SELF SERVICE RESET PASSWORD MANAGEMENT

COM OBJECT GUIDE

TOOLS4EVER
IDENTITY GOVERNANCE & ADMINISTRATION

Contents

1. Introduction

Self Service Reset Password Management (SSRPM) supports Microsoft's Component Object Model (COM, see: *Appendix A: Component Object Model (COM)* on page 82 for more information). This document describes how COM is used in combination with SSRPM. Using COM with SSRPM is referred to as SSRPM COM.

SSRPM COM is a part of Self Service Reset Password Management and installed together with the SSRPM Admin Console. SSRPM COM can be used by client applications to access the SSRPM Service via SSRPM COM objects:

Figure 1: SSRPM Service accessed with SSRPM COM via SSRPM COM objects in a client application

SSRPM COM contains several COM objects, of which the main functions are:

- Enroll a user into SSRPM on the SSRPM Service
- Reset a password of an enrolled user on the SSRPM Service

All SSRPM COM objects are contained in a single file: SSRPMCOM.DLL. This DLL is installed with the SSRPM Admin Console. The COM objects are automatically registered on the computer when the SSRPM Admin Console is installed.

2. SSRPM COM object reference

This chapter describes the all COM objects of SSRPM COM, which are:

- SSRPM: the main COM object;
- SSRPMProfile: stores all SSRPM Profile data and settings.

2.1. SSRPM COM object: SSRPM

This is the main SSRPM COM object. The object is used to connect to an SSRPM Service, access SSRPM user data, reset an account or enroll into SSRPM.

2.1.1. SSRPM.Connect

Purpose: Connect to an SSRPM Service.

Description:
This interface method connects to an SSRPM Service. Usually, this is the first method called after the SSRPM COM object has been created. This method always uses IPv4 to establish the connection.

Syntax: *Connect([In] String ServerName, [In] Number PortNumber)*

Parameters:

ServerName:	The name of the computer that runs the SSRPM Service to which the COM object must connect. The name can be specified as DNS or NETBIOS name.
PortNumber:	The port on which the specified SSRPM Service is listening. The default port is `37946`.

Return value: None.

Example (ASP code):

```
Set SSRPM = Server.CreateObject("SSRPMCOM.SSRPM")
SSRPM.Connect("SERVER_A",37946)
```

2.1.2. SSRPM.ConnectEx

Purpose: Connect to an SSRPM Service.

Description:
This interface method connects to an SSRPM Service. Usually, this is the first method called after the SSRPM COM object has been created.

Syntax: *Connect([In] String ServerName, [In] Number PortNumber, [In] Number IPv4Only)*

Parameters:

ServerName:	The name of the computer that runs the SSRPM Service to which the COM object must connect. The name can be specified as DNS or NETBIOS name.
PortNumber:	The port on which the specified SSRPM Service is listening. The default port is `37946`.
IPv4Only	A flag specifying if the connection should be over IPv4. Valid values are 0 and 1, by default the value is 0.

Return value: None.

Example (ASP code):

```
Set SSRPM = Server.CreateObject("SSRPMCOM.SSRPM")
SSRPM.Connect("SERVER_A",37946,0)
```


2.1.3. SSRPM.ConnectMultiple

Purpose: Connect to the first available SSRPM Service.

Description:
This interface method tries to connect to an SSRPM Service. Usually, this is the first method called after the SSRPM COM object has been created. The method always uses IPv4 to establish the connection. This method is used if multiple SSRPM Services are configure fail over.

Syntax: *ConnectMultiple([In] StringArray ServerNames, [In] Number PortNumber)*

Parameters:

ServerNames:	The name of the computers that run the SSRPM Services to which the COM object must connect. The names can be specified as DNS or NETBIOS names.
PortNumber:	The port on which the specified SSRPM Service is listening. The default port is `37946`.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Set SSRPM = Server.CreateObject("SSRPMCOM.SSRPM")
Servers[0] = "ServerA"
Servers[1] = "ServerB"
SSRPM.ConnectMultiple(Servers,37946)
```

2.1.4. SSRPM.ConnectMultipleEx

Purpose: Connect to the first available SSRPM Service.

Description:
This interface method tries to connect to an SSRPM Service. Usually, this is the first method called after the SSRPM COM object has been created. This method is used if multiple SSRPM Services are configure fail over.

Syntax: *ConnectMultipleEx([In] StringArray ServerNames, [In] Number PortNumber, [In] Number IPv4Only)*

Parameters:

ServerNames:	The name of the computers that run the SSRPM Services to which the COM object must connect. The names can be specified as DNS or NETBIOS names.
PortNumber:	The port on which the specified SSRPM Service is listening. The default port is `37946`.
IPv4Only	A flag specifying if the connection should be over IPv4. Valid values are 0 and 1, by default the value is 0.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Set SSRPM = Server.CreateObject("SSRPMCOM.SSRPM")
Servers[0] = "ServerA"
Servers[1] = "ServerB"
SSRPM.ConnectMultipleEx(Servers,37946,0)
```

2.1.5. SSRPM.GetUserData

Purpose: Identify a user and retrieve the applicable SSRPM Profile.

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetUserData interface method retrieves the SSRPM Profile which is applicable for the current user.

Syntax: *GetUserData([In] String Domain, [In] String Account, [In] String Password, [Out] SSRPMProfile Profile)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
Password:	The password of the current user.
Profile:	A newly created SSRPM Profile object. This object will contain the applicable SSRPM Profile of the current user.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Set Profile = Server.CreateObject("SSRPMCOM.SSRPMProfile")
RetVal = SSRPM.GetUserData("CATS","John Smith","#4EdFt6r",Profile)
```

2.1.6. SSRPM.GetResetDataEx

Purpose: Identify a user and retrieve all questions the user must answer.

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetResetData interface method retrieves all questions that a user must answer to eventually reset his or her password.

Syntax: *GetResetData([In] String Domain, [In] String Account, [Out] StringArray QuestionList, [In] SSRPMProfile Profile)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
QuestionList:	An array which will contain all user questions.
Profile:	The SSRPM profile of the user

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim QuestionArray
Set Profile  = Server.CreateObject("SSRPMCOM.SSRPM")
RetVal = SSRPM.GetResetData("CATS","John Smith",QuestionArray, Profile)
```

2.1.7. SSRPM.Enroll

Purpose: Enroll a user into SSRPM.

Description:
The GetUserData interface method (see: *SSRPM.GetUserData* on page 4) must be called first to identify a user which must enroll into SSRPM. The Enroll interface method enrolls a user into SSRPM.

Syntax: *Enroll([In] StringArray QuestionList, [In] StringArray AnswerList)*

Parameters:
QuestionList: An array of user questions.
AnswerList: An array of user answers for all specified user questions.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim QuestionArray(5),AnswerArray(5)

'Fill arrays with specified questions and answers from user

RetVal = SSRPM.Enroll(QuestionArray,AnswerArray)
```

2.1.8. SSRPM.EnrollEx

Purpose: Enroll a user into SSRPM.

Description:
The GetUserData interface method (see: *SSRPM.GetUserData* on page 4) must be called first to identify a user which must enroll into SSRPM. The EnrollEx interface method enrolls a user into SSRPM.

Syntax: *Enroll([In] StringArray QuestionList, [In] StringArray AnswerList, [In] String EmailAddress, [In] String MobilePhoneNumber)*

Parameters:
QuestionList: An array of user questions.
AnswerList: An array of user answers for all specified user questions.
EmailAddress: The email address of the user for advanced authentication
MobilePhoneNumber: The mobile phone number of the user for advanced authentication

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim QuestionArray(5),AnswerArray(5)

'Fill arrays with specified questions and answers from user

RetVal = SSRPM.Enroll(QuestionArray,AnswerArray,EmailAddress,
MobilePhoneNumber)
```

2.1.9. SSRPM.GetResetProfileOptions

Purpose: Get the enabled SSRPM Profile reset options.

Description:
The GetResetData interface method (see: *SSRPM.GetResetData* on page 7) must be called first to identify the user of which the applicable SSRPM Profile reset options must be retrieved.

The GetResetProfileOptions interface method returns the mask number of all enabled SSRPM Profile reset options. This number identifies which settings are enabled within the current SSRPM Profile. Please refer to 'SSRPMProfile.GetOptions for a complete list of available objects.

> Note: The mask number can be used within a logical And-expression to check if a certain SSRPM Profile option has been enabled. This is shown in the example (ASP Code) below, which checks if the 'Show incorrect answer' option has been enabled.

Syntax: *GetResetProfileOptions()*

Parameters: None.

Return value: The mask number of all enabled SSRPM Profile reset options.

Example (ASP code):

```
ResetProfileOptions = SSRPM.GetResetProfileOptions()


If(ResetProfileOptions And 32) Then
    response.write("Incorrect answers may be shown.")
Else
    response.write("Incorrect answers may not be shown.")
End if
```

2.1.10. SSRPM.GetResetData

Purpose: Identify a user and retrieve all questions the user must answer.

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetResetData interface method retrieves all questions that a user must answer to eventually reset his or her password.

Syntax: *GetResetData([In] String Domain, [In] String Account, [Out] StringArray QuestionList)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The a ccount name of the current user.
QuestionList:	An array which will contain all user questions.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim QuestionArray
RetVal = SSRPM.GetResetData("CATS","John Smith",QuestionArray)
```

2.1.11. SSRPM.IsBlocked

Purpose: Verify if a user has been blocked by SSRPM.

Description:
Call this method after a call to SSRPM.GetUserData or SSRPM.GetResetData to check if the user is blocked by SSRPM.

Syntax: *IsBlocked()*

Parameters: None.

Return value: 0 if the user is not blocked, 1 if the user is blocked.

Example (ASP code):

```
If(SSRPM.IsBlocked() = 1)
   response.write("You have been blocked by SSRPM! Please try again later.")
```


2.1.12. SSRPM.IsEnrolled

Purpose: Verify if a user has been enrolled into SSRPM.

Description:
Call this method after a call to SSRPM.GetUserData to check if the user is enrolled into SSRPM.

Syntax: *IsEnrolled([in] String Domain, [in] String Account, [in] String Password)*

Parameters:

Domain:	The domain of the user account.
Account:	The user name.
Password:	The password of the user account.

Return value: 0 if the user is enrolled, non-zero if the user is not enrolled or if the call failed.

Example (ASP code):

```
If(SSRPM.IsEnrolled("CATS","John Smith","#4EdFt6r") = 0)    Then
   response.write("You have been enrolled in SSRPM!")
End If
```

2.1.13. SSRPM.IsEmailAuthenticationRequired

Purpose: Check if email authentication is required for the current user.

Description:
Call this method after a call to SSRPM.GetUserData to check if the user is enrolled into SSRPM.

Syntax: *IsEmailAuthenticationRequired([In] ActionType)*

Parameters:

ActionType:	The action type: 0: Enrollment 1: Reset 2: Unlock The action type for which you are checking if email authentication is required. Because although email authentication might be disabled for a reset action using the web interface, you will still need to provide the email address during an enrollment using the web interface, because you will need the email address if you reset the account using the SSRPM reset wizard.

Return value: 1 if enabled, zero if not enabled and -47 if the call failed.

Example (ASP code):

```
If(SSRPM.IsEmailAuthenticationRequired(1) = 1)
   response.write("You have need to provide an email address to enroll in
SSRPM!")
```

2.1.14. SSRPM.IsSmsAuthenticationRequired

Purpose: Check if SMS authentication is required for the current user.

Description:
Call this method after a call to SSRPM.GetUserData to check if the user is enrolled into SSRPM.

Syntax: *IsSmsAuthenticationRequired([In] ActionType)*

Parameters:

ActionType:	The action type: 0: Enrollment 1: Reset 2: Unlock The action type for which you are checking if email authentication is required. Because although SMS authentication might be disabled for a reset action using the web interface, you will still need to provide the phone number during an enrollment using the web interface, because you will need the phone number if you reset the account using the SSRPM reset wizard.

Return value: 1 if enabled, zero if not enabled and -47 if the call failed.

Example (ASP code):

```
If(SSRPM.IsSmsAuthenticationRequired(1) = 1)

   response.write("You have need to provide an mobile phone number to enroll
in SSRPM!")
```

2.1.15. SSRPM.IsPINCodeAccepted

Purpose: Verify if a PIN code sent by SSRPM is valid.

Description:
Call this method to verify if a PIN code sent with SSRPM.SendAADMessage is valid.

Syntax: *IsPincodeAccepted([In] String Pincode, [In] DeliveryMethod)*

Parameters:

Pincode:	The pincode provided by the user
DeliveryMethod:	The method by which the message was sent. Select 1 for SMS and 2 for email.

Return value: 0 if the PIN code is valid, non-zero if it is not.

Example (ASP code):

```
If(SSRPM.IsPinCodeAccpted("1234", 1) = 0)

   response.write("You have entered the correct PIN code!")
```

2.1.16. SSRPM.GetEmailAddress

Purpose: Get the email address of current user.

Description:
The GetEmailAddress interface method (see: *SSRPM.GetResetData* on page 7) must be called first to identify the user of which the email address must be retrieved.

Syntax: *GetEmailAddress([Out] String EmailAddress)*

Parameters:
EmailAddress: The email address of the current user.

Return value: 0 is successful, non-zero otherwise.

Example (ASP code):

```
RetVal = SSRPM.GetEmailAddress(EmailAddress)


If(RetVal = 0) Then
    response.write("Your email address:" & EmailAddress & "." )
Else
    response.write("An error occurred when getting email address.")
End if
```

2.1.17. SSRPM.GetMobilePhoneNumber

Purpose: Get the mobile phone number address of current user.

Description:
The GetMobilePhoneNumber interface method (see: *SSRPM.GetResetData* on page 7) must be called first to identify the user of which the mobile phone number must be retrieved.

Syntax: *GetMobilePhoneNumber([Out] String MobilePhoneNumber)*

Parameters:
MobilePhoneNumber: The mobile phone number of the current user.

Return value: 0 is successful, non-zero otherwise.

Example (ASP code):

```
RetVal = SSRPM.GetMobilePhoneNumber(MobileNumber)


If(RetVal = 0) Then
    response.write("Your mobile phone number  address:" & MobileNumber & "."
)
Else
    response.write("An error occurred when getting mobile phone number.")
End if
```

2.1.18. SSRPM.CheckAnswers

Purpose: Check answers of a user.

Description:
The GetResetData interface method (see: *SSRPM.GetResetData* on page 7) must be called first to identify the user of which the answers must be checked. The CheckAnswers interface method checks if all questions of a user are answered correctly.

Syntax: *CheckAnswers([In] StringArray QuestionList, [In] StringArray AnswerList, [Out] StringArray InCorrectQuestionList)*

Parameters:

QuestionList:	An array of user questions.
AnswerList:	An array of user answers for all specified user questions.
IncorrectQuestionList:	An array which (optionally) will contain an all questions which are answered incorrectly. This is not the case if the SSRPM Profile option 'Show incorrect answer' has been disabled (see: *SSRPM.GetResetProfileOptions* on page 6) in the applicable SSRPM Profile.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise (-11 when the questions are answered incorrectly).

Example (ASP code):

```
Dim QuestionArray(5),AnswerArray(5),InCorrectAnswerArray

'Fill arrays with questions and specified answers from user

RetVal = SSRPM.CheckAnswers(QuestionArray,AnswerArray,InCorrectQuestionArray)

If (RetVal <> 0) Then
    InCorrectQuestionNr = UBound(InCorrectAnswerArray)

    If(InCorrectQuestionNr <> 0) Then
        response.write(➔
          "You've answered " &InCorrectQuestionNr& " questions invalid.")
    End if
End If
```

2.1.19. SSRPM.ResetAccount

Purpose: Reset a user password.

Description:
The GetResetData interface method (see: *SSRPM.GetResetData* on page 7) must be called first to identify the user of which the password must be reset. The ResetAccount interface method resets the password of a user.

Syntax: *ResetAccount([In] StringArray QuestionList, [In] StringArray AnswerList, [In] String NewPassword, [Out] StringArray InCorrectQuestionList)*

Parameters:

QuestionList:	An array of user questions.
NewPassword:	The new password to which the old password must be set.
AnswerList:	An array of user answers for all specified user questions.
IncorrectQuestionList:	An array which (optionally) will contain an all questions which are answered incorrectly. This is not the case if the SSRPM Profile option 'Show incorrect answer' has been disabled (see: *SSRPM.GetResetProfileOptions* on page 6) in the applicable SSRPM Profile.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise (-11 when the questions are answered incorrectly).

Example (ASP code):

```
Dim QuestionArray(5),AnswerArray(5),InCorrectQuestionArray


'Fill arrays with questions and specified answers from user


RetVal = ➔
SSRPM.ResetAccount(QuestionArray,AnswerArray,"#4EdFt6r",InCorrectQuestionArray)


If (RetVal = 0) Then
    response.write("The password has been reset successfully!")
ElseIf (RetVal = -11) Then
    response.write("The questions are answered incorrectly.")
End If
```

2.1.20. SSRPM.UnlockAccount

Purpose: Unlock a user account.

Description:
The GetResetData interface method (see: *SSRPM.GetResetData* on page 7) must be called first to identify the user of which the account must be unlocked. The UnlockAccount interface method unlocks a user account in the Active Directory.

Syntax: *UnlockAccount([In] StringArray QuestionList, [In] StringArray AnswerList, [Out] StringArray InCorrectQuestionList)*

Parameters:

QuestionList:	An array of user questions.
AnswerList:	An array of user answers for all specified user questions.
IncorrectQuestionList:	An array which (optionally) will contain an all questions which are answered incorrectly. This is not the case if the SSRPM Profile option 'Show incorrect answer' has been disabled (see: *SSRPM.GetResetProfileOptions* on page 6) in the applicable SSRPM Profile.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise (-11 when the questions are answered incorrectly).

Example (ASP code):

```
Dim QuestionArray(5),AnswerArray(5),InCorrectQuestionArray


'Fill arrays with questions and specified answers from user


RetVal = ➔
SSRPM.UnlockAccount(QuestionArray,AnswerArray,InCorrectQuestionArray)


If (RetVal = 0) Then
    response.write("The account has been unlocked successfully!")
Else
    response.write("An error has occurred while unlocking the account.")
End If
```

2.1.21. SSRPM.UnEnroll

Purpose: Unenroll a user from SSRPM.

Description:
The GetUserData interface method (see: *SSRPM.GetUserData* on page 4) must be called first to identify a user which must unenrolled from SSRPM. The Unenroll interface method unenrolls a user from SSRPM.

Syntax: *Unenroll()*

Parameters: None.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
'unenroll selected user
RetVal = SSRPM.UnEnroll()
```

2.1.22. SSRPM.SetUserLanguage

Purpose: Set the language of the user. The language is used for notification emails when an account is reset, enrolled, unlocked or blocked.

Description:

Syntax: S*etUserLanguage()*

Parameters:

LanguageID:	The preferred language of the message. If the language is not available English will be used. The value for English is 9

Return value: 0

Example (ASP code):

```
RetVal = SSRPM.SetUserLanguage(9)
```

2.1.23. SSRPM.SetLogFileLocation

Purpose: Set the location of the log file.

Description:
If the location of the log file is set the COM object will log to that file, assuming it has all the required permissions

Syntax: S*etLogFileLocation()*

Parameters:

LogFileLocation	The path of the log file.

Return value: 0

Example (ASP code):

```
RetVal = SSRPM.SetLogFileLocation("c:\SSRPMComObjectLog.txt")
```

2.1.24. SSRPM.SendAACMessage

Purpose: Send an advanced authentication message to a user.

Description:
Send an advanced authentication message to a user.

Syntax: *SendAACMessage([In] String Domain, [In] String Account, [In] Number DeliveryMethod, [In] MessageActionType, [In] LanguageID)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
DeliveryMethod:	The method by which the message will be sent. Select 1 for SMS and 2 for email.
MessageActionType:	The type of message to send. 1: Send (Pincode) 2: Test 3: Resend(Pincode)
LanguageID:	The preferred language of the message. If the language is not available English will be used. The value for English is 9

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
RetVal = Session("SSRPM").SendAACMessage("CATS", "John Smith", 1, 1, 9)


If (RetVal <> 0) Then
   response.write("Failed to send message.")
End If
```

2.1.25. SSRPM.SendAACMessageTo

Purpose: Send an advanced authentication message to a user.

Description:
Send an advanced authentication message to a user. The SendAACMessageTo interface method allows you to send a message to a user of which the email address or phone number is not known within SSRPM. This can be used to send a test message during the enrollment of a new user.

Syntax: *SendAACMessage([In] String Domain, [In] String Account, [In] Number DeliveryMethod, [In] MessageActionType, [In] LanguageID)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
ToAddress:	The email address to which to test the message.
DeliveryMethod:	The method by which the message will be sent. Select 1 for SMS and 2 for email.
MessageActionType:	The type of message to send. 1: Send (Pincode) 2: Test 3: Resend(Pincode)
LanguageID:	The preferred language of the message. If the language is not available English will be used. The value for English is 9

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise**.**

Example (ASP code):

```
RetVal = Session("SSRPM").SendAACMessage("CATS", "John Smith", "jsmith@cats", 2, 2, 9)


If (RetVal <> 0) Then
   response.write("Failed to send message.")
End If
```

2.1.26. SSRPM.GetPasswordComplexityRules

Purpose: Get the password complexity rules of the domain

Description:
This interface method gets the password complexity rules of the domain from which the enrolled user originates.

Syntax: *GetPasswordComplexityRules([Out] Number MinimumPasswordLength,[Out] Number PasswordComplexity)*

Parameters:

MinimumPasswordLength:	The minimum password length.
PasswordComplexity:	0 if the password complexity rules are disabled, nonzero if they are enabled.

Return value: 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
SSRPM.GetPasswordComplexityRules(MinLength,PasswordComplexity)
```

2.1.27. SSRPM.GetPCMPasswordComplexityRules

Purpose: Get the password complexity rules of the domain from the Password Complexity Manager (PCM).

Description:
This interface method gets the password complexity rules of the domain from which the enrolled user originates.

Syntax: *GetPasswordComplexityRules([In] String Domain, [In] String Account, [Out] StringArray Rules)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
Rules:	An array of string containing the password rules

Return value: 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim PasswordComplexityRules


RetVal = Session("SSRPM").GetPCMPasswordComplexityRules("CATS", "John Smith",
PasswordComplexityRules)
```

2.1.28. SSPRM.GetPCMPasswordComplexityRules

Purpose: Get the password complexity rules of the domain from the Password Complexity Manager (PCM).

Description:
This interface method gets the password complexity rules of the domain from which the enrolled user originates.

Syntax: *GetPasswordComplexityRules([In] String Domain, [In] String Account, [In] Number Language, [Out] StringArray Rules)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
Language:	The language ID in which you want the password rules. English = 9.
Rules:	An array of string containing the password rules

Return value: 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim PasswordComplexityRules


RetVal = Session("SSRPM").GetPCMPasswordComplexityRules("CATS", "John", 9,
PasswordComplexityRules)
```

2.1.29. SSRPM.GetHelpdeskAuthenticationData

Purpose: Get the helpdesk Caller identification data of the specified user.

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetHelpdeskAuthenticationData interface method retrieves the list of questions applicable to the specified user.

Syntax: *GetHelpdeskAuthenticationData([In] String Domain, [In] String Account, [Out] StringArray QuestionList)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
QuestionList:	An array of user questions.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim QuestionList

RetVal = SSRPM.GetHelpdeskAuthenticationData("CATS","John
Smith","#4EdFt6r",Profile)
```

2.1.30. SSRPM.GetHelpdeskAuthenticationDataByCanonicalName

Purpose: Get the helpdesk Caller identification data of the specified user by canonical name.

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetHelpdeskAuthenticationDataByCanonicalName interface method retrieves the list of questions applicable to the specified user.

Syntax: *GetHelpdeskAuthenticationDataByCanonicalName([In] String CanonicalName, [Out] StringArray QuestionList)*

Parameters:

CanonicalName:	The canonical name of the user.
QuestionList:	An array of user questions.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim QuestionList

RetVal = SSRPM.GetHelpdeskAuthenticationDataByCanonicalName("cats.animals/Users/John
Smith", QuestionList)
```

2.1.31. SSRPM.GetHelpdeskGetUserCollection

Purpose: Get users for helpdesk Caller identification that match the criteria (domain and name).

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetHelpdeskGetUserCollection interface method retrieves the list of account names of users that match the domain and resemble the name.

Syntax: *GetHelpdeskGetUserCollection([In] String Domain, [In] String Account, [Out] StringArray AccountNameList)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
AccountNameList:	An array of account names

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim AccountNameList

RetVal = SSRPM.GetHelpdeskGetUserCollection ("CATS","John Smit",AccountNameList)
```

2.1.32. SSRPM.GetHelpdeskRequiredAnswerCharacterIndices

Purpose: Get an array containing the indices of the required characters of the answer to the specified question for the current user.

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetUserData interface method retrieves the list of questions applicable to the specified user.

Syntax: *GetHelpdeskRequiredAnswerCharacterIndices([In] String Questoin, [Out] StringArray RequiredCharIndices)*

Parameters:

Question:	The question
RequiredCharIndices:	The account name of the current user.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim IndicesList

RetVal = SSRPM.GetHelpdeskRequiredAnswerCharacterIndices("What was your childhood nickname?", IndicesList)
```

2.1.33. SSRPM.HelpdeskAuthenticateUser

Purpose: Check answers of a user.

Description:
The GetHelpdeskAuthenticationData interface method (see: *SSRPM.GetResetData* on page 7) must be called first to identify the user of which the answers must be checked. The HelpdeskAuthenticationUser interface method checks if all questions of a user are answered correctly.

Syntax: *HelpdeskAuthenticateUser([In] StringArray QuestionList, [In] StringArray AnswerList, [Out] StringArray InCorrectQuestionList)*

Parameters:

QuestionList:	An array of user questions.
AnswerList:	An array of user answers for all specified user questions.
IncorrectQuestionList:	An array which (optionally) will contain an all questions which are answered incorrectly. This is not the case if the SSRPM Profile option 'Show incorrect answer' has been disabled (see: *SSRPM.GetResetProfileOptions* on page 6) in the applicable SSRPM Profile.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise (-11 when the questions are answered incorrectly).

Example (ASP code):

```
Dim QuestionArray(5),AnswerArray(5),InCorrectAnswerArray


'Fill arrays with questions and specified answers from user


RetVal =
SSRPM.HelpdeskAuthenticateUser(QuestionArray,AnswerArray,InCorrectQuestionArray)


If (RetVal <> 0) Then
    InCorrectQuestionNr = UBound(InCorrectAnswerArray)


    If(InCorrectQuestionNr <> 0) Then
         response.write(➔
           "You've answered " &InCorrectQuestionNr& " questions invalid.")
    End if
End If
```

2.1.34. SSRPM.HelpdeskVerifyUserInput

Purpose: Check answers of a user.

Description:
The GetHelpdeskAuthenticationData or GetHelpdeskAuthenticationDataByCanonicalName methods must be called first to identify the user of which the answers must be checked. The HelpdeskVerifyUserInput interface method checks if all questions of a user are answered correctly. It returns

Syntax: *HelpdeskVerifyUserInput([In] StringArray QuestionList, [In] StringArray AnswerList, [Out] IncorrectCharacterIndexListPerQuestions)*

Parameters:

QuestionList:	An array of user questions.
AnswerList:	An array of user answers for all specified user questions.
IncorrectCharacterIndexListPerQuestions:	An array which contains all an array with indices of the characters of the answers which are answered incorrectly.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim QuestionArray(5),AnswerArray(5),IncorrectCharacterIndexListPerQuestions
MaxAnswerLength = 15


RetVal = SSRPM.HelpdeskVerifyUserInput(QuestionArray,AnswerArray,
IncorrectCharsIndicesPerQ)


If (RetVal <> 0) Then
    NrOfQuestions= UBound(IncorrectCharsIndicesPerQ)
    For(QuestionNr = 0 To NrOfQuestions
      MaxChars = UBound(IncorrectCharsIndicesPerQ, 1)
      For(CharIndex = 0 To MaxChars
          If(Not Empty(IncorrectCharsIndicesPerQ(QuestionNr, CharIndex)) )
Then
            response.write("You've the character at position " & CharIndex &
" of the answer to question " & QuestionNr &" is incorrect.")
          End if
      Next
    Next
End If
```

2.1.35. SSRPM.ValidatePhoneNumber

Purpose: Checks if the phone number matches the requirements, i.e. is a valid phone number.

Description:
Call this method to validate the phone number.

Syntax: ValidatePhoneNumber*([In] String PhoneNumber)*

Parameters:
PhoneNumber: The phone number that will be validated.

Return value: 0 if the phone number is valid, -36 if the phone number is invalid.

Example (ASP code):

```
If(SSRPM.ValidatePhoneNumber("1234") = 0)
   response.write("You have entered a valid phone number!")
```

2.1.36. SSRPM.ValidateEmailAddress

Purpose: Checks if the email address matches the requirements, i.e. is a valid email address and is not excluded for email authentication.

Description:
Call this method to validate the phone number.

Syntax: ValidateEmailAddress*([In] String EmailAddress)*

Parameters:
EmailAddress: The email address that will be validated.

Return value: 0 if the email address is valid, -37 if the email address is invalid and -47 if the email address is excluded for email authentication.

Example (ASP code):

```
If(SSRPM.ValidateEmailAddress("person@tools4ever.com") = 0)
   response.write("You have entered a invalid email address!")
```

2.1.37. SSRPM.SetClientIpAddress

Purpose: Configures the COM object to use the specified IP address as the source IP in the operations log.

Description:
Call this method to set the client IP address in the COM object.

Syntax: SetClientIpAddress*([In] String IpAddress)*

Parameters:

IpAddress:	The IP address that will be used in the operation log.

Return value: 0

Example (ASP code):

```
SSRPM.SetClientIpAddress("127.0.0.1")
```

2.1.38. SSRPM.ChangeAccountPassword

Purpose: Change the password of the specified account.

Description:
Call this method to change the password of a specified account.

Syntax: ChangeAccountPassword*([In] String Domain, [In] String Account, [In] String OldPassword, [In] String NewPassword)*

Parameters:

Domain:	The domain name
Account:	The username of the account
OldPassword:	The current password
NewPassword:	The new password

Return value: 0

Example (ASP code):

```
SSRPM.ChangeAccountPassword("MyDomain", "MyAccount", "MyOldPassword",
"MyNewPassword")
```

2.1.39. SSRPM.GetPasswordComplexityRulesEx

Purpose: Get the password complexity rules of the domain

Description:
This interface method gets the password complexity rules of the domain from which the enrolled user originates.

Syntax: *GetPasswordComplexityRulesEx([In] String Domain, [In] String Account, [Out] Number MinimumPasswordLength,[Out] Number PasswordComplexity)*

Parameters:

Domain:	The domain name
Account:	The username of the account
MinimumPasswordLength:	The minimum password length.
PasswordComplexity:	0 if the password complexity rules are disabled, nonzero if they are enabled.

Return value: 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
SSRPM.GetPasswordComplexityRulesEx("MyDomain", "MyAccount",
MinLength,PasswordComplexity)
```

2.1.40. SSRPM.GetPasswordComplexityRulesMethodOfAccount

Purpose: Get the password complexity rules method of the specified account, if any exists. If the user doesn't fall under the scope of a profile, it will return 0.

Description:
Call this method to determine which password complexity rules should be shown: none, Windows or PCM.

Syntax: GetProfileOptionsOfAccount*([In] String Domain, [In] String Account, [In] String Password, [Out] Number PasswordComplexityRulesMethod)*

Parameters:

Domain:	The domain name
Account:	The username of the account
Password:	The current password
PasswordComplexityRulesMethod:	The password complexity rules method of the specified account. 0 - Unknown/None 1 - Windows 2 - PCM

Return value: 0

Example (ASP code):

```
Dim ProfileOptions

SSRPM.GetProfileOptionsOfAccount("MyDomain", "MyUser", "MyPassword",
PasswordComplexityRulesMethod)
```

2.1.41. SSRPM.IsAscEnabled

Purpose: Return 1 if advanced sequence configuration is enabled for the current user, otherwise it will return 0.

Description:
Call this method to determine if the advanced sequence configuration option is enabled.

Syntax: IsAscEnabled*()*

Return value: 0 or 1

Example (ASP code):

```
If(SSRPM.IsAscEnabled()) Then
   response.write("ASC enabled!")
End If
```

2.1.42. SSRPM.GetAscSequenceOptions

Purpose: Get an array with all the available sequences as flags.

Description:
Call this method to get an array with all the available sequences as flags for the current user. This is used to be able present the user with possible authentication paths.

Syntax: GetAscSequenceOptions*([Out] Array Sequences)*

Parameters:

Sequences: An array with flag values representing the authentication methods in the sequence.
Flag values:

- 0: Invalid entry
- 1: Administrator-defined questions
- 2: User-defined questions
- 4: Advanced Authentication

Possible values in the array: 3, 5, 6

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim AscSequenceOptions
Retval = Session("SSRPM").GetAscSequenceOptions(AscSequenceOptions)
```

2.1.43. SSRPM.AscNextPage

Purpose: Get the next page in the ASC sequence.

Description:
Call this method to get the next page in the specifed sequence.

Syntax: AscNextPage*([in] Number Procedure, [in] Number Sequence, [out] String Page)*

Parameters:

Procedure:	A number representing the current procedure, reset or unlock. Possible values: 1: Enroll (currently not used) 2: Unlock 3: Reset
Sequence:	The flag value representing the authentication methods in the selected sequence. See GetAscSequenceOptions for details.
Page:	The file name of the next page.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Retval = Session("SSRPM").AscNextPage(3, Session("AscSequence"), Page)
```

2.1.44. SSRPM.AscPrevPage

Purpose: Get the previous page in the ASC sequence.

Description:
Call this method to get the previous page in the specifed sequence.

Syntax: AscPrevPage*([out] String Page)*

Parameters:
Page: The file name of the previous page.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Retval = Session("SSRPM").AscPrevPage(Page)
```

2.1.45. SSRPM.AscGetCurrentDefinitionID

Purpose: Get the flag value of the current page in the sequence.

Description:
Call this method to get the flag value of the current page in the sequence.

Syntax: AscGetCurrentDefinitionID*()*

Return value: Number.
Possible values:

0: Invalid entry

1: Administrator-defined questions

2: User-defined questions

4: Advanced Authentication

8192: Starting page in sequence

Example (ASP code):

```
If(Session("SSRPM").AscGetCurrentDefinitionID() = 2) Then
      Response.Write("User-defined questions!")
End If
```

2.1.46. SSRPM.AscAddPageToHistory

Purpose: Add the page to the ASC history sequence.

Description:
Call this method to add the page to the history of the ASC sequence. Normally this already handled by AscNextPage, this is only necessary for pages that add an additional step in the process, such as the option to choose between SMS or email authentication.

Syntax: AscAddPageToHistory*([in] Number Procedure, [in] Number DefinitionID)*

Parameters:

Page:	▪ The name of the page.
DefinitionID:	The flag value representing the authentication methods in the selected sequence. See AscGetCurrentDefinitionID for more details.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
RetVal = Session("SSRPM").AscAddPageToHistory("Identify.asp", 8192)
```

2.1.47. SSRPM.AscSetUserResultSuccessForCurrentPage

Purpose: Indicates that the authentication for the current method was successful.

Description:
Call this method to indicate that the authentication for this method was successful. This is only for navigation purposes, during a reset or unlock the authentication methods are checked again on the service.

Syntax: AscSetUserResultSuccessForCurrentPage()

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
RetVal = Session("SSRPM").AscSetUserResultSuccessForCurrentPage()
```

2.1.48. SSRPM.AscGetQuestionsOfCurrentPage

Purpose: Get the questions of applicable to the current authentication method.

Description:
Call this method to get the questions of applicable to the current authentication method, so either administrator- or user-defined questions.

Syntax: AscGetQuestionsOfCurrentPage*([in] Number Procedure, [in] Number DefinitionID)*

Parameters:
QuestionList: An array which will contain the user's questions applicable to the current page.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim FilteredQuestionList

RetVal = Session("SSRPM").AscGetQuestionsOfCurrentPage(FilteredQuestionList)
```

2.1.49. SSRPM.PCMTestPassword

Purpose: Test the specified password against the password complexity rules specifed in the Password Complexity Manager (PCM).

Description:
This interface method tests the specified password against the password complexity rules defined in PCM that apply to the specified user.

Syntax: *PCMTestPassword([In] String Domain, [In] String Account, [In] String NewPassword, [In] String OldPassword, [In] Number Language, [In] Number Options, [Out] StringArray Rules, [Out] BooleanArray ValidRules)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
NewPassword:	The new password of the user.
OldPassword	The current password of the user.
Language:	The language ID in which you want the password rules. English = 9.
Options:	A flag value indicating what type of rules should be tested. Possible values: 0: All 1: Ignore password similarity (use this if the current password is unknown).
Rules:	An array of string containing the password rules.
ValidRules:	An array of boolean values indicating if the password matches a rule, the index of the rul in the *ValidRules* parameter matches the index of the rules in the parameter *Rules.*

Return value: 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim PasswordComplexityRules

Dim ValidRules

RetVal = PcmTestPassword("CATS", "John", "Welcome123", "Secret123", 9, 1,
PasswordComplexityRules, ValidRules)
```

2.1.50. SSRPM.GetEmailAddressObscured

Purpose: Get the obscured email address of current user, where obscured means that only the first 3 digits* of the local part (before the '@' sign) of the e-mail address are visible and the entire domain part (after the '@'-sign) is visible.
*If the local part is less than 6 characters, the amount of visible characters will also be reduced.

Description:
The GetEmailAddress interface method (see: *SSRPM.GetResetData* on page 7) must be called first to identify the user of which the email address must be retrieved.

Syntax: *GetEmailAddressObscured([Out] String EmailAddress)*

Parameters:
EmailAddress: The email address of the current user.

Return value: 0 is successful, -55 if this option has been disabled and non-zero otherwise.

Example (ASP code):

```
RetVal = SSRPM.GetEmailAddressObscured(EmailAddress)


If(RetVal = 0) Then
    response.write("Your email address:" & EmailAddress & "." )
Else
    response.write("An error occurred when getting email address.")
End if
```


2.1.51. SSRPM.GetMobilePhoneNumberObscured

Purpose: Get the obscured mobile phone number address of current user, where obscured means that only the last 3 digits are visible.

Description:
The GetMobilePhoneNumberObscured interface method (see: *SSRPM.GetResetData* on page 7) must be called first to identify the user of which the mobile number must be retrieved.

Syntax: *GetMobilePhoneNumberObscured([Out] String MobilePhoneNumber)*

Parameters:
MobilePhoneNumber: The mobile phone number of the current user.

Return value: 0 is successful, -55 if this option has been disabled and non-zero otherwise.

Example (ASP code):

```
RetVal = SSRPM.GetMobilePhoneNumberObscured(MobileNumber)


If(RetVal = 0) Then
    response.write("Your mobile phone number  address:" & MobileNumber & "."
)
Else
    response.write("An error occurred when getting mobile phone number.")
End if
```

2.1.52. SSRPM.AscSetSelectedAscSequence

Purpose: Configures the COM object to use the selected authentication sequence selected by the user.

Description:
Call this method to set the select authentication sequence in the COM object.

Syntax: AscSetSelectedAscSequence*([In] Long AscSequence)*

Parameters:
AscSequence: The flag value representing the selected authentication sequence.

Return value: 0

Example (ASP code):

```
AscSequence = Request.Form("AscOption")
SSRPM.AscSequence(AscSequence)
```

2.1.53. SSRPM.AscNextStep

Purpose: Get the ID of the next step in the ASC sequence.

Description:
Call this method to get the ID of the next step in the specifed sequence.

Syntax: AscNextStep*([in] Number Sequence, [in] Number StepId, [in] Number DefinitionID)*

Parameters:

Sequence:	The flag value representing the authentication methods in the selected sequence. See GetAscSequenceOptions for details.
StepId:	The ID of the next step. Possible options are: 0: IDENTIFY 1: ASCSELECTSEQUENCE 2: AACEMAIL 3: AACSMS, 4: AACSELECTMETHOD 5: QUESTIONANSWER 6: RESETORUNLOCK 7: GETPASSWORD 8: UNLOCK

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Retval = Session("SSRPM").AscNextStep(3, StepId)
```

2.1.54. SSRPM.AscPrevStep

Purpose: Get the ID of the previous step in the ASC sequence.

Description:
Call this method to get the ID of the previous step in the sequence.

Syntax: AscPrevStep*()*

Return value: The ID of the previous step.

Example (ASP code):

```
StepId = Session("SSRPM").AscPrevStep()
```

2.1.55. SSRPM.AscAddStepIDToHistory

Purpose: Add the specified step to the ASC history sequence.

Description:
Call this method to add the page to the history of the ASC sequence. Normally this already handled by AscNextStep, this is only necessary for pages that add an additional step in the process, such as the option to choose between SMS or email authentication.

Syntax: AscAddStepIDToHistory*([in] Number Procedure, [in] Number DefinitionID)*

Parameters:

StepId:	▪ The ID of the step.
DefinitionID:	The flag value representing the authentication methods in the selected sequence. See AscGetCurrentDefinitionID for more details.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
RetVal = Session("SSRPM").AscAddStepIDToHistory(StepId, 8192)
```

2.1.56. SSRPM.AscGetQuestionTypeCount

Purpose: Get the number of questions per type for the current user.

Description:
Call this method to get the number questions per type for the current user, so number of administrator- or user-defined questions.

Syntax: AscGetQuestionTypeCount*([in] Number NrOfAdminQuestions, [in] Number NrOfUserQuestions*

Parameters:
NrOfAdminQuestions: The number of administrative questions for the user.
NrOfUserQuestions: The number of user-defined questions for the user.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
RetVal = Session("SSRPM").AscGetQuestionTypeCount(NrOfAdminQuestions,
NrOfUserQuestions)
```

2.1.57. SSRPM.GetTrustedDomains

Purpose: Get the trusted domains available from the SSRPM service.

Description:
Call this method to get a list of domain names available from the SSRPM service. This can be used to popuplate a dropdown menu.

Syntax: GetTrustedDomains*([In] String Domains)*

Parameters:
TrustedDomains: The trusted domains.

Return value: 0

Example (ASP code):

```
Dim TrustedDomainList

SSRPM.GetTrustedDomains(TrustedDomainList)
```


2.1.58. SSRPM.GetUserDataOfCurrentUser

Purpose: Loads the applicable SSRPM Profile and data of the currently logged on user.

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetUserDataOfCurrentUser interface method retrieves the SSRPM Profile which is applicable for the user under whose credentials this call is made.

Syntax: GetUserDataOfCurrentUser*([Out] IUnknown *IProfile)*

Parameters:

IProfile:	A interface to the current profile created by the SSRPM COM object. This object will contain the applicable SSRPM Profile of the current user.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Retval = Session("SSRPM").GetUserDataOfCurrentUser(IProfile)
```

2.1.59. SSRPM.IsCurrentUserEnrolled

Purpose: Return 1 if the user is enrolled, otherwise it will return 0.

Description:
Call this method to determine if the current user is enrolled. Returns -47 if the user is not initialized.

Syntax: IsCurrentUserEnrolled*()*

Parameters: None.

Return value: 0, 1 or -47

Example (ASP code):

```
If(SSRPM.IsCurrentUserEnrolled()) Then
   response.write("User is enrolled!")
End If
```


2.1.60. SSRPM.SendAACMessageToCurrentUser

Purpose: Send an advanced authentication message to for the user under whose credentials this call is made.

Description:
Send an advanced authentication message to a user. The SendAACMessageToCurrentUser interface method allows you to send a message to the user under whose credentials the call is made. Returns a PIN code GUID for validation messages. The validation message contain a PIN code that is used to validate e-mail addresses or mobile phone numbers of users that are not yet enrolled.

Syntax: *SendAACMessageToCurrentUser([in] String ToAddress, [in] Number DeliveryMethod, [in] Number MessageActionType, [in] Number LanguageId, [out] Object PinCodeGuid)*

Parameters:

ToAddress:	The destination to which to send the message.
DeliveryMethod:	The method by which the message will be sent. Select 1 for SMS and 2 for email.
MessageActionType:	The type of message to send. 1: Send (PIN code) 2: Test 3: Resend (PIN code) 4. Validation (PIN code) 5. Validation resend (PIN code)
LanguageID:	The preferred language of the message. If the language is not available English will be used. The value for English is 9
PinCodeGuid	A GUID that is associated with the PIN code, applies mostly to the validation send and resend messages.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
RetVal = Session("SSRPM").SendAACMessageToCurrentUser("J.Smith@company.com", 2, 1, 9, PinCodeGuid)


If (RetVal <> 0) Then
   response.write("Failed to send message.")
End If
```

2.1.61. SSRPM.EnrollCurrentUser

Purpose: Enroll a user into SSRPM.

Description:
The GetUserDataOfCurrentUser interface method (see: *SSRPM.GetUserDataOfCurrentUser*) must be called first to identify a user which must enroll into SSRPM. The EnrollCurrentUser interface method enrolls the user under whose credentials this call is made into SSRPM .

Syntax: *EnrollCurrentUser([In] StringArray QuestionList, [In] StringArray AnswerList, [In] String EmailAddress, [In] String MobilePhoneNumber)*

Parameters:

QuestionList:	An array of user questions.
AnswerList:	An array of user answers for all specified user questions.
EmailAddress:	The email address of the user for advanced authentication
MobilePhoneNumber:	The mobile phone number of the user for advanced authentication

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim QuestionArray(5),AnswerArray(5)

'Fill arrays with specified questions and answers from user

RetVal = SSRPM.EnrollCurrentUser(QuestionArray, AnswerArray, EmailAddress,
MobilePhoneNumber)
```

2.1.62. SSRPM.GetWebTokenByID

Purpose: Get the GUID of the web token. This call is used to validate that the website is actually the SSPRM web interface.

Description:
Call this method to get the web token GUID which should be included in the HTML of the web interface.

Syntax: GetWebTokenByID*([In] String TokenID, [Out] Object TokenGuid)*

Parameters:

TokenID:	The ID of the token.
TokenGuid:	The token as a GUID.

Return value: 0

Example (ASP code):

```
SSRPM.GetWebTokenByID("10", TokenGuid)
```

2.1.63. SSRPM.SendAACMessageToEx

Purpose: Send an advanced authentication message to a user.

Description:
Send an advanced authentication message to a user. The SendAACMessageTo interface method allows you to send a message to a user of which the email address or phone number is not known within SSRPM. This can be used to send a test message during the enrollment of a new user. Returns a PIN code GUID for validation messages. The validation messages contain a PIN code that is used to validate e-mail addresses or mobile phone numbers of users that are not yet enrolled.

Syntax: *SendAACMessageToEx([in] String Domain, [in] String Account, [in] String Password, [in] String ToAddress, [in] Number DeliveryMethod, [in] Number MessageActionType, [in] Number LanguageId, [out] Object PinCodeGuid)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
Password:	The password of the account.
ToAddress:	The destination to which to send the message.
DeliveryMethod:	The method by which the message will be sent. Select 1 for SMS and 2 for email.
MessageActionType:	The type of message to send. 1: Send (PIN code) 2: Test 3: Resend (PIN code) 4. Validation (PIN code) 5. Validation resend (PIN code)
LanguageID:	The preferred language of the message. If the language is not available English will be used. The value for English is 9
PinCodeGuid	A GUID that is associated with the PIN code, applies mostly to the validation send and resend messages.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
RetVal = Session("SSRPM").SendAACMessageToEx("MyDomain", "j.smith", "Password123",
"J.Smith@company.com", 2, 1, 9, PinCodeGuid)


If (RetVal <> 0) Then
   response.write("Failed to send message.")
End If
```

2.1.64. SSRPM.IsValidationPINCodeAccepted

Purpose: Verify if a PIN code sent by SSRPM to validate the e-mail address or mobile phone number during enrollment is valid.

Description:
Call this method to verify if a PIN code sent with SSRPM.SendAACMessage is valid.

Syntax: *IsValidationPincodeAccepted([In] String Pincode, [In] DeliveryMethod)*

Parameters:

Pincode:	The pincode provided by the user
DeliveryMethod:	The method by which the message was sent. Select 1 for SMS and 2 for email.
PincodeGuid:	The GUID of the validation PIN code.

Return value: 0 if the PIN code is valid, non-zero if it is not.

Example (ASP code):

```
If(SSRPM.IsValidationPinCodeAccpted("1234", 1, "3bdfe7df-b33f-4070-a435-
c90aa37d7d2b") = 0)

   response.write("You have entered the correct PIN code!")
```

2.1.65. SSRPM.GetResetProfileOptionsEx

Purpose: Get the enabled SSRPM Profile reset options.

Description:
The GetResetData interface method (see: *SSRPM.GetResetData* on page 7) must be called first to identify the user of which the applicable SSRPM Profile reset options must be retrieved.

The GetResetProfileOptionsEx interface method returns the mask number of all enabled SSRPM Profile reset options. This number identifies which settings are enabled within the current SSRPM Profile. Please refer to 'SSRPMProfile.GetOptions for a complete list of available objects.
This call should be used instead of GetResetProfileOptions because the underlying variable contain the profile options was changed to support more mask values.

> Note: The mask number can be used within a logical And-expression to check if a certain SSRPM Profile option has been enabled. This is shown in the example (ASP Code) below, which checks if the 'Show incorrect answer' option has been enabled.

Syntax: *GetResetProfileOptionsEx()*

Parameters: None.

Return value: The mask number of all enabled SSRPM Profile reset options.

Example (ASP code):

```
ResetProfileOptions = SSRPM.GetResetProfileOptionsEx()

If(ResetProfileOptions And 32) Then
    response.write("Incorrect answers may be shown.")
Else
    response.write("Incorrect answers may not be shown.")
End if
```

2.1.66. SSRPM.UnEnrollCurrentUser

Purpose: Unenroll a user from SSRPM.

Description:
The GetUserDataOfCurrentUser interface method (see: *SSRPM.GetUserDataOfCurrentUser*) must be called first to identify a user which must unenrolled from SSRPM. The Unenroll interface method unenrolls a user from SSRPM.

Syntax: UnEnrollCurrentUser*()*

Parameters: None.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
'unenroll selected user
RetVal = SSRPM.UnEnrollCurrentUser()
```


2.1.67. SSRPM.SetSourceApplication

Purpose: Configures the COM object to act like the specified source application for behavioral purposes (e.g. SMS authentication for specific scenario's).

Description:
Call this method to set the behavior of the COM object to the specified source application.

Syntax: SetSourceApplication*([In] Number SourceApp)*

Parameters:

SourceApp:	A number representing the application: 1: Enrollment Wizard 2: Reset Wizard 4: COM object (Web interface)

Return value: 0

Example (ASP code):

```
SSRPM.SetSourceApplication(2)
```

2.1.68. SSRPM.GetAdSelfServiceData

Purpose: Get Active Directory self service data for a user.

Description:
Get Active Directory self service data for a user. This function returns an array of string arrays that can be used to build the AD Self Service functionality.

Syntax: *GetAdSelfServiceData([In] String Domain, [In] String Account, [In] String Password, [In] Array of StringArrays ADSelfServiceData)*

Parameters:

Domain: The name of the domain of which the user is a member of.
Account: The account name of the user.
Password: The password of the account.
ADSelfServiceData: An array of string arrays where each string array contains 5 entries.
1: The name of the attribute
2: The options of that attribute. This mask value can contain the following flags:
    0: None
    1: Allow write
    2: Allow empty
    4: Validate
3: The validation regular expression.
4: The value of the attribute.
5: The data type of the attribute
    0: Unkown
    1: String
    2: Multi-line string
    3: Binary data
    4: String array

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise .

Example (ASP code):

```
Dim ADSelfServiceData

RetVal = Session("SSRPM").GetAdSelfServiceData("CATS", "J.Smith", "Password123", ADSelfServiceData)
```

2.1.69. SSRPM.SetAdSelfServiceData

Purpose: Update the Active Directory information of a user.

Description:
Updates the Active Directory for a user with the specified data. This function expects an array of string arrays where each element has two values, the name of the attribute and the specified value. The service will perform additional validation before propagating the changes to Active Directory.

Syntax: *SetAdSelfServiceData([In] String Domain, [In] String Account, [In] String Password, [In] Array of StringArrays ADSelfServiceData)*

Parameters:

Domain:	The name of the domain of which the user is a member of.
Account:	The account name of the user.
Password:	The password of the account.
ADSelfServiceData:	An array of string arrays where each string array contains 2 entries. 1: The name of the attribute 2: The value of the attribute.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise .

Example (ASP code):

```
Dim ADSelfServiceData

RetVal = Session("SSRPM").SetAdSelfServiceData("CATS", "J.Smith", "Password123", ADSelfServiceData)
```

2.1.70. SSRPM.GetAdSelfServiceDataOfCurrentUser

Purpose: Get Active Directory self service data for the user under whose credentials this call was made.

Description:
Get Active Directory self service data for the user under whose credentials this call was made. This function returns an array of string arrays that can be used to build the AD Self Service functionality.

Syntax: *GetAdSelfServiceData*OfCurrentUser*([In] Array of StringArrays ADSelfServiceData)*

Parameters:
ADSelfServiceData: An array of string arrays where each string array contains 5 entries.
1: The name of the attribute
2: The options of that attribute. This mask value can contain the following flags:
0: None
1: Allow write
2: Allow empty
4: Validate
3: The validation regular expression.
4: The value of the attribute.
5: The data type of the attribute
0: Unkown
1: String
2: Multi-line string
3: Binary data
4: String array

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise .

Example (ASP code):

```
Dim ADSelfServiceData

RetVal = Session("SSRPM").GetAdSelfServiceDataOfCurrentUser(ADSelfServiceData)
```

2.1.71. SSRPM.SetAdSelfServiceDataOfCurrentUser

Purpose: Set Active Directory self service data for the user under whose credentials this call was made.

Description:
Updates the Active Directory for the user under whose credentials this call was made with the specified data. This function expects an array of string arrays where each element has two values, the name of the attribute and the specified value. The service will perform additional validation before propagating the changes to Active Directory.

Syntax: *SetAdSelfServiceDataOfCurrentUser([In] Array of StringArrays ADSelfServiceData)*

Parameters:
ADSelfServiceData: An array of string arrays where each string array contains 2 entries.
1: The name of the attribute
2: The value of the attribute.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise .

Example (ASP code):

```
Dim ADSelfServiceData

RetVal = Session("SSRPM").SetAdSelfServiceDataOfCurrentUser(ADSelfServiceData)
```

2.1.72. SSRPM.GetHelpdeskGetUserCollectionEx

Purpose: Get users for helpdesk Caller identification that match the criteria (domain and name).

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetHelpdeskGetUserCollectionEx interface method retrieves the list of account names of users that match the domain and resemble the name.

Syntax: *GetHelpdeskGetUserCollectionEx([In] StringArray Columns, [In] StringArray Needles, [Out] StringArray UserList)*

Parameters:

Columns:	The name of the column on which to filter the users.
Needles:	The filter terms to use for the specified columns.
UserList:	An array of users where each entry is an array of strings containing the name of the attribute and its value.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim Columns
Dim Needles
Dim UserList
RetVal = SSRPM.GetHelpdeskGetUserCollectionEx(Columns,Needles",UserList)
```


2.1.73. SSRPM.GetHelpdeskAuthenticationDataByCanonicalNameEx

Purpose: Get the helpdesk Caller identification data of the specified user by canonical name and loads the related profile in to the COM object.

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetHelpdeskAuthenticationDataByCanonicalName interface method retrieves the list of questions applicable to the specified user.

Syntax: *GetHelpdeskAuthenticationDataByCanonicalNameEx([In] String CanonicalName, [Out] SSRPMProfile Profile, [Out] StringArray QuestionList)*

Parameters:

CanonicalName:	The canonical name of the user.
Profile:	A newly created SSRPM Profile object. This object will contain the applicable SSRPM Profile of the current user.
QuestionList:	An array of user questions.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim QuestionList
Set Profile = Server.CreateObject("SSRPMCOM.SSRPMProfile")

RetVal = SSRPM.GetHelpdeskAuthenticationDataByCanonicalName("cats.animals/Users/John Smith", Profile, QuestionList)
```

2.1.74. SSRPM.SendEventNotification

Purpose: Send notification for specific events.

Description:
Call this method to send a notification message for a specific event.

Syntax: *SendEventNotification([In] Number EventType, [In] Array of stringarrays KeyWords)*

Parameters:

EventType:	The ID of the event. Supported events: 8192: IP address has been blocked
KeyWords:	An array of string arrays containing keywords that can be used in the message template. Each string array should contain two string, where the first value is the name of the keywords and the second the value.

Return value: 0

Example (ASP code):

```
SSRPM.SendEventNotification.(8192, KeyWords)
```

2.1.75. SSRPM.GetPasswordComplexityNameSimilarityTokens

Purpose: Get the tokens of the user using SSRPM that are required to check the password similarity complexity rule configured in the Active Directory .

Description:
Get the tokens that are required to check the password similarity complexity rule configured in the Active Directory. This function returns a string array with the tokens that can't be part of the password.

Syntax: GetPasswordComplexityNameSimilarityTokens*([In] StringArray Tokens)*

Parameters:
Tokens: An array of strings that are not allowed in the password

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise .

Example (ASP code):

```
Dim Tokens
RetVal = Session("SSRPM").GetPasswordComplexityNameSimilarityTokens(Tokens)
```

2.1.76. SSRPM.GetOnboardingData

Purpose: Identify a user and retrieve all the onboarding data of that user.

Description:
A connection with the SSRPM Service must be made first with the Connect interface method (see: *SSRPM.Connect* on page 2). The GetOnboardingData interface method retrieves all questions that a user must answer to onboard.

Syntax: *GetOnboardingData([In] String ID,[In] String Domain, [In] String Account, [Out] StringArray OnboardingDataList, [In] SSRPMProfile Profile)*

Parameters:

ID:	The user idenitifer, i.e. the user attribute with options value '1'.
Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
OmboardingData	An array of onboarding user attributes. Each attribute has the following attributes: ID - Number Name - The name of the attribute Value - The value of the attribute Options - A mask value representing the type of attribute. Possible mask values: 1: User identifier 2: Validation attribute 4: Mobile phone number 8: E-mail address 16: Case sensitive comparison
Profile:	The SSRPM profile of the user

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim OnboardingData

Set Profile  = Server.CreateObject("SSRPMCOM.SSRPM")

RetVal = SSRPM.GetOnboardingData("J.smith", "CATS","John
Smith",OnboardingData, Profile)
```

2.1.77. SSRPM.ValidateOnboardingAttributes

Purpose: Validate the specified values of the attributes of a user match the stored values in the database.

Description:
The GetOnboardingData interface method (see: *SSRPM.GetOnboardingData*) must be called first to identify the user of which the attributes must be validated. The ValidateOnboarding attributes interface method checks if all values of the attributes of a user are entered correctly.

Syntax: *ValidateOnboardingAttributes([In] String ID, [In] Array of StringArray AttributeValuesList, [Out] Array of StringArray InCorrectAttributesList)*

Parameters:

ID:	The user idenitifer, i.e. the user attribute with options value '1'.
AttributeList:	An array of attributes (see *SSRPM.GetOnboardingData* more information). Each string array should contain: 1) ID - Database ID of the attribute 2) Name - The name of the attribute 3) Vaue - The specified value of the attribute
IncorrectAttributesList:	An array which (optionally) will contain an all attributes which are answered incorrectly.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise (-11 when the questions are answered incorrectly).

Example (ASP code):

```
Dim AttributesArray(5),InCorrectAttributesArray

'Fill arrays with attributesArray with the specified answers from user

RetVal = SSRPM.CheckAnswers(1,AttributesArray,InCorrectAttributesArray)

If (RetVal <> 0) Then
    InCorrectAttributeNr = UBound(InCorrectAttributesArray)

    If(InCorrectAttributeNr <> 0) Then
         response.write(➔
           "You've answered " &InCorrectAttributeNr& " attributes invalid.")
    End if
End If
```

2.1.78. SSRPM.SendOnboardingValidationMessageTo

Purpose: Send an onboarding message to a user.

Description:
Send an onboarding authentication message to a user. The SendOnboardingValidationMessageTo interface method allows you to send a message to a user of which the email address or phone number is stored in the onboarding attributes within SSRPM. Returns a PIN code GUID of the validation messages. The validation messages contain a PIN code that is used to authenticate the user.

Syntax: *SendOnboardingValidationMessageTo([in] String ID, [in] Number DeliveryMethod, [in] Number MessageActionType, [in] Number LanguageId, [out] Object PinCodeGuid)*

Parameters:

Domain:	The name of the domain of which the current user is a member of.
Account:	The account name of the current user.
Password:	The password of the account.
ToAddress:	The destination to which to send the message.
DeliveryMethod:	The method by which the message will be sent. Select 1 for SMS and 2 for email.
MessageActionType:	The type of message to send. 1: Send (PIN code) 2: Test 3: Resend (PIN code) 4. Validation (PIN code) 5. Validation resend (PIN code) 6. Onboarding (PIN code)
LanguageID:	The preferred language of the message. If the language is not available English will be used. The value for English is 9
PinCodeGuid	A GUID that is associated with the PIN code, applies mostly to the validation send and resend messages.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
RetVal = Session("SSRPM").SendOnboardingValidationMessageTo("12345", 6, 1, 9, PinCodeGuid)


If (RetVal <> 0) Then
   response.write("Failed to send message.")
End If
```

2.1.79. SSRPM.IsOnboardingPINCodeAccepted

Purpose: Verify if a PIN code sent by SSRPM to authenticate a user using the e-mail address or mobile phone number during onboarding is valid.

Description:
Call this method to verify if a PIN code sent with SSRPM.SendOnboardingValidationMessageTo is valid.

Syntax: *IsOnboardingPINCodeAccepted I([In] String Pincode, [In] DeliveryMethod)*

Parameters:

Pincode:	The pincode provided by the user
DeliveryMethod:	The method by which the message was sent. Select 1 for SMS and 2 for email.
PincodeGuid:	The GUID of the validation PIN code.

Return value: 0 if the PIN code is valid, non-zero if it is not.

Example (ASP code):

```
If(SSRPM.SendOnboardingValidationMessageTo("1234", 1, "3bdfe7df-b33f-4070-
a435-c90aa37d7d2b") = 0)
   response.write("You have entered the correct PIN code!")
```


2.1.80. SSRPM.OnboardingSetPassword

Purpose: Set the user password to a random password and enables the account, if disabled.

Description:
The GetOnboardingData interface method (see: *SSRPM.GetOnboardingData*) must be called first to identify the user of which the password must be set.The OnboardingSetPassword interface method sets a random password for that of the user.

Syntax: *OnboardingSetPassword([In] String ID, [In] Number SelectedValidationMethod, [In] Array of StringArrays Attributes, [In] String SmsPinCode, [In] String SmsPinCodeGuid, [In] String EmailPinCode, [In] String EmailPinCodeGuid, [Out] String NewPassword)*

Parameters:

ID:	The user idenitifer, i.e. the user attribute with options value '1'.
SelectedValidationMethod:	The validation method that the user is using. Possible values: 2: Validation attributes 4: Mobile phone number 8: E-mail address
Attributes:	An array of onboarding user attributes. See *SSRPM.GetOnboardingData* for more information
SmsPinCode:	The PIN code entered by the user which was sent by SMS.
SmsPinCodeGuid:	The GUID associated with PIN code entered by the user which was sent by SMS.
EmailPinCode:	The PIN code entered by the user which was sent by e-mail.
EmailPinCodeGuid:	The GUID associated with PIN code entered by the user which was sent by e-mail.
NewPassword:	The new randomly generated password.

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise (-11 when the questions are answered incorrectly).

Example (ASP code):

```
Dim AttributeArray


'Fill arrays with questions and specified answers from user


RetVal = SSRPM.OnboardingSetPassword("1234", 2, AttributeArray, "", "", "", "", NewPassword)


If (RetVal = 0) Then
    response.write("The password has been set successfully!")
ElseIf (RetVal = -11) Then
    response.write("The values of the attributes are incorrect.")
End If
```

2.1.81. SSRPM.OnboardingChangePassword

Purpose: Change the password of the user who is onboarding.

Description:
Call this method to change the password of a specified account. This call will also mark the onboarding of the user as succesful and will remove the associated onboarding data of the user.

Syntax: OnboardingChangePassword*([In] String ID, [In] String OldPassword, [In] String NewPassword)*

Parameters:

ID:	The user idenitifer, i.e. the user attribute with options value '1'.
OldPassword:	The current password
NewPassword:	The new password

Return value: 0

Example (ASP code):

```
SSRPM.OnboardingChangePassword("1234", "MyOldPassword", "MyNewPassword")
```

2.1.82. SSRPM.OnboardingImport

Purpose: Add, update or remove onboarding data in the database.

Description:
The OnboardingImport interface method is used to import the users onboarding data. The users array and the attributes array should have the same length and order.

Syntax: *OnboardingSetPassword([In] String Action, [In] Array of String Arrays Users, [In] Array of StringArrays Attributes)*

Parameters:

Action:	The action that should be performed. Possible options: "new"or "delete". When using the "new" action and a user already exists, all the data of that user will be overwritten. The "delete" action only requires the that the user identifier attribute is present.
Users	An array containing user information: 1) Account SID (can be empty) 2) Domain Name (required) 3) SAMAccountName (required) 4) OnboardingDate - The date after which a user is allowed to onboard (can be empty).
Attributes:	An array containing user's attribute information: 1) Name 2) Value 3) Options

Return value: Number. 0 if the interface method executed successfully; non-zero otherwise.

Example (ASP code):

```
Dim UsersArray, AttributesArray

'Fill arrays user data

RetVal = SSRPM.OnboardingImport("new", UsersArray, AttributesArray)

If (RetVal = 0) Then
    response.write("The password has been set successfully!")
ElseIf (RetVal = -11) Then
    response.write("The values of the attributes are incorrect.")
End If
```

2.2. SSRPM COM object: SSRPMProfile

This object contains the SSRPM Profile which is applicable for an enrolled user. An SSRPM Profile contains settings like for instance the number of questions a user needs to answer, or the minimum length an answer must be and is applicable for one or more organizational units (OU's) or a whole domain.

See the "Administrator's Guide" for more information about SSRPM Profiles, of which the latest version is available on the *Tools4ever website* http://www.tools4ever.com.

2.2.1. SSRPMProfile.GetOptions

Purpose: Get the enabled SSRPM Profile options.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4).

The interface method returns the mask number of all enabled SSRPM Profile options of the SSRPM Profile object. This number identifies which settings are enabled within the current SSRPM Profile. See below for a table with all SSRPM Profile options with their corresponding mask number.

Note: The mask number can be used within a logical And-expression to check if a certain SSRPM Profile option has been enabled. This is shown in the example (ASP Code) below, which checks if Account Blocking has been enabled.

Mask table:

SSRPM Profile Option	Mask Number
A single user cannot have the same answer for multiple questions	1
An answer cannot be a word which is in the corresponding question	2
Case-sensitive comparison of answers	4
Store answers with irreversible encryption	8
Check answers immediately	16
Show incorrect answer	32
Show password complexity rules	64
Hide answers in the SSRPM Enrollment Wizard and SSRPM Reset Wizard	128
Enable Account Blocking	256
Enable E-mail Notification	512
Add an Answer confirmation box to the SSRPM Enrollment Wizard to prevent typo's	1024
Warn users on that they need to re-enroll after a certain period.	2048
Force user enrollment and re-enrollment	4096
Enable minimum answer length	8192
Don't unlock password during reset	16384
Disable password reset	32768
Allow unlock account	65536
Use minimum question length for user defined questions	131072
Use random questions	262144
Don't enforce password history	524288
Show the user new questions upon error (in combination with random questions)	1048576
Enforce the maximum number of resets	2097152
Use 256 hashing for the answers	4194304
Allow offline logon	8388608
Use reversible encryption for the answers	16777216
Show the password complexity rules from PCM	33554432
Enable helpdesk Caller Identification	67108864
Notify end-users of certain events	134217728
Allow end-users to select advanced authentication method	268435456
Enabled/Disable change password functionality	536870912
Block account after a number of incorrect PIN code attempts	1073741824

Note: See the "Administrator's Guide" for more information about the SSRPM Profiles options, of which the latest version is available on the *Tools4ever website* http://www.tools4ever.com.

Syntax: *GetOptions()*

Parameters: None.

Return value: The mask number of all enabled SSRPM Profile options.

Example (ASP code):

```
ProfileOptions = Profile.GetOptions()


If(ProfileOptions And 256) Then
    response.write("Account Blocking has been enabled.")
Else
    response.write("Account Blocking has not been enabled.")
End if
```

2.2.2. SSRPMProfile.GetNrOfMandatoryQuestionsNeededToEnroll

Purpose: Get the amount of mandatory administrator defined questions.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the number of mandatory administrator defined questions of the SSRPM Profile object.

Mandatory Administrator defined questions are questions which are defined by the administrator *and must be answered by the user.*

Syntax: *GetNrOfMandatoryQuestionsNeededToEnroll()*

Parameters: None.

Return value: The number of mandatory administrator defined questions.

Example (ASP code):

```
AdminQuestions = Profile.GetNrOfMandatoryDefinedQuestionsNeededToEnroll()
```

2.2.3. SSRPMProfile.GetNrOfAdminDefinedQuestionsNeededToEnroll

Purpose: Get the amount of administrator defined questions.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the number of administrator defined questions of the SSRPM Profile object.

Administrator defined questions are questions which are defined by the administrator from which the user must choose when enrolling.

Syntax: *GetNrOfAdminDefinedQuestionsNeededToEnroll()*

Parameters: None.

Return value: The number of administrator defined questions.

Example (ASP code):

```
AdminQuestions = Profile.GetNrOfAdminDefinedQuestionsNeededToEnroll()
```

2.2.4. SSRPMProfile.GetNrOfUserDefinedQuestionsNeededToEnroll

Purpose: Get the amount of user defined questions.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the number of user defined questions of the SSRPM Profile object.

User defined questions are questions which are created by a user itself in which case the user must make up its own questions when enrolling.

Syntax: *GetNrOfUserDefinedQuestionsNeededToEnroll()*

Parameters: None.

Return value: The number of user defined questions.

Example (ASP code):

```
UserQuestions = Profile.GetNrOfUserDefinedQuestionsNeededToEnroll()
```

2.2.5. SSRPMProfile.GetMinimumQuestionLength

Purpose: Get the minimum question length.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the minimum number of characters that a user defined question must contain of the SSRPM Profile object.

Syntax: *GetMinimumQuestionLength()*

Parameters: None.

Return value: The minimum number of characters a user defined question must contain.

Example (ASP code):

```
MinimumQuestionLength = Profile.GetMinimumQuestionLength()
```

2.2.6. SSRPMProfile.GetMinimumAnswerLength

Purpose: Get the minimum answer length.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the minimum number of characters an answer must contain of the SSRPM Profile object.

Syntax: *GetMinimumAnswerLength()*

Parameters: None.

Return value: The minimum number of characters an answer must contain.

Example (ASP code):

```
MinimumAnswerLength = Profile.GetMinimumAnswerLength()
```


2.2.7. SSRPMProfile.GetMandatoryQuestionsList

Purpose: Get a list of all mandatory admin defined questions.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). This interface method retrieves a list of all mandatory administrator defined questions of the SSRPM Profile object.

Mandatory Administrator defined questions are questions which are defined by the administrator and must be answered during enrollment.

Syntax: *GetMandatoryQuestionsList([In] String Language, [Out] StringArray QuestionList)*

Parameters:

Language:	The language name of the questions. For instance: "English", "French" or "German". If the specified language is not available or empty the default language will be used (English).
QuestionList:	An array, which will contain all mandatory administrator defined questions of the SSRPM Profile.

Return value: None.

Example (ASP code):

```
Dim MandatoryQuestionArray

Profile.GetMandatoryQuestionsList "English",MandatoryQuestionArray
```

2.2.8. SSRPMProfile.GetAdminDefinedQuestionsList

Purpose: Get a list of all administrator defined questions.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). This interface method retrieves a list of all default administrator defined questions of the SSRPM Profile object.

Administrator defined questions are questions which are defined by the administrator from which the user must choose when enrolling.

Syntax: *GetAdminDefinedQuestionsList([In] String Language, [Out] StringArray QuestionList)*

Parameters:

Language:	The language name of the questions. For instance: "English", "French" or "German". If the specified language is not available or empty the default language will be used (English).
QuestionList:	An array, which will contain all administrator defined questions of the SSRPM Profile.

Return value: None.

Example (ASP code):

```
Dim AdminQuestionArray

Profile.GetAdminDefinedQuestionsList "English",AdminQuestionArray
```

2.2.9. SSRPMProfile.IsEmailAuthenticationEnabled

Purpose: Checks if email authentication is enabled.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). This interface method checks if email authentication is enabled.
None

Syntax: *IsEmailAuthenticationEnabled()*

Parameters: None

Return value: 1 (yes) or 0 (no).

Example (ASP code):

```
If(Profile.IsEmailAuthenticationEnabled() = 1 Then
      Response.write("Email authentication is enabled.")
End If
```

2.2.10. SSRPMProfile.IsSmsAuthenticationEnabled

Purpose: Checks if SMS authentication is enabled.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). This interface method checks if SMS authentication is enabled.

Syntax: *IsSmsAuthenticationEnabled()*

Parameters: None

Return value: 1 (yes) or 0 (no).

Example (ASP code):

```
If(Profile.IsSmsAuthenticationEnabled() = 1 Then
      Response.write("SMS authentication is enabled.")
End If
```


2.2.11. SSRPMProfile.IsEmailAddressOptionalInput

Purpose: Checks if providing an email address during enrollment is optional.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). This interface method checks if providing an email address during enrollment is optional

Syntax: *IsEmailAuthenticationEnabled()*

Parameters: None

Return value: 1 (yes) or 0 (no).

Example (ASP code):

```
If(Profile.IsEmailAddressOptionalInput() = 1 Then
      Response.write("Email address is optional.")
End If
```

2.2.12. SSRPMProfile.IsPhoneNumberAuthenticationEnabled

Purpose: Checks if providing an mobile phone number during enrollment is optional.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). This interface method checks if providing an mobile phone number during enrollment is optional

Syntax: *IsPhoneNumberAuthenticationEnabled()*

Parameters: None

Return value: 1 (yes) or 0 (no).

Example (ASP code):

```
If(Profile.IsPhoneNumberAuthenticationEnabled() = 1 Then
      Response.write("Mobile phone number is optional.")
End If
```

2.2.13. SSRPMProfile.GetEmailAuthenticationEnrollmentOptions

Purpose: Get the enrollment options for email authentication of the current profile.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the minimum number of characters that a user defined question must contain of the SSRPM Profile object.

The GetEmailAuthenticationEnrollmentOptions interface method returns the mask number of all enabled SSRPM email authentication options for the enrollment process. This number identifies which settings are enabled within the current SSRPM Profile.

> Note: The mask number can be used within a logical And-expression to check if a certain SSRPM Profile option has been enabled.

Mask values:

Show data:	*1*	Show the enrollment data for email authentication (i.e. email address)
Allow edit:	*2*	Allow user to change the provided enrollment data.
Allow test message:	*4*	Allow user to send a test message
Limit test messages:	8	There is a maximum number of test message the user can send.
Input optional:	16	Providing the information for authentication is optional.

Syntax: *GetEmailAuthenticationEnrollmentOptions()*

Parameters: None.

Return value: The Mask number of all enabled enrollment options of the current profile relating to email authentication.

Example (ASP code):

```
EnrollmentOptions = Profile.GetEmailAuthenticationEnrollmentOptions()


If(EnrollmentOptions And 4) Then
    response.write("The user is allowed to send test messages.")
Else
    response.write("The user is NOT allowed to send test messages.")
End if
```

2.2.14. SSRPMProfile.GetSmsAuthenticationEnrollmentOptions

Purpose: Get the enrollment options for email authentication of the current profile.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the minimum number of characters that a user defined question must contain of the SSRPM Profile object.

The GetSmsAuthenticationEnrollmentOptions interface method returns the mask number of all enabled SSRPM SMS authentication options for the enrollment process. This number identifies which settings are enabled within the current SSRPM Profile.

> Note: The mask number can be used within a logical And-expression to check if a certain SSRPM Profile option has been enabled.

Mask values:

Show data:	*1*	Show the enrollment data for email authentication (i.e. phone number)
Allow edit:	*2*	Allow user to change the provided enrollment data.
Allow test message:	*4*	Allow user to send a test message
Limit test messages:	8	There is a maximum number of test message the user can send.
Input optional:	16	Providing the information for authentication is optional.

Syntax: *GetSmsAuthenticationEnrollmentOptions()*

Parameters: None.

Return value: The Mask number of all enabled enrollment options of the current profile relating to SMS authentication.

Example (ASP code):

```
EnrollmentOptions = Profile.GetSmsAuthenticationEnrollmentOptions()


If(EnrollmentOptions And 4) Then
    response.write("The user is allowed to send test messages.")
Else
    response.write("The user is NOT allowed to send test messages.")
End if
```

2.2.15. SSRPMProfile.GetEmailMaxAllowedResends

Purpose: Get the maximum number of resends a user is allowed to trigger with the current profile relating to email authentication.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the maximum number of resends a user can trigger .

Syntax: *GetEmailMaxAllowedResends()*

Parameters: None.

Return value: Number.

Example (ASP code):

```
MaxResends = Profile.GetEmailMaxAllowedResends()
```

2.2.16. SSRPMProfile.GetSmsMaxAllowedResends

Purpose: Get the maximum number of resends a user is allowed to trigger with the current profile relating to SMS authentication.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the maximum number of resends a user can trigger .

Syntax: *GetSmsMaxAllowedResends()*

Parameters: None.

Return value: Number.

Example (ASP code):

```
MaxResends = Profile.GetSmsMaxAllowedResends()
```


2.2.17. SSRPMProfile.GetEmailResetResendDelay

Purpose: Get the delay between resends triggered by a user with the current profile relating to email authentication.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the delay interval (in seconds) which needs to pass before the user is allowed to trigger another resend.

Syntax: *GetEmailResetResendDelay()*

Parameters: None.

Return value: Number.

Example (ASP code):

```
Delay = Profile.GetEmailResetResendDelay()
```

2.2.18. SSRPMProfile.GetSmsResetResendDelay

Purpose: Get the delay between resends triggered by a user with the current profile relating to SMS authentication.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the delay interval (in seconds) which needs to pass before the user is allowed to trigger another resend.

Syntax: *GetSmsResetResendDelay()*

Parameters: None.

Return value: Number.

Example (ASP code):

```
Delay = Profile.GetSmsResetResendDelay()
```


2.2.19. SSRPMProfile.GetEmailMaxAllowedTests

Purpose: Get the maximum number of test messages a user is allowed to trigger with the current profile relating to email authentication.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the maximum number of tests a user can trigger .

Syntax: *GetEmailMaxAllowedTests()*

Parameters: None.

Return value: Number.

Example (ASP code):

```
MaxResends = Profile.GetEmailMaxAllowedTests()
```

2.2.20. SSRPMProfile.GetSmsMaxAllowedTests

Purpose: Get the maximum number of test messages a user is allowed to trigger with the current profile relating to SMS authentication.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the maximum number of tests a user can trigger .

Syntax: *GetSmsMaxAllowedTests()*

Parameters: None.

Return value: Number.

Example (ASP code):

```
MaxResends = Profile.GetSmsMaxAllowedTests()
```


2.2.21. SSRPMProfile.GetEmailEnrollTestDelay

Purpose: Get the delay between test messages triggered by a user with the current profile relating to email authentication.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the delay interval (in seconds) which needs to pass before the user is allowed to trigger another test.

Syntax: *GetEmailEnrollTestDelay()*

Parameters: None.

Return value: Number.

Example (ASP code):

```
Delay = Profile.GetEmailEnrollTestDelay()
```

2.2.22. SSRPMProfile.GetSmsEnrollTestDelay

Purpose: Get the delay between test messages triggered by a user with the current profile relating to SMS authentication.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the delay interval (in seconds) which needs to pass before the user is allowed to trigger another test.

Syntax: *GetSmsEnrollTestDelay()*

Parameters: None.

Return value: Number.

Example (ASP code):

```
Delay = Profile.GetSmsEnrollTestDelay()
```


2.2.23. SSRPMProfile.GetSmsPrefix

Purpose: Get the configured phone number prefix for SMS authentication of this profile.

Description:
The GetSmsPrefix interface method (see: *SSRPM.GetResetData* on page 7) must be called first to get the appropriate profile.

Syntax: *GetSmsPrefix([Out] String Prefix)*

Parameters:
Prefix: The prefix used for SMS authentication.

Return value: 0 is successful, non-zero otherwise.

Example (ASP code):

```
RetVal = SSRPM.GetSmsPrefix(Prefix)


If(RetVal = 0) Then
    response.write("Your prefix:" & Prefix & "." )
Else
    response.write("An error occurred when getting prefix.")
End if
```

2.2.24. SSRPMProfile.GetMaximumQuestionsPerPage

Purpose: Get the maximum number of questions per page.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the minimum number of characters that a user defined question must contain of the SSRPM Profile object.

Syntax: *GetMaximumQuestionsPerPage()*

Parameters: None.

Return value: The maximum number of questions per page.

Example (ASP code):

```
MaximumQuestionsPerPage = Profile.GetMaximumQuestionsPerPage()
```

2.2.25. SSRPMProfile.GetEmailAuthenticationResetOptions

Purpose: Get the enabled options for the reset action of the e-mail authentication.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4).

The interface method returns the mask number of all enabled options of the reset action of e-mail authentication. See below for a table with all options with their corresponding mask number.

> Note: The mask number can be used within a logical And-expression to check if a certain option has been enabled. This is shown in the example (ASP Code) below, which checks if allow resend option has been enabled.

Mask table:

SSRPM Profile Option	Mask Number
Allow resend of PIN code	1
Limit number of resends	2
Allow reset without PIN code (if e-mail address is unknown)	4
E-mail authentication disabled when using COM object	8
E-mail authentication disabled when using Reset Wizard	16
E-mail authentication disabled when using Offline Logon	32
Show partial e-mail address of recipient	64

Syntax: *GetEmailAuthenticationResetOptions()*

Parameters: None.

Return value: The mask number of all enabled options.

Example (ASP code):

```
ProfileOptions = Profile.GetEmailAuthenticationResetOptions()


If(ProfileOptions And 1) Then
    response.write("Resend has been enabled.")
Else
    response.write("Resend has not been enabled.")
End if
```

2.2.26. SSRPMProfile.GetSmsAuthenticationResetOptions

Purpose: Get the enabled options for the reset action of the SMS authentication.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4).

The interface method returns the mask number of all enabled options of the reset action of SMS authentication. See below for a table with all options with their corresponding mask number.

> Note: The mask number can be used within a logical And-expression to check if a certain option has been enabled. This is shown in the example (ASP Code) below, which checks if allow resend option has been enabled.

Mask table:

SSRPM Profile Option	Mask Number
Allow resend of PIN code	1
Limit number of resends	2
Allow reset without PIN code (if phone number is unknown)	4
E-mail authentication disabled when using COM object	8
E-mail authentication disabled when using Reset Wizard	16
E-mail authentication disabled when using Offline Logon	32
Show partial e-mail address of recipient	64

Syntax: *GetSmsAuthenticationResetOptions()*

Parameters: None.

Return value: The mask number of all enabled options.

Example (ASP code):

```
ProfileOptions = Profile.GetSmsAuthenticationResetOptions()

If(ProfileOptions And 1) Then
    response.write("Resend has been enabled.")
Else
    response.write("Resend has not been enabled.")
End if
```

2.2.27. SSRPMProfile.GetEmailPincodeLength

Purpose: Get the PIN code length of the e-mail PIN code.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the length of the PIN code that the user should enter.

Syntax: *GetEmailPincodeLength()*

Parameters: None.

Return value: The length of the PIN code that the user should enter.

Example (ASP code):

```
PinCodeLength = Profile.GetEmailPincodeLength()
```


2.2.28. SSRPMProfile.GetSmsPincodeLength

Purpose: Get the PIN code length of the SMSl PIN code.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the length of the PIN code that the user should enter.

Syntax: *GetSmsPincodeLength()*

Parameters: None.

Return value: The length of the PIN code that the user should enter.

Example (ASP code):

```
PinCodeLength = Profile.GetSmsPincodeLength()
```

2.2.29. SSRPMProfile.GetOptionsEx

Purpose: Get the enabled SSRPM Profile options.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4).

The interface method returns the mask number of all enabled SSRPM Profile options of the SSRPM Profile object. This number identifies which settings are enabled within the current SSRPM Profile. See below for a table with all SSRPM Profile options with their corresponding mask number.

Note: The mask number can be used within a logical And-expression to check if a certain SSRPM Profile option has been enabled. This is shown in the example (ASP Code) below, which checks if Account Blocking has been enabled.

Mask table:

SSRPM Profile Option	Mask Number
A single user cannot have the same answer for multiple questions	1
An answer cannot be a word which is in the corresponding question	2
Case-sensitive comparison of answers	4
Store answers with irreversible encryption	8
Check answers immediately	16
Show incorrect answer	32
Show password complexity rules	64
Hide answers in the SSRPM Enrollment Wizard and SSRPM Reset Wizard	128
Enable Account Blocking	256
Enable E-mail Notification	512
Add an Answer confirmation box to the SSRPM Enrollment Wizard to prevent typo's	1024
Warn users on that they need to re-enroll after a certain period.	2048
Force user enrollment and re-enrollment	4096
Enable minimum answer length	8192
Don't unlock password during reset	16384
Disable password reset	32768
Allow unlock account	65536
Use minimum question length for user defined questions	131072
Use random questions	262144
Don't enforce password history	524288
Show the user new questions upon error (in combination with random questions)	1048576
Enforce the maximum number of resets	2097152
Use 256 hashing for the answers	4194304
Allow offline logon	8388608
Use reversible encryption for the answers	16777216
Show the password complexity rules from PCM	33554432
Enable helpdesk Caller Identification	67108864
Notify end-users of certain events	134217728
Allow end-users to select advanced authentication method	268435456
Enabled/Disable change password functionality	536870912
Block account after a number of incorrect PIN code attempts	1073741824
Require authentication to enroll when using the browser client.	2147483648
Require authentication to re-enroll when using the browser client.	4294967296
Enabled AD Self Service functionality	8589934592
Enabled fuzzy answer comparison	17179869184

Note: See the "Administrator's Guide" for more information about the SSRPM Profiles options, of which the latest version is available on the *Tools4ever website* http://www.tools4ever.com.

Syntax: *GetOptionsEx()*

Parameters: None.

Return value: The mask number of all enabled SSRPM Profile options.

Example (ASP code):

```
ProfileOptions = Profile.GetOptionsEx()


If(ProfileOptions And 256) Then
    response.write("Account Blocking has been enabled.")
Else
    response.write("Account Blocking has not been enabled.")
End if
```

2.2.30. SSRPMProfile.GetMaximumCIVCharactersPerQuestion

Purpose: Get the maximum number of characters used for CIV validation per question.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the maximum number of maximum number of characters used for CIV validation per question.

Syntax: *GetMaximumCIVCharactersPerQuestion()*

Parameters: None.

Return value: Number.

Example (ASP code):

```
MaxChars = Profile.GetMaximumCIVCharactersPerQuestion()
```

2.2.31. SSRPMProfile.GetCIVSelectionLimit

Purpose: Get the limit of the selection bandwidth for a CIV character. Meaning a CIV character can be at most the nth characters of the answer.

Description:
An SSRPM Profile object must be created and retrieved from the SSRPM Service first with the GetUserData interface method (see: *SSRPM.GetUserData* on page 4). The interface method returns the limit of the selection bandwidth for a CIV character.

Syntax: *GetCIVSelectionLimit()*

Parameters: None.

Return value: Number.

Example (ASP code):

```
MaxSelectionLimit = Profile.GetCIVSelectionLimit()
```


3. SSRPM COM with Visual Basic Script

A common usage of SSRPM COM is using SSRPM COM in Visual Basic scripts. Visual Basic (VB) is a very general development environment to create Windows applications and scripts. The integration of VB script and SSRPM using SSRPM COM objects allows you to access SSRPM user information, like for instance: the applicable SSRPM Profile settings of a user.

Within this section, SSRPM COM with VB script is described with an example.

3.1. Example Visual Basic Script

This section contains an example of SSRPM COM with VB script. The goal of this example is to show all questions of an enrolled user, which this user has specified within the SSRPM enrollment process.

> Note: The script is kept as simple as possible to make it easier to understand and is meant for demonstration purposes.

The example VB script can be found in the '\Examples\VbScript' directory within the SSRPM Admin Console directory (which is by default: 'C:\Program Files\Tools4ever\SSRPM\Admin Console'), called: 'GetQuestions.vbs'.

See below for the complete example script:

```
Option Explicit

Dim SSRPM,QuestionArray,Message,RetVal
Set SSRPM = CreateObject("SSRPMCOM.SSRPM")
RetVal = SSRPM.Connect("LION",37946)

RetVal = SSRPM.GetResetData("CATS","John Smith",QuestionArray)

If RetVal = 0 Then

  For Each Question In QuestionArray
    Message = Message & Question & Chr(13)
  Next

  MsgBox Message,VbOkOnly,"SSRPM COM with VB Script Example"

Else

  MsgBox "An error occurred while getting questions from user (code: " ➔
          & RetVal & ").",VbOkOnly,"Error getting questions"

End If
```

In the next sections, parts of the scripts are shown with some comments for each section.

3.1.1. Script section: Connecting to the SSRPM Service

Before the question can be get and shown a connection with the SSRPM Service needs to be setup first. This is done by creating the 'SSRPM' COM object of SSRPM COM and calling the 'Connect' interface method of the object.

First, the needed variables are declared:

```
Dim SSRPM,QuestionArray,Message,RetVal
```

The 'SSRPM' variable will hold the SSRPM COM object. The variable 'QuestionArray' will hold an array of text values with all user questions. The 'Message' variable will hold the text value of all questions together and will be used to show all questions in a message box and the 'RetVal' variable will be used to store the return value which can be returned by an interface method.

The SSRPM COM object SSRPM is then created with the 'CreateObject' call:

```
Set SSRPM = CreateObject("SSRPMCOM.SSRPM")
```

The argument 'SSRPMCOM.SSRPM' specifies the SSRPM COM library and the type of object (SSRPM) of which an instance must be created. If the SSRPM COM library is not installed and/or registered, the 'CreateObject' call will fail.

The SSRPM COM object now connects to the SSRPM Service with the statement:

```
RetVal = SSRPM.Connect("LION",37946)
```

The 'Connect' interface method takes two arguments: the name of the computer that runs the SSRPM Service and the port number used by the SSRPM Service. 37946 is the default SSRPM Service port number.

3.1.2. Script section: Getting questions from the SSRPM Service

The SSRPM COM object is connected to the SSRPM Service. Now the questions can be retrieved from the SSRPM Service by calling the 'GetResetData' interface method of the SSRPM COM object. The SSRPM COM object gets all questions from the specified user:

```
RetVal = SSRPM.GetResetData("CATS","John Smith",QuestionArray)
```

The 'GetResetData' interface method takes three arguments: the name of the domain of which the enrolled user is a member of, the account name of the enrolled user and the declared 'QuestionArray' variable. On success, the return value of the 'GetResetData' interface method is zero and the 'QuestionArray' variable will contains a text array of all user questions.

A loop is implemented to copy each question from the returned question array within the 'QuestionArray' variable to one text variable ('Message'):

```
For Each Question in QuestionArray
  Message = Message & Question & Chr(13)
Next
```

Each question is separated with a new line, by using carriage return characters ('Chr(13)') between each question within the 'Message' variable. The 'Message' variable is used to show all questions in a message box:

```
MsgBox Message,VbOkOnly,"SSRPM COM with VB Script Example"
```


3.1.3. Testing and executing the script

To test the script in your environment, modify the Visual Basic script with your favorite editor. Make sure that the user and domain name is a user that has been enrolled into SSRPM and that the SSRPM COM object connects to the appropriate SSRPM Service.

To eventually execute the script, open a command prompt and enter the name of the script: *GetQuestions.vbs*.

This will automatically initialize Windows Scripting Host to execute the script. When the script executes successfully a message box will be prompted, which shows all questions of the specified user:

Figure 2: VB Script example output

4. SSRPM COM with IIS

Users can use SSRPM with an internet browser via SSRPM COM objects in Advanced Server Pages (ASP) with Internet Information Services (IIS).

In such an environment three main components are involved:

1. The internet browser that shows the web-pages and is used to enter input fields;
2. The IIS web-server, which hosts the ASP pages and to which the browser connects;
3. The SSRPM Service, contacted by the SSRPM COM objects used by the ASP pages running on the IIS web-server.

See the figure below how these components interact with each other:

Figure 3: SSRPM COM with IIS and ASP Pages

SSRPM is shipped with a fully functional web interface. Please refer to the document 'Web Interface Guide' for a complete list of features and how to install and configure the SSRPM Web Interface.

5. Appendix A: Component Object Model (COM)

The Component Object Model (COM) is a technology that is used by applications to interact with other applications. The COM technology is designed by Microsoft. The most important Microsoft applications that use COM are:

- Internet Information Services (IIS)
- Office applications
- Visual Basic Scripting (VB) and Visual Basic

5.1. COM objects and interfaces

In principle, a COM object is a piece of software that implements one or more functions. Different COM objects support different functions. The functions of a COM object are accessible by means of the interface of the COM object. The COM object itself is regarded as a black box that implements one or more functions accessible through its interfaces.

Applications that support COM can create COM objects. By accessing the interface functions of the COM object, the functions of the COM object are executed by the calling application. The syntax to create COM objects and access the interface functions of a COM object is extremely general: this is the main reason why COM objects can be used by so many different applications: The same COM object can be created and used in ASP-pages, Word documents and Visual Basic scripts.

All applications that support COM use some kind of programming or script language to implement COM. The procedure used is always the same:

1. The COM object is created;
2. The interface functions of the COM object are accessed;
3. Returned variables can be processed in the application.

An application can use multiple COM objects and COM objects can use other COM objects.

5.2. COM registration

In order for an application to use a COM object and the interface functions of the object, the COM object must be registered. Once a COM object is registered, all applications that support COM can use the COM object. In most cases, COM objects are registered automatically.

5.3. Type library

In most cases, the COM object code is contained in a file with the .DLL extension. Such a file contains all code needed to use the COM objects it implements. Besides the COM objects, the file can also contain a so called type library that describes the COM objects and the interfaces that are used to access the COM objects.

6. Index

T