   ```
   setenv LD_LIBRARY_PATH /lib
   ```

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

   ```
   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

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
   ```

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

   ```
   cd samples/c
   cp cdtexample.cgi /usr/local/apache/cgi-bin
   ```

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

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
   ```

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

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

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

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

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

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

API Information for the Java Environment

This section describes the **com.rsa.cookieapi** package.

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

Note: If you migrate code from your Web Agent 5.2 installation to your Web Agent 5.3 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 JSP web server page or a servlet.

Input Arguments

Table 1:

Request	The HTTPServletRequest object created by the servlet container.

Methods

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 retrieves the last-error code value.

Architecture

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

Input Arguments

None required.

Calling or Command Sequence

For examples of how to use this method, refer to the sample code provided in the **rsacookieapi/samples/jsp** directory.

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 retrieves the **Default Shell** field stored in the Web access authentication cookie. The value in this field is the same as the **Default Shell** 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 return 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 "Outputs and Post Conditions."

Input Arguments

None required.

Outputs and Post Conditions

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

For more information, see "Error Handling."

Table 2:

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 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 of how to use this method, refer to the sample code provided in the **rsacookieapi/samples/jsp** directory.

Error Handling

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

RSAGetTagField

Description

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

The **RSAGetTagField** method retrieves 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 return 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 "Outputs and Post Conditions."

Input Arguments

Table 3:

Tag	The 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.

Outputs and Post Conditions

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

Table 4:

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 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.
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 of how to use this method, refer to the sample code provided in the **rsacookieapi/samples/jsp** directory.

Error Handling

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

RSAGetUserName

Description

```
String RSAGetUserName()
```

The **RSAGetUserName** method retrieves the username stored in the Web access authentication cookie.

Architecture

This method returns the username as a String object. If the return is an empty string, the caller can use the **RSAGetLastError** method to retrieve one of the defined error codes. For more information, see "Outputs and Post Conditions."

Input Arguments

None required.

Outputs and Post Conditions

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

Table 5:

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 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 of how to use this method, refer to the sample code provided in the **rsacookieapi/samples/jsp** directory.

Error Handling

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

Input Arguments

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

Tag	The 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.

Newcookie	Takes the cookie created by the initial RSASetTagField
Tag	The 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.

Outputs and Post Conditions

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

Table 6:

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 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 of how to use this method, refer to the sample code provided in the **rsacookieapi/samples/jsp** directory.

Error Handling

To handle errors appropriately, use the value returned by this method at a decision point in your code. A successful return allows processing to continue. To handle a failure, your code must call the **RSAGetLastError** method and take the 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 have been 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 return 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 "Outputs and Post Conditions."

API Functions for the C/C++ Environment

Functions listed in this section are suitable for use in a C development environment.

Linux does support Gnu compilers, therefore the `-compat` argument is not necessary.

- If you use one of the C-shell variants (**csh**, **tcsh**), issue the following command:

  ```
  setenv CPPC path_to_C++_compiler make -compat
  ```

- If you use the Bourne shell or a Bourne-compatible shell (such as **sh**, **bash**, or **ksh)**, issue the following command:

  ```
  CPPC = path_to_C++_compiler make -compat
  ```

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.

RSAGetLastError

Description

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

The **RSAGetLastError** function retrieves the last error code value.

Architecture

This function returns the last-error code value. The **Ouputs and Post Conditions** section of each reference page notes the conditions under which the function sets the last-error code.

Input Arguments

None.

Calling or Command Sequence

For examples of how to use this function, refer to the sample code provided in the **rsacookieapi/samples/c** directory.

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 retrieves the **Default Shell Field** value stored in the Web access authentication cookie. The value in the cookie is the same as the **Default Shell** 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 call **RSAGetLastError** to retrieve one of the defined error codes. For more information, see "Outputs and Post Conditions."

Input Arguments

Table 7:

szInstance	The value of the CGI variable **SERVER_NAME** unmodified.
Cookie	The value of the CGI variable **HTTP_COOKIE** unmodified.
User	The value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	The value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	The value of the CGI variable **HTTP_USER_AGENT** unmodified.

Outputs 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."

For information on field and parameter settings that result in particular error codes, see "Troubleshooting C/C++ and Perl Programs" on page 38.

Table 8:

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 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 of how to use this function, refer to the sample code provided in the **rsacookieapi/samples/c** directory.

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 call the **RSAGetLastError** function and take the 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 retrieves 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 call **RSAGetLastError** to retrieve one of the defined error codes. For more information, see "Outputs and Post Conditions."

Input Arguments

Table 9:

szInstance	The value of the CGI variable **SERVER_NAME** unmodified.
Cookie	The value of the CGI variable **HTTP_COOKIE** unmodified.
User	The value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	The value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	The value of the CGI variable **HTTP_USER_AGENT** unmodified.
Tag	The name of the field to retrieve.
Encrypted	A Boolean flag indicating that the content of the field was encrypted when it was created by the call to the **RSASetTagField** function.

Outputs 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."

For information on field and parameter settings that result in particular error codes, see "Troubleshooting C/C++ and Perl Programs" on page 38.

Table 10:

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 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.
105	**RSACOOKIE_ERROR_ CIPHERSUITE_ERROR**	The decryption routine failed to decrypt the tag in the cookie.

Calling or Command Sequence

For examples of how to use this function, refer to the sample code provided in the **rsacookieapi/samples/c** directory.

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 call the **RSAGetLastError** function and take the appropriate action.

RSAGetUserName

Description

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

The **RSAGetUserName** function retrieves the username stored in the Web access authentication cookie.

Architecture

This function returns the username 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 "Outputs and Post Conditions."

Input Arguments

Table 11:

szInstance	The value of the CGI variable **SERVER_NAME** unmodified.
Cookie	The value of the CGI variable **HTTP_COOKIE** unmodified.
User	The value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	The value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	The value of the CGI variable **HTTP_USER_AGENT** unmodified.

Outputs and Post Conditions

This function returns the username 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."

For information on field and parameter settings that result in particular error codes, see "Troubleshooting C/C++ and Perl Programs" on page 38.

Table 12:

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 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 of how to use this function, refer to the sample code provided in the **rsacookieapi/samples/c** directory.

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, your code must call the **RSAGetLastError** function and take the 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 argument **Data** in the Web access authentication cookie. If the tag identified by the **Tag** argument already exists it will be replaced.

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

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 "Outputs and Post Conditions."

Input Arguments

Table 13:

szInstance	The value of the CGI variable **SERVER_NAME** unmodified.
Cookie	The value of the CGI variable **HTTP_COOKIE** unmodified.
User	The value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	The value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	The value of the CGI variable **HTTP_USER_AGENT** unmodified.
Tag	The 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 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.

Outputs 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."

For information on field and parameter settings that result in particular error codes, see "Troubleshooting C/C++ and Perl Programs" on page 38.

Table 14:

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 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.

Table 14:

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 of how to use this function, refer to the sample code provided in the **rsacookieapi/samples/c** directory.

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 call the **RSAGetLastError** function and take the 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 "Outputs and Post Conditions."

Input Arguments

Table 15:

szInstance	The value of the CGI variable **SERVER_NAME** unmodified.
Cookie	The value of the CGI variable **HTTP_COOKIE** unmodified.
User	The value of the CGI variable **REMOTE_USER** unmodified.
BrowserIP	The value of the CGI variable **REMOTE_ADDR** unmodified.
Agent	The value of the CGI variable **HTTP_USER_AGENT** unmodified.
Tag	The name of the field to delete.

Outputs 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."

For information on field and parameter settings that result in particular error codes, see "Troubleshooting C/C++ and Perl Programs" on page 38.

Table 16:

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 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 of how to use this function, refer to the sample code provided in the **rsacookieapi/samples/c** directory.

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 call the **RSAGetLastError** function and take the appropriate action.

RSAFreeMemory

Description

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

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

Architecture

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

Input Arguments

Table 17:

Buffer	The address of the buffer returned by the previous call.

Calling or Command Sequence

For examples of how to use this function, refer to the sample code provided in the **rsacookieapi/samples/c** directory.

Outputs and Post Conditions

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

API Application for the Perl Script Environment

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

The Perl API consists of an applet named **rsacookie** that is installed in the web server’s **cgi** directory. The applet can be called only in the context of a Perl script running as a CGI application on the web server. Attempting to call this applet directly will result in error code 2, indicating that the required CGI variables are missing.

RSACookie

Description

```
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 -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. Any errors will result in output being emitted as **stderr** and the error code being set 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 (\”).

Outputs and Post-Conditions

For information on field and parameter settings that result in particular error codes, see "Troubleshooting C/C++ and Perl Programs" on page 38.

Table 18:

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 of how to use this API application, refer to the sample code provided in the **rsacookieapi/samples/perl** directory.

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 the appropriate action.

Troubleshooting C/C++ and Perl Programs

This appendix 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

C/C++ API Calls:

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

Perl API Calls:

```
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
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

C/C++ API Calls:

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

Perl API Call:

```
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	Error Code 0
	If process runs out of memory, Error Code 103: RSACOOKIE_ERROR_NOT_ENOUGH_MEMORY	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
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

C/C++ API Calls:

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

and

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

and

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

and

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

and

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

Perl API Calls:

```
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

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:

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

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

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

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