HP Business Process Monitor

For the Windows ® operating system

Software Version: 9.23

BPM Monitoring Solutions Best Practices

Document Release Date: December 2013

Software Release Date: December 2013

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© Copyright 2013 Hewlett-Packard Development Company, L.P.

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

Intel®, Pentium®, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and other countries.

iPod is a trademark of Apple Computer, Inc.

Java is a registered trademark of Oracle and/or its affiliates.

Microsoft®, Windows®, Windows NT®, and Windows® XP are U.S registered trademarks of Microsoft Corporation.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

UNIX® is a registered trademark of The Open Group.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: **http://h20230.www2.hp.com/selfsolve/manuals**

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to: **http://h20229.www2.hp.com/passport-registration.html**

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support Online web site at: **http://www.hp.com/go/hpsoftwaresupport**

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

Chapter 1: Overview 6
Chapter 2: How to Use Script Parameters and Functions 7
Script Parameters 7
Script Functions 9
Chapter 3: Cloud Email Provider – Office 365 Transaction Flow 10
General Flow 10
Transaction — Office365_1_Login 12
Navigate to https://portal.microsoftonline.com 13
Click in the someone@example.com Textbox 14
Type LR.getParam('userName') in someone@example.com Textbox 15
Click in the Password Textbox 16
Type ********** in the Password Textbox 17
Click Sign in 18
Transaction — Office365_2_send mail 19
Click New Mail 20
Type LR.getParam('userName') in To Field 20
Click in Subject Field 21
Evaluate JavaScript Code LR.evalC('getRand'); 21
Type //LR.evalC('getRand'); L...am('randnum2'); in Subject Field 22
Click Send 22
Click Settings 23
Click Refresh 23
Transaction — Office365_3_delete mail 24
Evaluate JavaScript code var Sub1=LR.getParam('randnum2'); 25
Click ArgsContext.Sub1 26
Click Delete 27
Transaction — Office365_4_Calendar 28
Move Mouse over Calendar 29
Click Calendar 29
Transaction — Office365_5_Logoff 30
Click LR.getParam('userNameSurname') 31
Click Sign out 32
Chapter 4: Cloud Email Provider – Gmail Transaction Flow 33
General Flow 33
Transaction — Gmail_1_Login 35

Navigate to https://mail.google.c...che=2&emr=1 ... 36
Click Sign in ... 37
Click in Username Textbox ... 38
Type LR.getParam('userName') in Username Textbox ... 39
Type ********** in Password Textbox ... 40
Click Sign in ... 42
Transaction — Gmail_2_Send_mail ... 43
Click Compose ... 44
Type LR.getParam('userName') in To Field ... 45
Click Subject Field ... 46
Evaluate JavaScript code LR.evalC('getRand'); ... 47
Type LR.getParam('randnum2'); in Subject Field ... 48
Click Send ... 49
Transaction — Gmail_3_Delete ... 50
Click Refresh ... 51
Evaluate JavaScript code var Sub=LR.getParam('randnum2') ... 51
Click ArgsContext.Sub Decorator ... 52
Click Button (6) ... 53
Transaction — Gmail_4_Logoff ... 54
Click LR.getParam('userName') ... 55
Click Sign out ... 55
Chapter 5: Cloud CRM Provider – Salesforce Transaction Flow ... 56
General Flow ... 56
Transaction SalesForce_1_Login ... 58
Navigate to 'https://login.salesforce.com/' ... 59
Type LR.getParam('userName') in User Name Textbox ... 60
Type ********** in Password Textbox ... 62
Click Log in to Salesforce ... 63
Transaction SalesForce_2_Create account ... 64
Click Accounts ... 65
Click New ... 66
Click Account Name Textbox ... 67
Type LR.evalC('getRand'); var...randnum2'); Sub; in Account Name Textbox ... 68
Click Save ... 69
Transaction SalesForce_3_Delete account ... 70
Click Accounts ... 71
Evaluate JavaScript code var Sub1=LR.getParam('randnum2') ... 72
Click ArgsContext.Sub1 ... 73

Click Delete 74
Transaction SalesForce_4_Logoff 75
Click User Menu 76
Click Logout 77
Chapter 6: Using Your Scripts to Monitor Your Cloud Service Provider 78
Isolating Performance Issues with HP BPM 78
Isolating Availability Issues with HP BPM 81
Defining, Tracking, and Reporting on SLAs 83
Appendix A: About TruClient for IE Protocol 84
Appendix B: Time to Value 86
Use Firebug Lite 86
Standard Steps 86
Navigate 86
Click Object (Link, Textbox, Button, or Decorator) 87
Type in Text Object 88
Evaluate JavaScript 88
Variables and ID Methods – How to Combine 89
Meaning of Wait Step 89
Appendix C: Script Validation Concept 91

Chapter 1: Overview

This document contains case studies (Chapters 3-5) which provide explanations to customers of how to monitor their own paid-for-service cloud applications.

The goal of these case studies is to show how to establish end user monitoring on cloud applications using TruClient for Internet Explorer protocol. These case studies provide the basic instructions required to facilitate building end user monitoring solutions using WYSIWYG protocol.

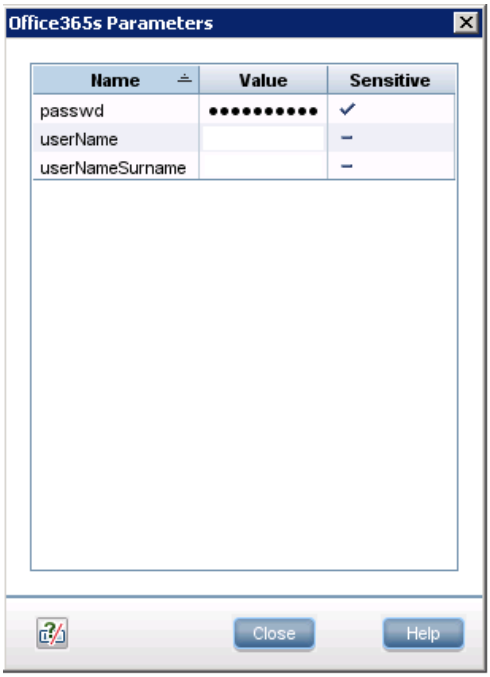
We recommend that you review Chapter 6 which describes how to use scripts to proactively monitor your cloud service provider.

Where applicable, the case studies are built around monitoring web applications. You can use TruClient for IE or Mobile protocol, depending on your business needs.

Chapter 2: How to Use Script Parameters and Functions

Script Parameters

Generally, parameters are used when values change frequently or you want to easily change them. Instead of changing the content of the script, you can just change the parameters. Script parameters defined in VuGen are seen in End User Management Administration in BSM.

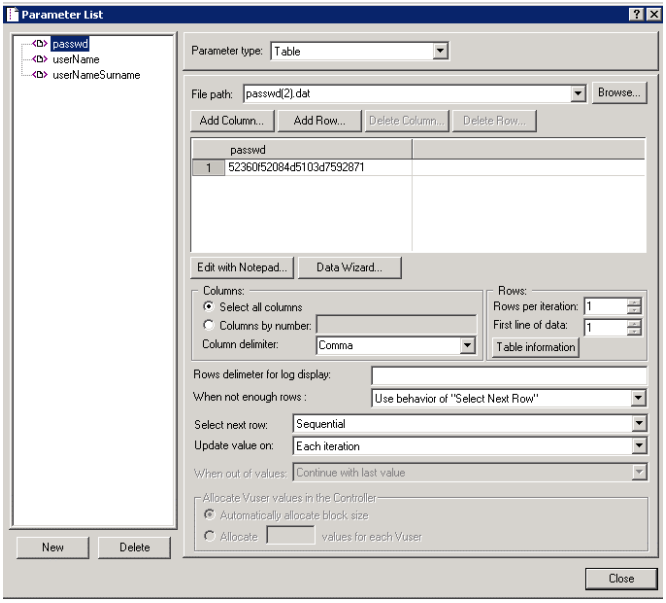


Script Parameters

To achieve this, create the parameters in VuGen. Notice that the value for the *passwd* parameter is marked as sensitive. Let us show you how we did this.

All parameters except *passwd* are plain text:

- **passwd** – Hidden and encrypted password.
- **userName** – Username in the form of an email address (user@domain.com).
- **userNameSurname** – This parameter is created because in some steps we need to click the full name of the user.



Password Parameter is Encrypted

To encrypt the password, we first need to use the Password Encoder:

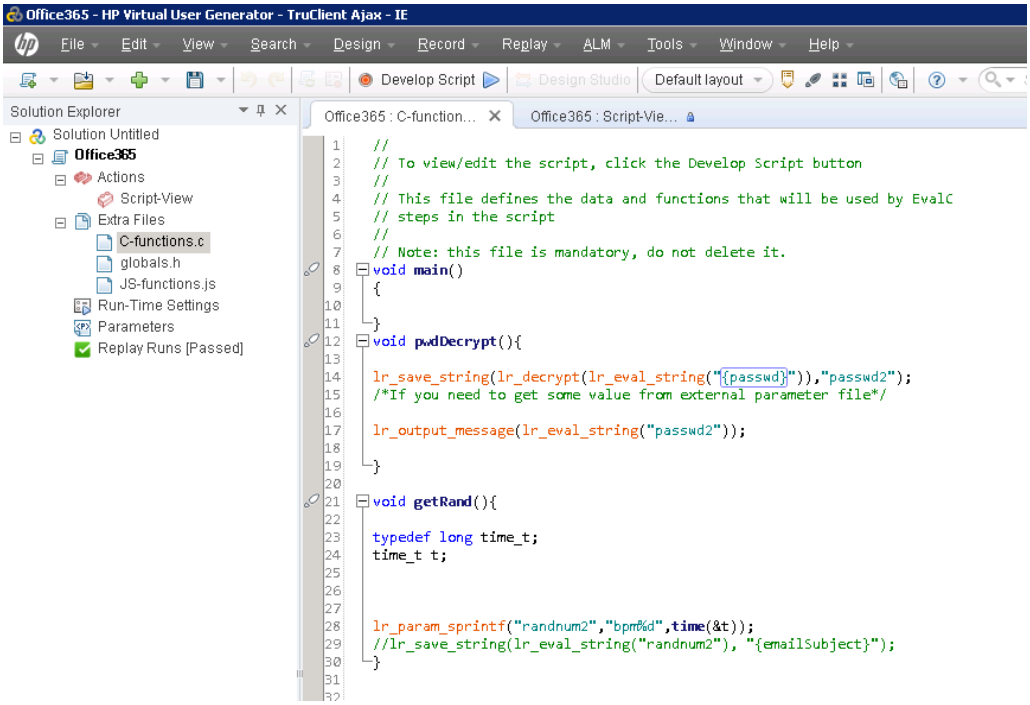
Password Encoder

The encoded string is then copied to the *passwd* value.

Script Functions

Functions are used to enhance a script. We use functions to achieve the following goals:

- Decrypt the parameter value and put it in a new parameter that is visible only within the script instance running on VuGen or the BPM instance.
- Generate a random number that is unique. In this example, we use Time as the random number.



Content of C-function.c File

To decrypt this code, we use a C function within the script. Functions are stored in the C-function.c file.

```
void pwdDecrypt(){
lr_save_string(lr_decrypt(lr_eval_string("{passwd}")),"passwd2");
}
```

The function *lr_eval_string* converts the *passwd* parameter to a string. Then *lr_decrypt* decrypts the string and *lr_save_string* copies the string as plain text to *passwd2*. This plain text password is contained within the boundaries of the script instance that is executed on the BPM instance.

Another function used is *getRand()*. It generates a unique Subject text that we can locate later and delete:

```
void getRand(){
typedef long time_t;
time_t t;
lr_param_sprintf("randnum2","bpm%d",time(&t));
}
```

These functions are used within the script.

Chapter 3: Cloud Email Provider – Office 365 Transaction Flow

This case study uses Microsoft Office 365 cloud service. The case study uses one flow/script with the following transactions:

1. **Login** – Open a login page, enter your credentials, and confirm that the first page loads.
2. **Create an email** – Create a new email and send it with a unique subject to the logged in user.
3. **Delete the email** – Check for this unique email, select it, and delete it.
4. **Calendar** – Click the Calendar link and verify that the calendar loads.
5. **Logoff** – Log off from the web application.

The main purpose of running this script is to confirm that the application's processes run successfully. If successful, the transactions complete without any errors.

You can view a sample Office365 script by downloading the following file: http://support.openview.hp.com/selfsolve/document/KM00658288/binary/BPM_Monitoring_Solutions-Office365.zip.

General Flow

Microsoft Office 365 is a subscription-based online office and software plus services suite which offers access to various services and software built around the Microsoft Office platform. In this flow, we test Microsoft Office 365's email and calendar functionality.

A sequence of transactions creates the business process flow. A transaction is a unit that is measurable by availability and performance. Therefore, we group activities that perform a specific transaction.

To describe and validate these transactions, logically group the transactions into the following steps:

1. Login.
   a. Navigate to https://portal.microsoftonline.com.
   b. Enter your credentials and confirm that the first page (Inbox) loads.
2. Create a new email.
   a. Click **New Mail**.
   b. Enter the email address used to login in step 1.
   c. Enter a unique subject.
   d. Send the email.
3. Delete the email.
   a. Refresh and check if the email arrived.
   b. Select the email.
   c. Delete the email.

4. Calendar
    a. Click the **Calendar** button and confirm that the Calendar page loads.
5. Logoff
    a. Log off from the web application

The following sections describe how to create these transactions.

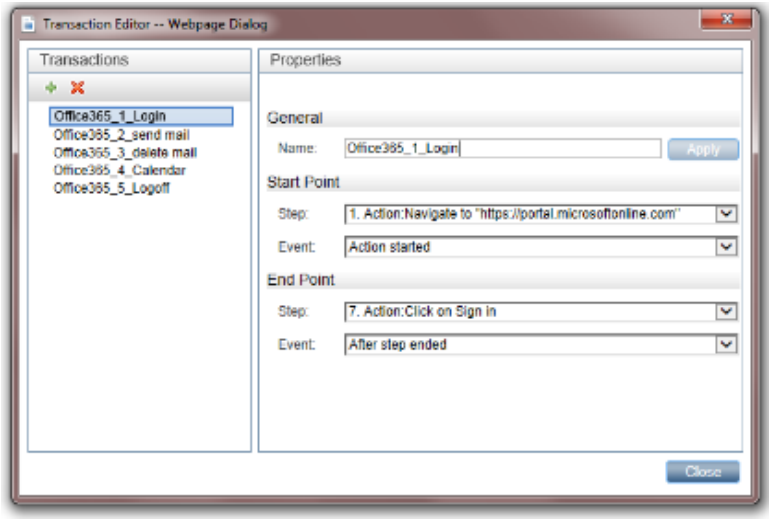
A good approach, before using this tool, is to know what we want to achieve. Create a blueprint as follows:

1. Run the Office 365 web application in Internet Explorer 9. Perform the steps of the flow and create annotations.
2. Determine the boundaries that form the transactions.
3. Select which values to assign to the parameters.
4. Select what values to protect and encrypt.

Record all the information you collect since it will come in handy later.

We recommend you start using the script. Refer to sections in this document for assistance.

Transaction — Office365_1_Login

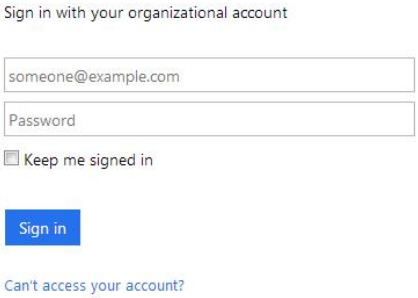


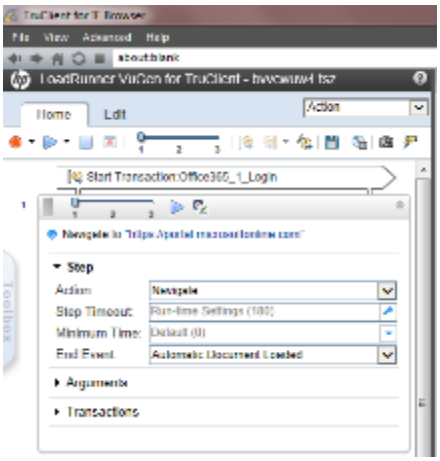
Define the name of the transaction. In this example, we call this transaction “Office365_1_Login”. Naming conventions should provide a clear view of the application, the transaction order, and create an overview of the business processes covered by this transaction.

Define the steps and events for the starting and ending points of the transaction. The most appropriate event for the start step is **Action started**. The most appropriate event for the end step is **After step ended**.

This transaction contains all the steps required for logging into the Office 365 application.

Navigate to https://portal.microsoftonline.com



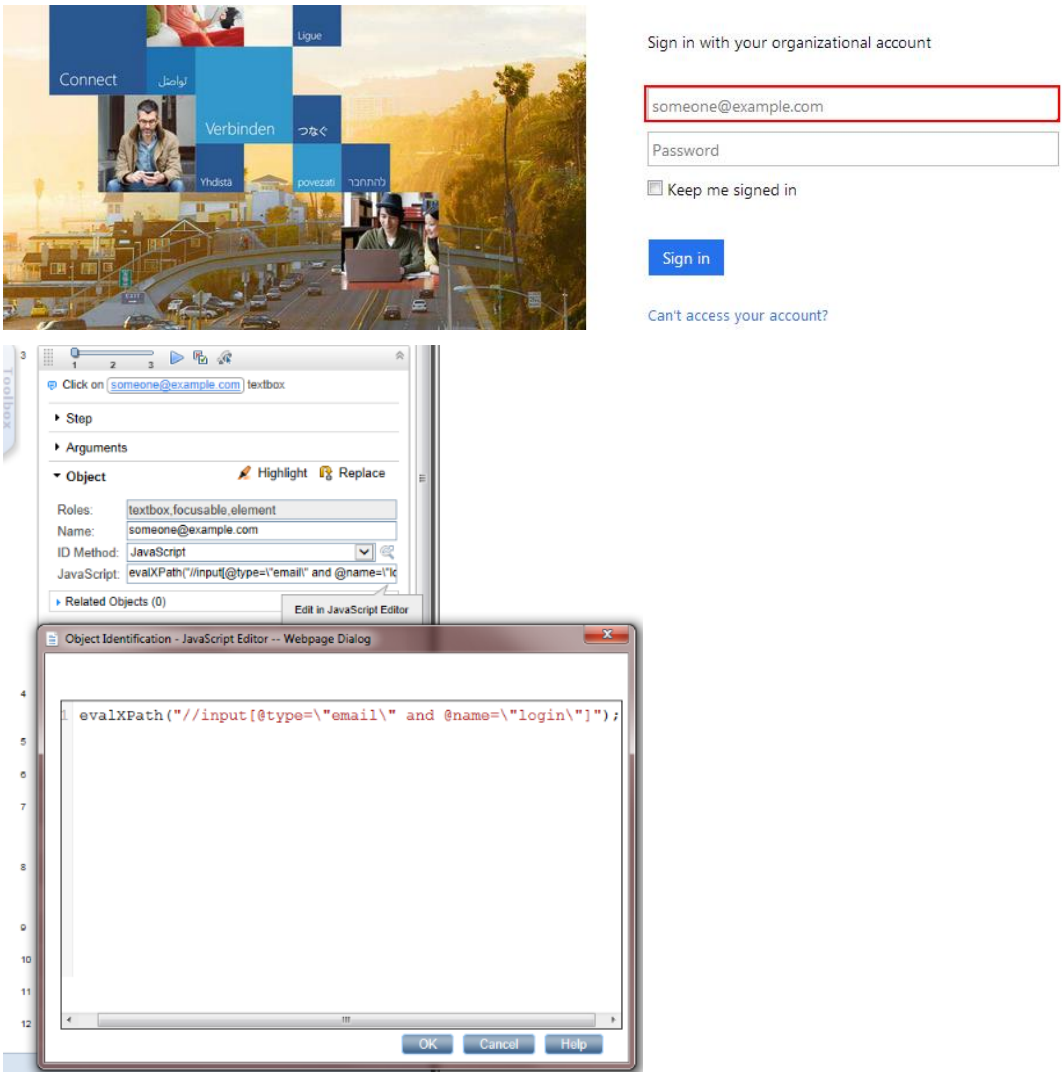


In the Navigation step, specify the URL.

While recording, VuGen automatically selects the appropriate End Event, in this case **Document downloaded**. The step ends when the process of loading a document completes. In most cases, you can freely select the default values.

If the pages load slowly, you can use a Wait step to make sure the page loads. For information on Wait steps, see *Meaning of Wait Step* on page 89.

Click in the someone@example.com Textbox



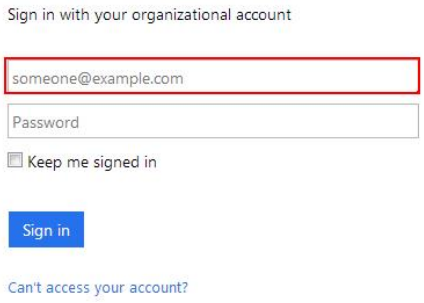
You can either select the **Automatic** or **JavaScript** ID method.

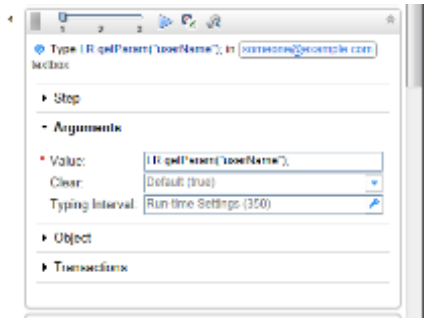
For the JavaScript ID Method, enter the following in the JavaScript field:

```
evalXPath("//input[@type=\"email\" and @name=\"login\"]");
```

There are cases when you will change the ID Method, as we will see later.

Type LR.getParam('userName') in someone@example.com Textbox

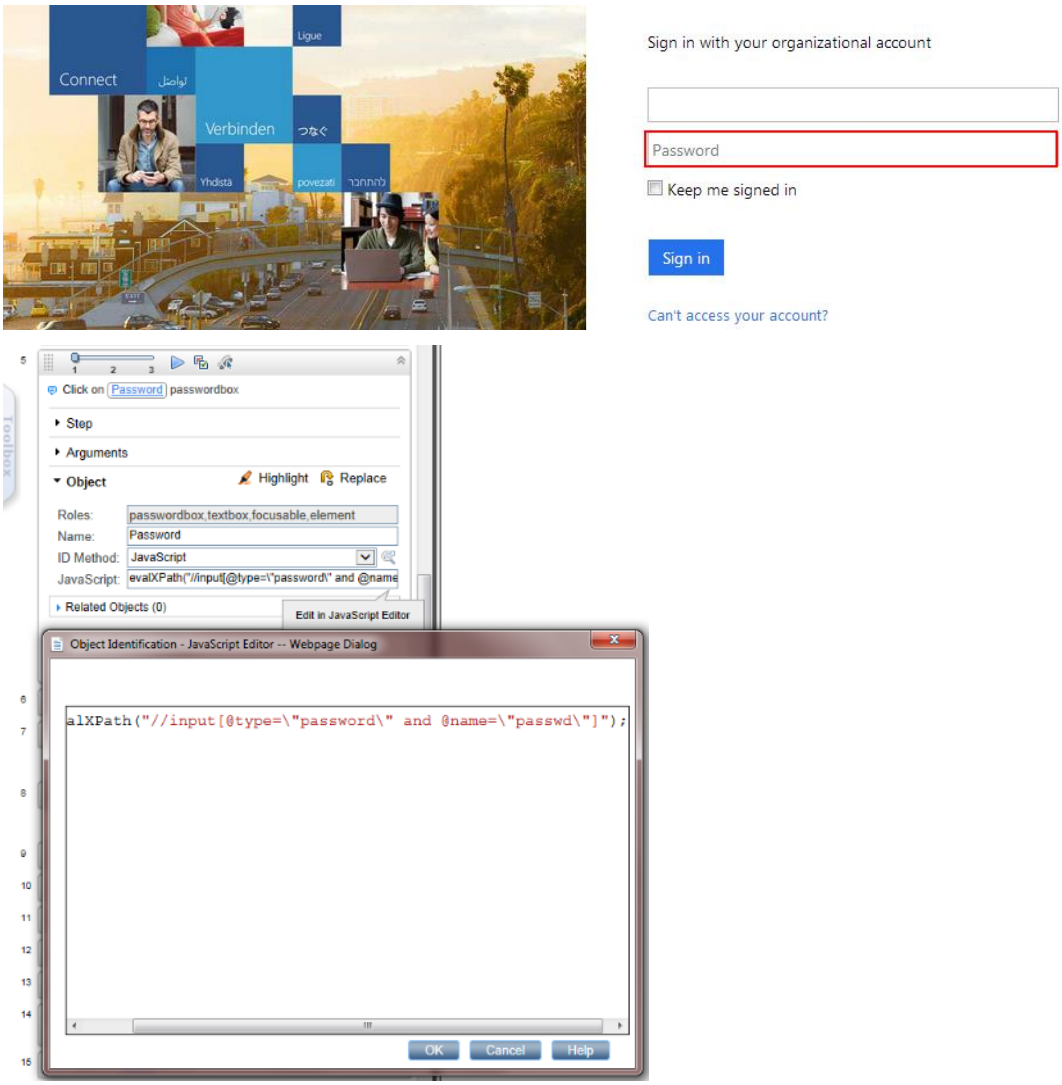




Type the username in the **someone@example.com** textbox.

Since we are using parameters, you can enter the *LR.getParam('userName');* command in the **Value** field and the result is the defined username.

Click in the Password Textbox

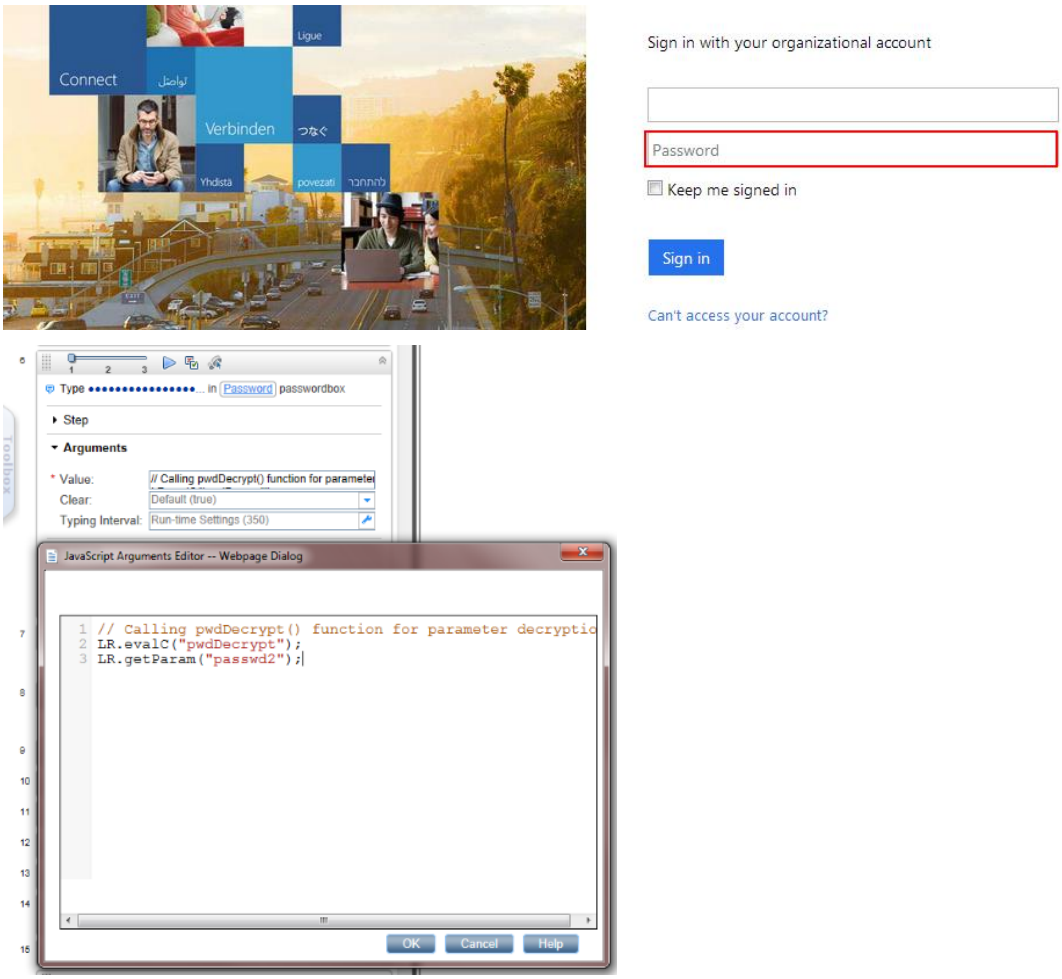


Similar to the *Click in the someone@example.com Textbox* step, you can select the **Automatic** or **JavaScript** ID method.

For the JavaScript ID Method, enter the following in the JavaScript field:

```
evalXPath("//input[@type=\"password\" and @name=\"passwd\"]");
```

Type ********** in the Password Textbox



Type a password in the **Password** textbox. To display an encrypted password, type the following:

```
// Calling pwdDecrypt() function for parameter decryption
LR.evalC("pwdDecrypt");
// This is a parameter with a decrypted value
LR.getParam("passwd2");
```

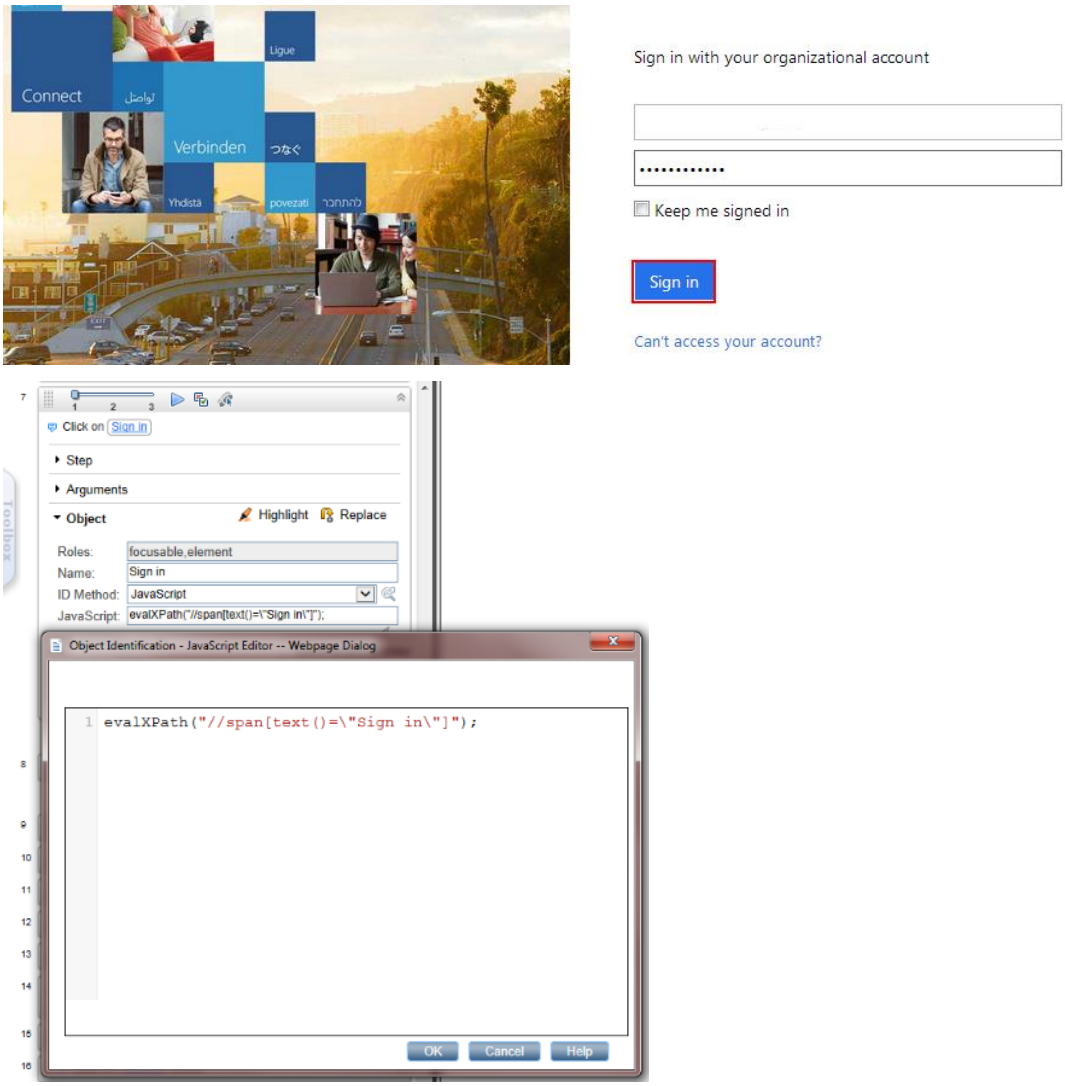
The pwdDecrypt() function is defined in C-function.c file:

```
void pwdDecrypt(){

lr_save_string(lr_decrypt(lr_eval_string("{passwd}")),"passwd2");
/*If you need to get some value from external parameter file*/
lr_output_message(lr_eval_string("passwd2"));
}
```

`passwd` is the parameter defined for external use (BSM – EUM) and is encrypted.

`passwd2` is the parameter seen only within this script instance.

Click Sign in



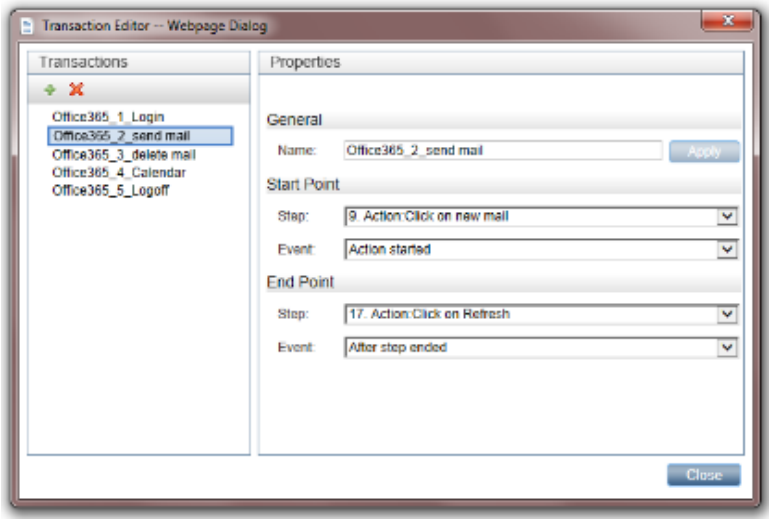
This is defined as the End step for Transaction Office365_1_Login.

The JavaScript ID Method used is:

evalXPath("//span[text()=\"Sign in\"]");

The End Event Step is set to automatic. So in this case, the End Event Step is Step Synchronous network completed. The step ends when all HTTP requests have completed excluding requests initiated by XMLHttpRequest.

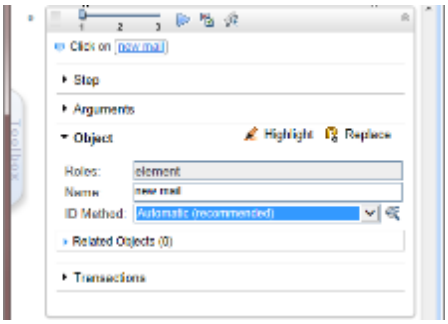
Transaction — Office365_2_send mail



The name of this transaction is “Office365_2_send mail”.

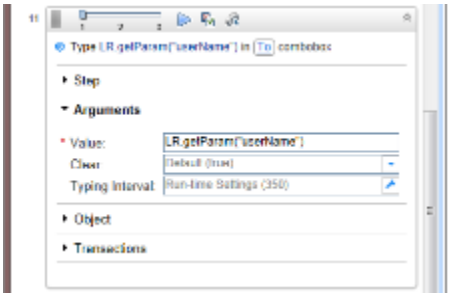
This transaction contains all the steps necessary for sending an email to the logged in user with a unique subject field.

Click New Mail



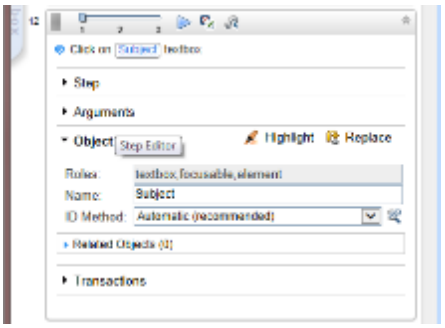
When you click **New Mail**, a form appears for sending a new email. By running the script multiple times, we can confirm that this element is recognized using the Automatic ID method.

Type LR.getParam('userName') in To Field



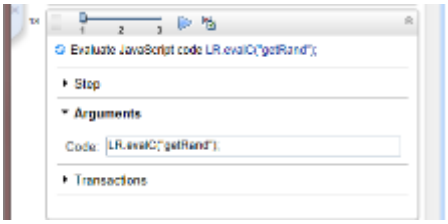
To make this script robust, we use parameters. If the username or password is changed at a later time, you do not need to change the script, only the parameter's value. The username is in the form of an email address.

Click in Subject Field



Click in the **Subject** field to select it.

Evaluate JavaScript Code LR.evalC('getRand');

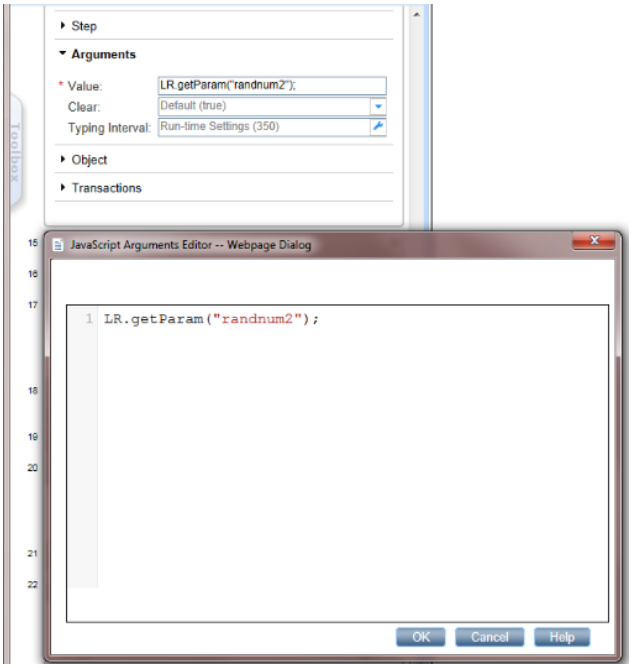


This is a very important step. By calling this function, we create a Subject value.

The *getRand()* function is defined in the C-function.c file:

```
void getRand(){
typedef long time_t;
time_t t;
lr_param_sprintf("randnum2","bpm%d",time(&t));
}
```

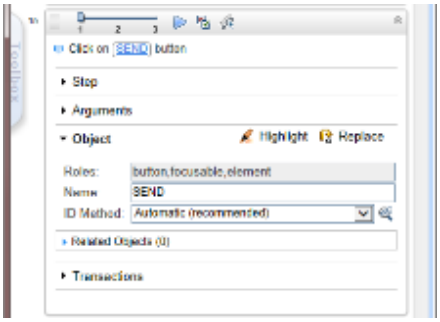
Type //LR.evalC('getRand'); L...am('randnum2'); in Subject Field



In this step, we acquire the value from the parameter:

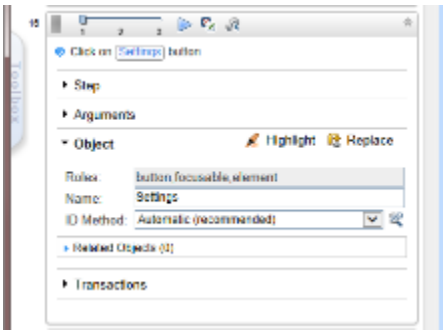
```
LR.getParam("randnum2");
```

Click Send



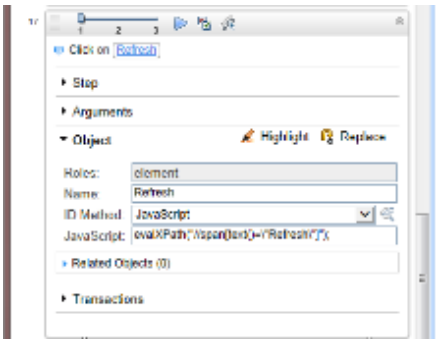
Click **Send**. The Send button is easy to recognize so there was no need for any additional corrections.

Click Settings



Click **Settings** to refresh the Inbox and check whether the email arrived.

Click Refresh

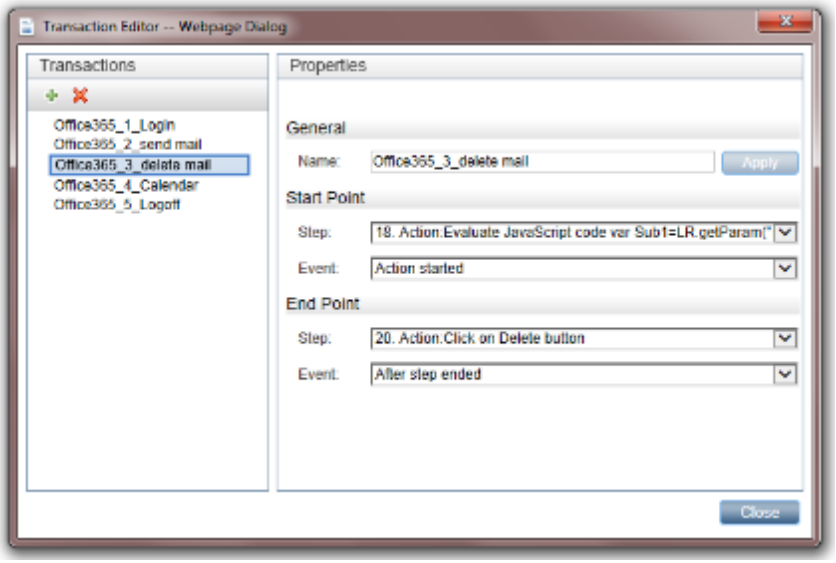


After clicking **Refresh**, the transaction completes and we confirm that the email arrived. If the email still has not arrived, repeat the previous step (**Click Settings**) and this step (**Click Refresh**).

We use the JavaScript ID Method:

```
evalXPath("//span[text()=\"Refresh\"]");
```

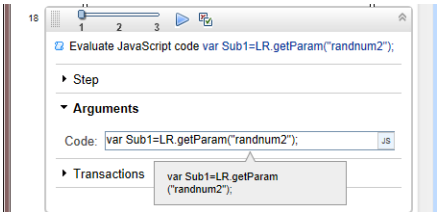
Transaction — Office365_3_delete mail



The name of this transaction is “Office365_3_delete mail”.

This transaction contains all the steps necessary for checking and deleting an email that was sent in the current interaction.

Evaluate JavaScript code var Sub1=LR.getParam('randnum2');

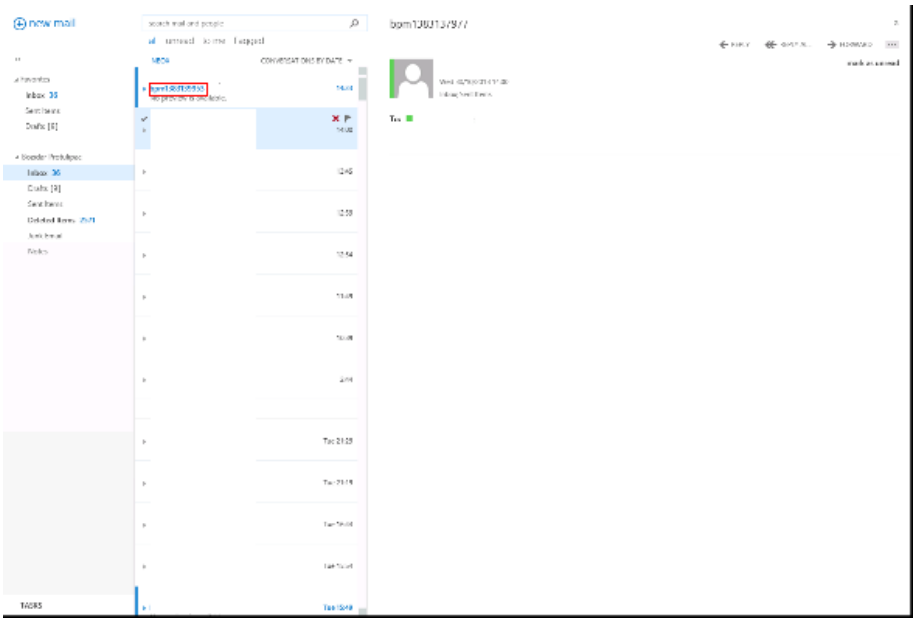


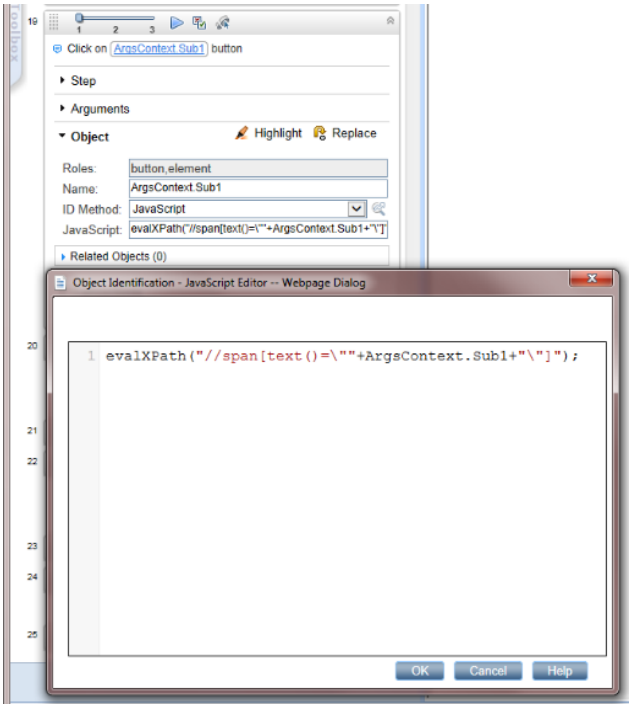
Create the *Sub1* variable and get its value from the parameter *randum2*:

```
var Sub1=LR.getParam("randnum2");
```

This variable is used to check (identify) and delete a specific email sent within this interaction. The life span of this variable is within the page. When you change the page, you must re-evaluate this JavaScript if you need a value from the *randum2* parameter.

Click ArgsContext.Sub1





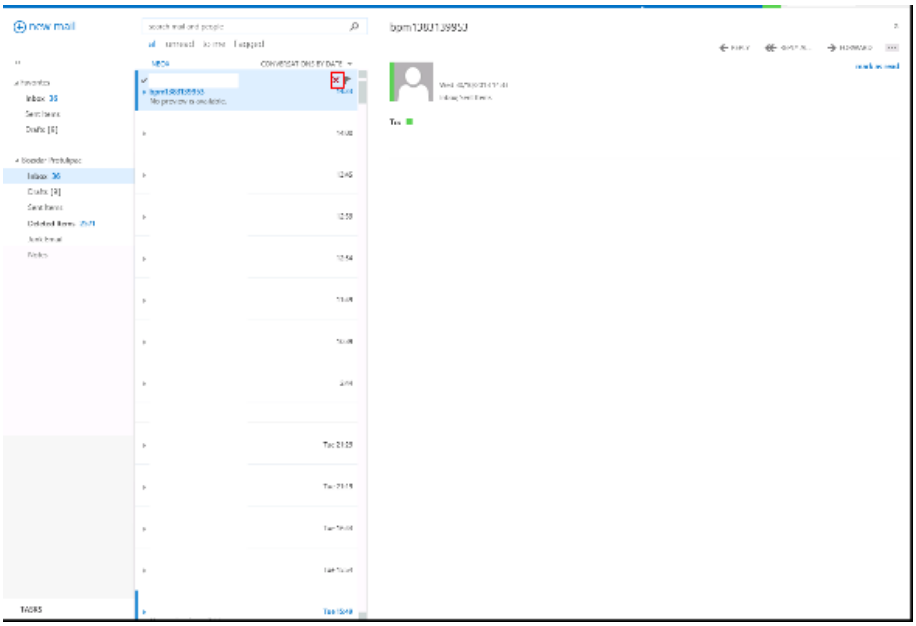
To identify an object, the name must be set as *ArgsContext.Sub1*

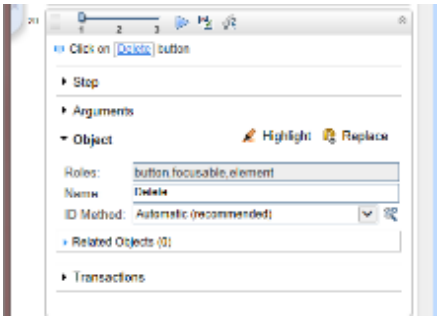
The ID Method is JavaScript:

```
evalXPath("//span[text()=\""+ArgsContext.Sub1+"\"]");
```

The specific email is found.

Click Delete

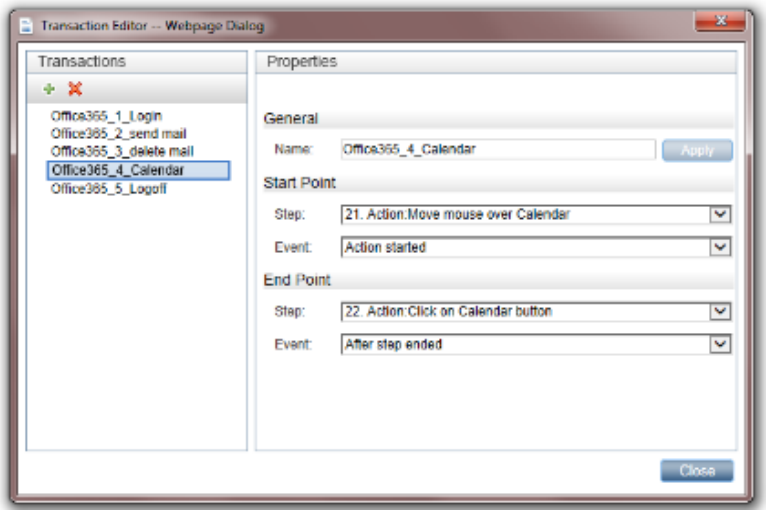




The **Delete** button is easy to recognize.

This step completes Office365_3_delete mail.

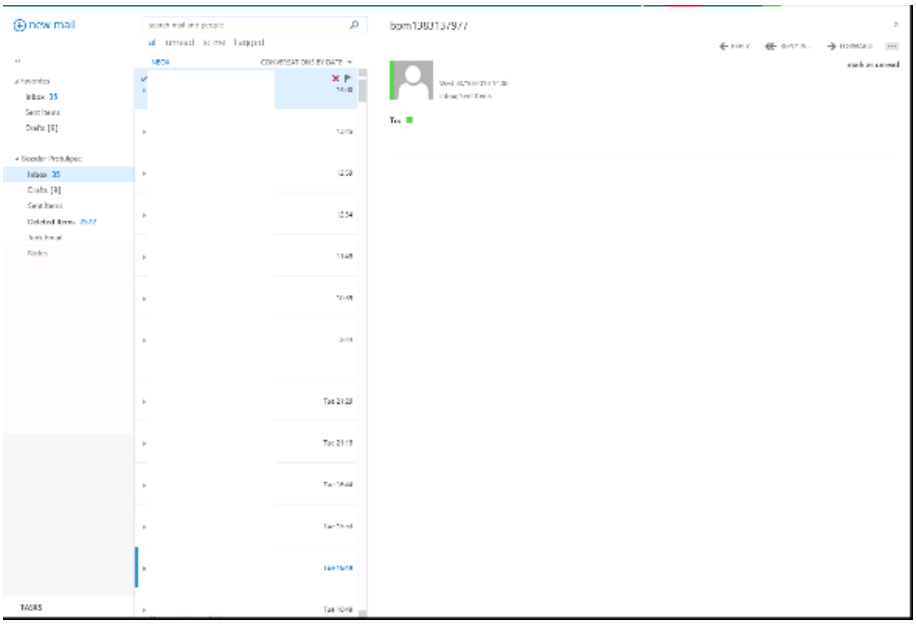
Transaction — Office365_4_Calendar

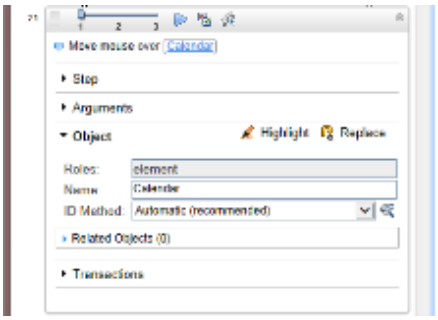


The name of this transaction is “Office365_4_Calendar”.

This transaction contains the steps necessary for confirming that the calendar loads.

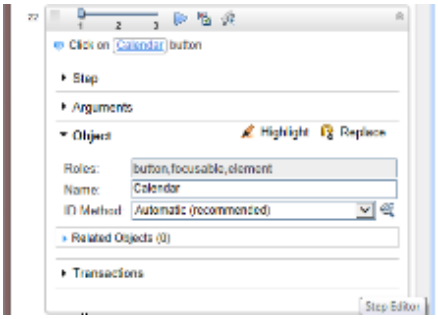
Move Mouse over Calendar





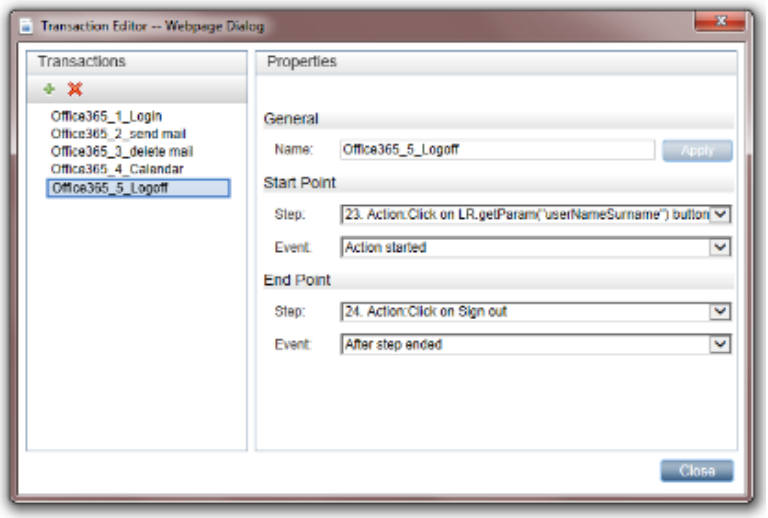
This section displays how to create an extra step (mouse over) before clicking an element.

Click Calendar



This is the end of transaction “Office365_4_Calendar.

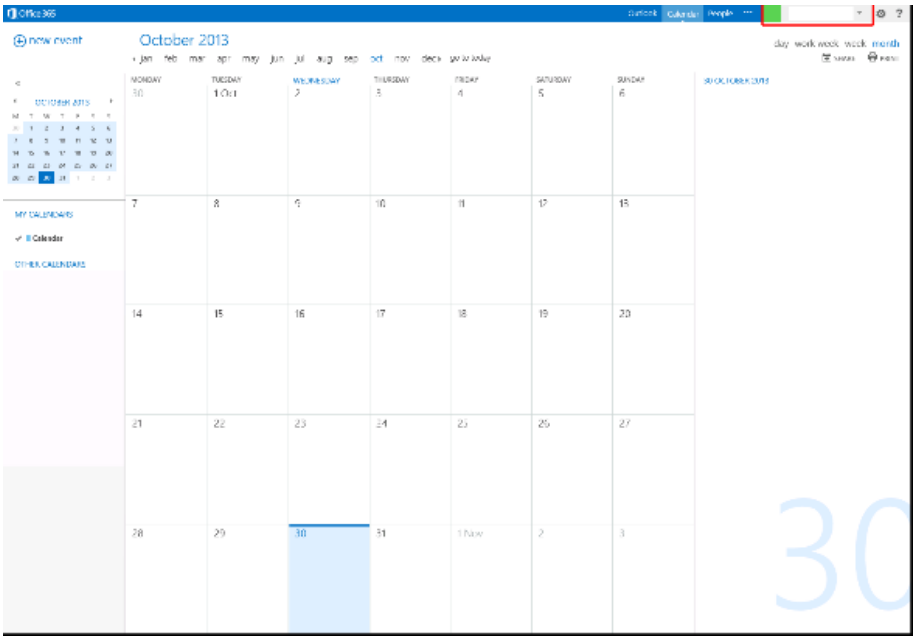
Transaction — Office365_5_Logoff



The name of this transaction is “Office365_5_Logoff”.

This transaction contains all the steps necessary for logging off.

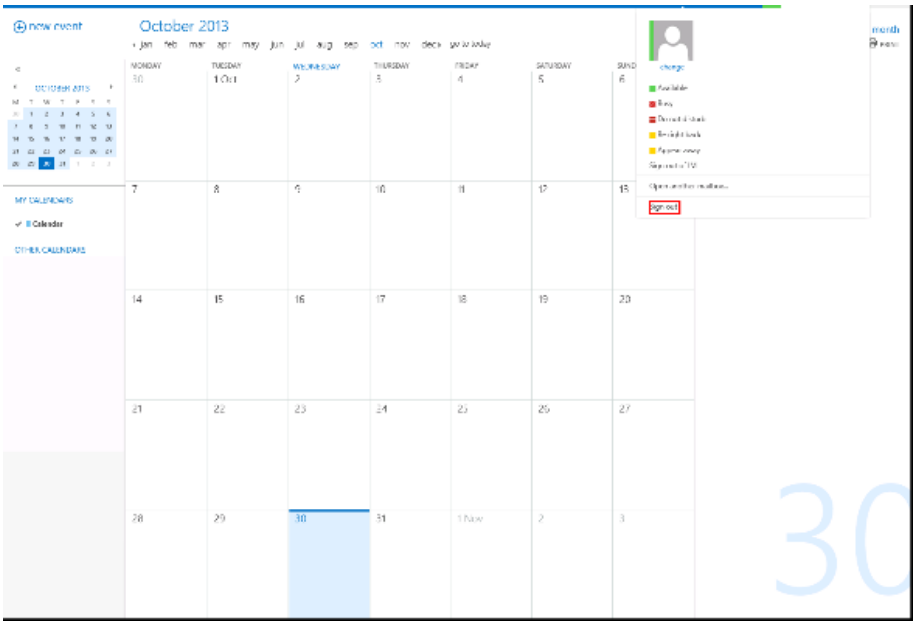
Click LR.getParam('userNameSurname')

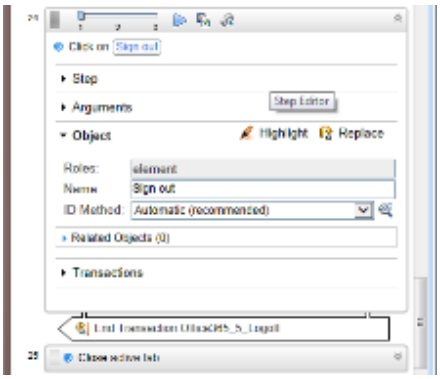


We used another parameter because for logging out of this application you must first click the full user name stored in the parameter *userNameSurname*.

```
LR.getParam("userNameSurname")
```

Click Sign out





By clicking **Sign out**, we end this transaction and script.

All we need to do now is close the active tab.

Chapter 4: Cloud Email Provider – Gmail Transaction Flow

This case study uses Gmail cloud service. The case study uses one flow/script with the following transactions:

1. **Login** – Open a login page, enter your credentials, and confirm that the first page loads.
2. **Create mail** – Create a new email, and send it with a unique subject to the logged in user.
3. **Delete mail** – Check for this unique email, select it, and delete it.
4. **Logoff** – Log off from the web application.

The main purpose of running this script is to confirm that the application's processes run successfully. If successful, the transactions complete without any errors.

You can view a sample Gmail script by downloading the following file: http://support.openview.hp.com/selfsolve/document/KM00658288/binary/BPM_Monitoring_Solutions-Google_Mail.zip

General Flow

Google Mail is another name for Google's email tool, Gmail. Gmail is a free, full-featured email service. Anyone can register for an account. Gmail is also available as part of Google Apps. In this flow, we test the email functionality of Gmail.

A sequence of transactions creates the business process flow. A transaction is a unit that is measurable by availability and performance. Therefore, we group activities that perform a specific transaction.

To describe and validate these transactions, logically group the transactions into the following steps:

1. Login.
   a. Navigate to https://gmail.com.
   b. Enter your credentials and confirm that the first page (Inbox) loads.
2. Create a new email.
   a. Click **Compose**.
   b. Enter the email address used to login in step 1.
   c. Enter a unique subject.
   d. Send the email.
3. Delete the email.
   a. Refresh and check if the email arrived.
   b. Select the email.
   c. Delete the email.
4. Logoff

a. Log off from the web application

The following sections describe how to create these transactions.

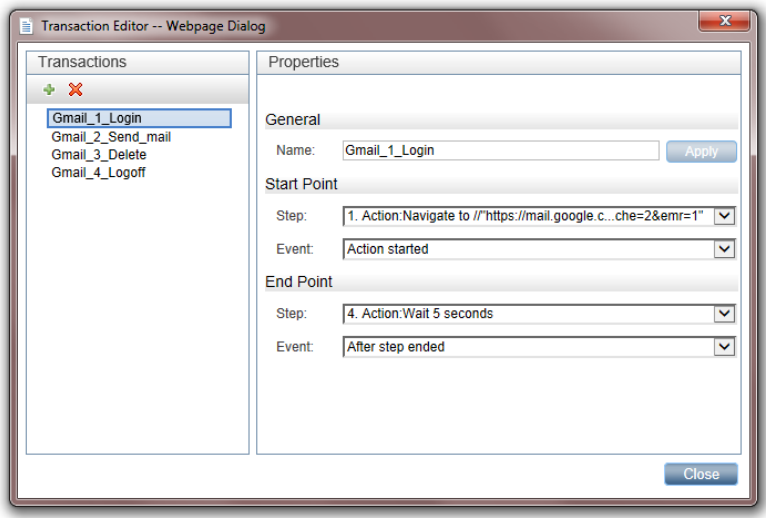
A good approach, before using this tool, is to know what we want to achieve. Create a blueprint as follows:

1. Run the Gmail web application in Internet Explorer 9. Perform the steps of the flow and create annotations.
2. Determine the boundaries that form the transactions.
3. Select which values to assign to the parameters.
4. Select what values to protect and encrypt.

Record all the information you collect since it will come in handy later.

We recommend you start using the script. Refer to sections in this document for assistance.

Transaction — Gmail_1_Login

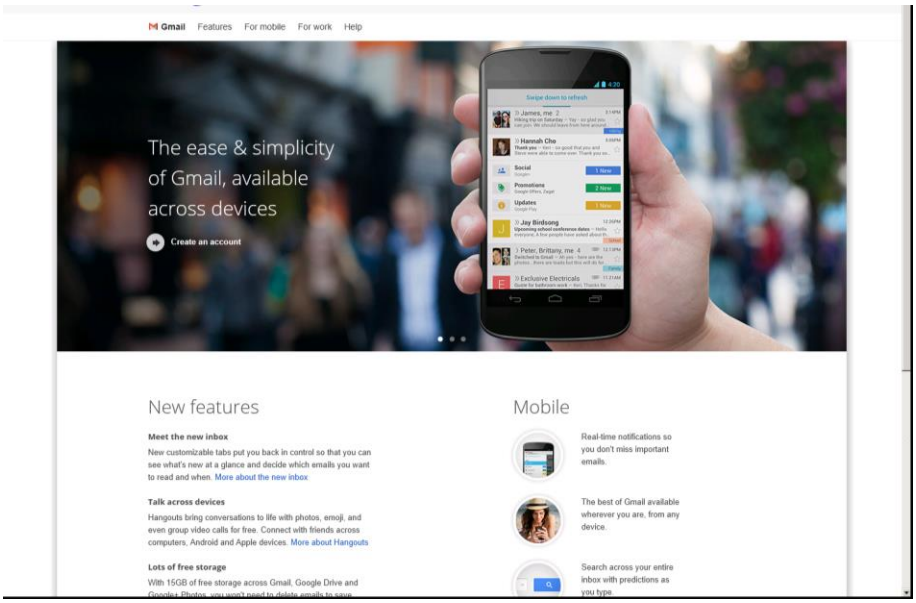


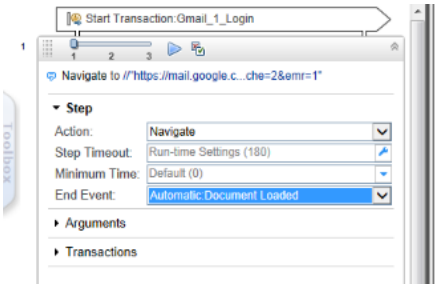
Define the name of the transaction. In this example, we call the transaction “Gmail_1_Login”. Naming conventions should provide a clear view of the application, the transaction order, and create an overview of the business processes covered by this transaction.

Define the steps and events for the starting and ending points of the transaction. The most appropriate event for the start step is **Action started**. The most appropriate event for the end step is **After step ended**.

This transaction contains all the steps required for logging into the Gmail application.

Navigate to https://mail.google.c...che=2&emr=1



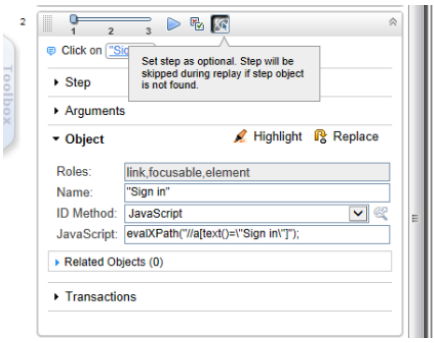


In the Navigation step, specify the URL.

While recording, VuGen automatically selects the appropriate End Event, in this case **Document downloaded**. The step ends when the process of loading a document completes. In most cases, you can freely select the default values.

If the pages load slowly, you can use a Wait step to make sure the page loads. For information on Wait steps, see *Meaning of Wait Step* on page 89.

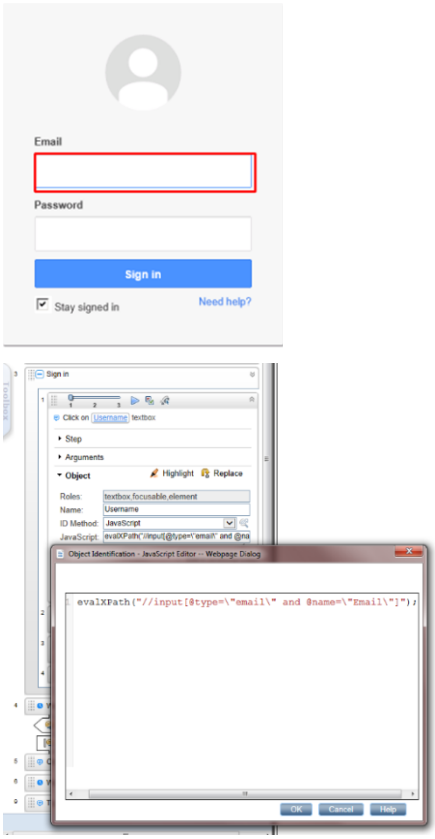
Click Sign in



Sometimes the application does not go straight to the login page. Therefore, insert this optional step, just in case. If the **Sign in** link does not appear, this step is ignored.

Note: Since providers make changes in their services, you may need to make changes to your scripts to maintain the same outcome.

Click in Username Textbox



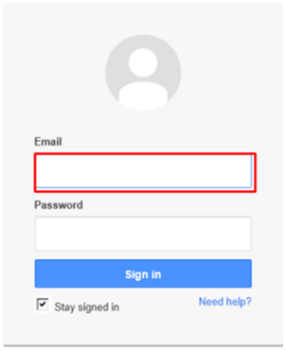
You can either select the **Automatic** or **JavaScript** ID method.

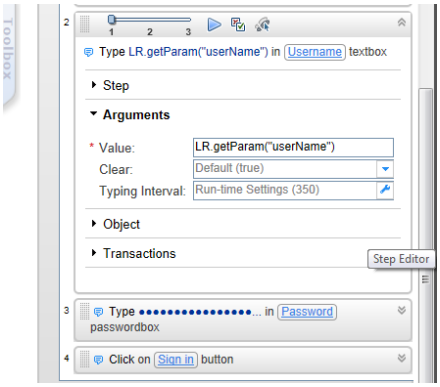
For the JavaScript ID Method, enter the following in the JavaScript field:

```
evalXPath("//input[@type=\"email\" and @name=\"Email\"]");
```

There are cases when you will change the ID Method, as we will see later.

Type LR.getParam('userName') in Username Textbox

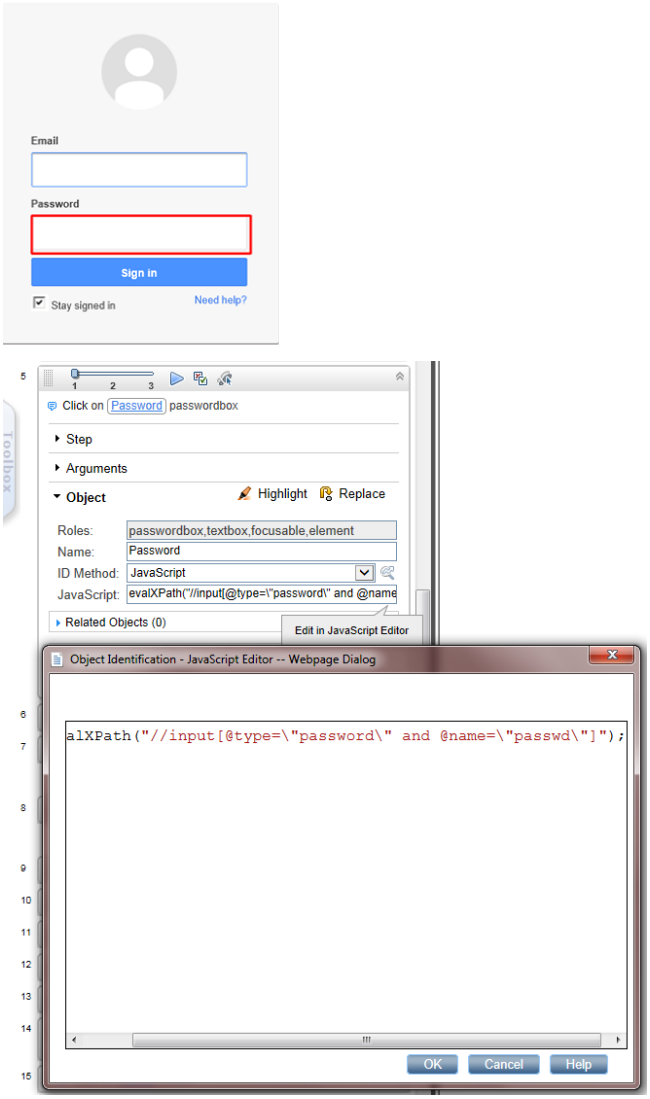




Type the username to the Username textbox.

Since we are using parameters, you can enter the *LR.getParam("userName");* command in the **Value** field and the result is the defined username.

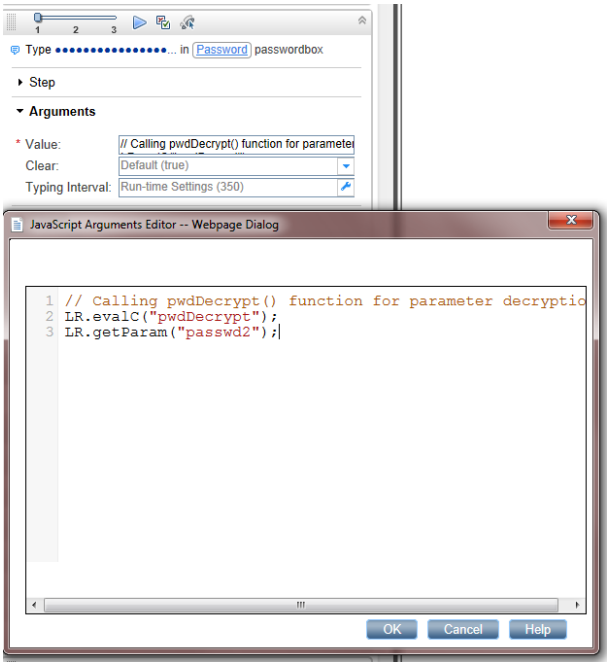
Type ********** in Password Textbox



Similar to the *Click in Username Textbox* step, you can select the **Automatic** or **JavaScript** ID method.

For the JavaScript ID Method, enter the following in the JavaScript field:

```
evalXPath("//input[@type=\"password\" and @name=\"passwd\"]");
```



Type a password in the **Password** textbox. To display an encrypted password (so no one can see it), type the following:

```
// Calling pwdDecrypt() function for parameter decryption
LR.evalC("pwdDecrypt");
// This is a parameter with a decrypted value
LR.getParam("passwd2");
```

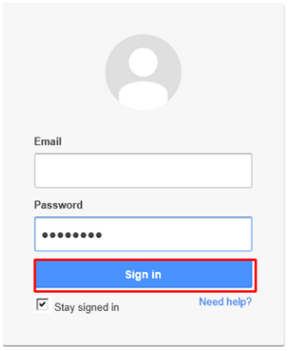
The **pwdDecrypt()** function is defined in C-function.c file:

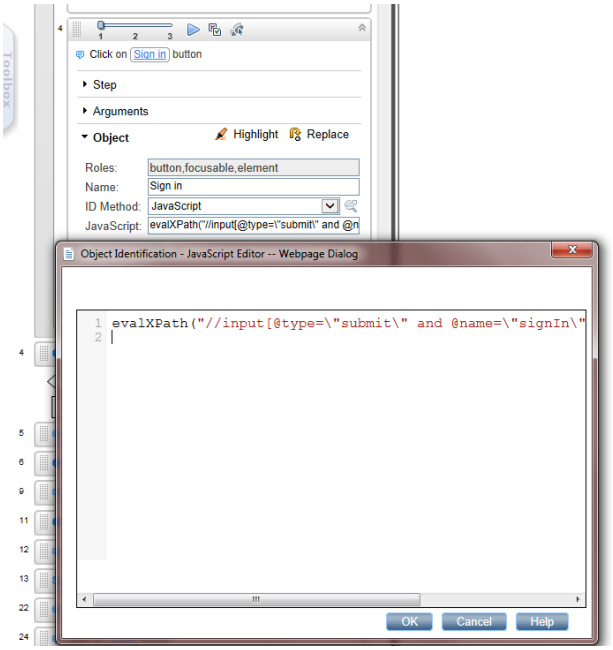
```
void pwdDecrypt(){

lr_save_string(lr_decrypt(lr_eval_string("{passwd}")),"passwd2");
/*If you need to get some value from external parameter file*/
lr_output_message(lr_eval_string("passwd2"));
}
```

`passwd` is the parameter defined for external use (BSM – EUM) and is encrypted.

`passwd2` is the parameter seen only within this script instance.

Click Sign in





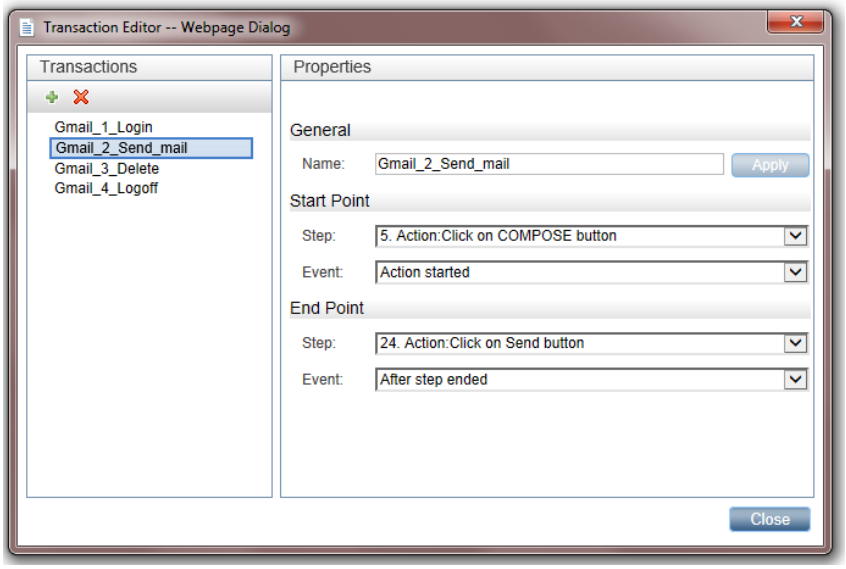
This is defined as the End step for Transaction “Gmail_1_Login”.

The JavaScript ID Method used is:

```
evalXPath("//input[@type=\"submit\" and @name=\"signIn\" and
@value=\"Sign in\"]");
```

The End Event Step is set to automatic. So in this case, the End Event Step is **Step Synchronous network completed**. The step ends when all HTTP requests have completed excluding requests initiated by XMLHttpRequest.

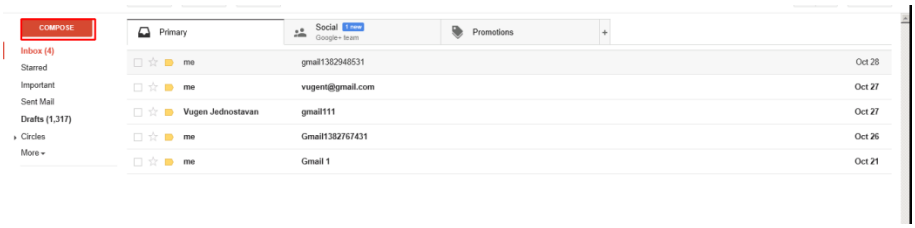
Transaction — Gmail_2_Send_mail

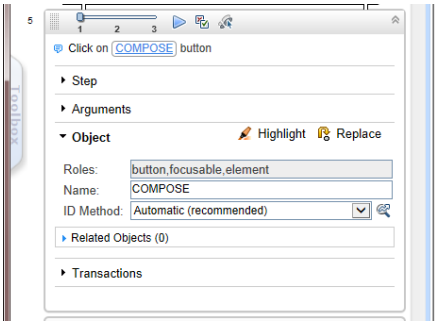


The name of this transaction is “Gmail_2_Send_mail”.

This transaction contains all the steps necessary for sending an email to the logged in user with a unique subject field.

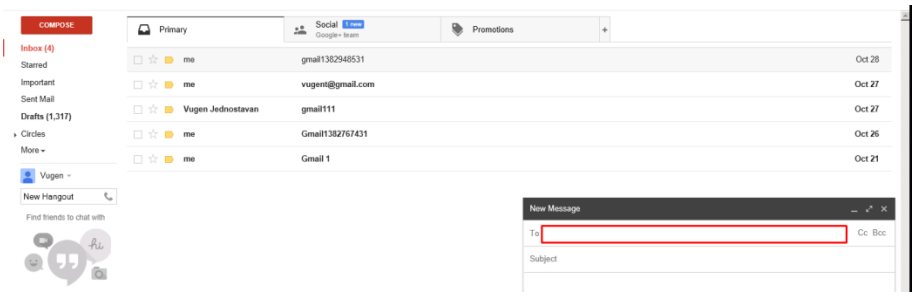
Click Compose

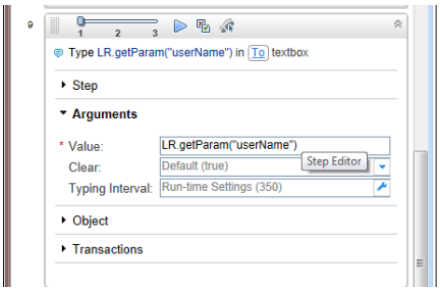




When you click **Compose**, a form appears for sending a new email. By running a script multiple times, we can confirm that this element is recognized using the Automatic ID method.

Type LR.getParam('userName') in To Field



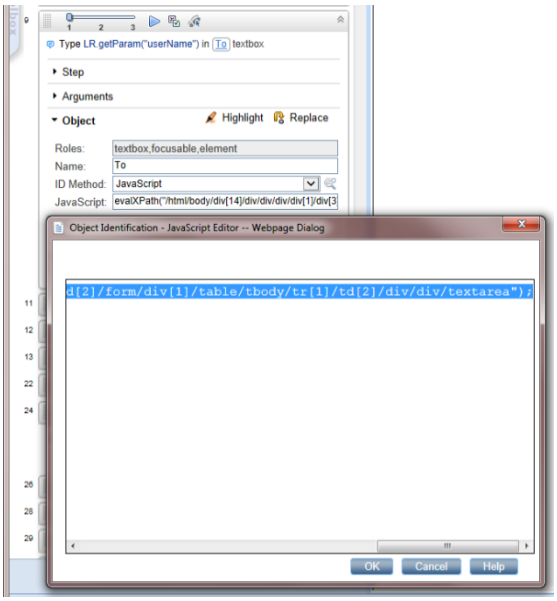


To make this script robust, we use parameters. Therefore, if the username or password is changed at a later time, you do not need to change the script, only the parameter's value. The username is in the form of an email address.

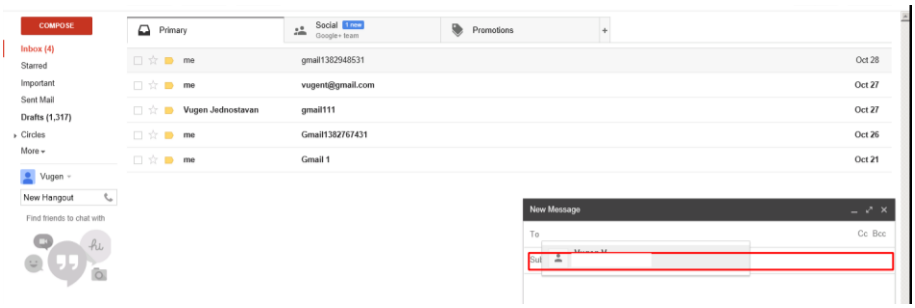
The ID method used in this example is JavaScript:

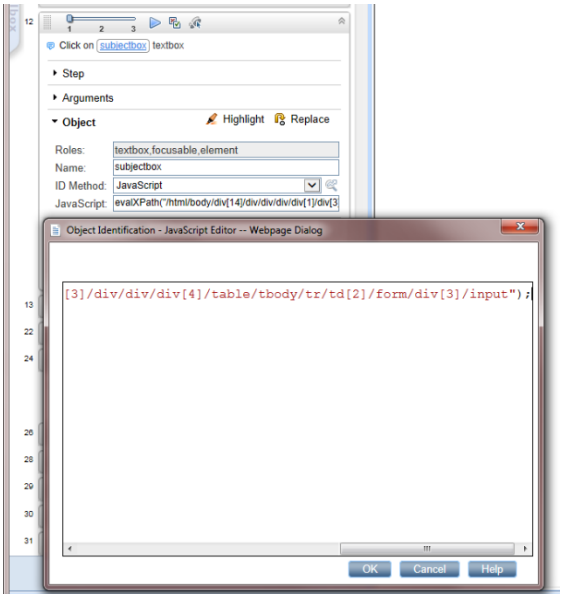
```
evalXPath("/html/body/div[14]/div/div/div/div[1]/div[3]/div[1]/div[1]
/div/div/div/div[3]/div/div/div[4]/table/tbody/tr/td[2]/form/div[1]/t
able/tbody/tr[1]/td[2]/div/div/textarea");
```

But this is not a good example, so it is better to use Firebug.



Click Subject Field





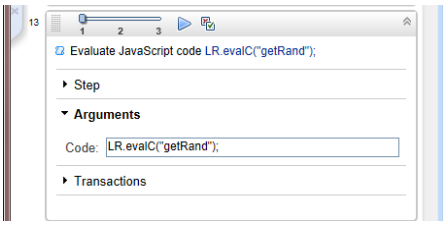
Click in the Subject field to select it. The ID method used is JavaScript:

```
evalXPath("/html/body/div[14]/div/div/div/div[1]/div[3]/div[1]/div[1]
/div/div/div/div[3]/div/div/div[4]/table/tbody/tr/td[2]/form/div[3]/i
nput");
```

But this is not a good example, therefore, it is better to use Firebug usage:

```
evalXPath("//input[@placeholder=\"Subject\" and
@name=\"subjectbox\"]");
```

Evaluate JavaScript code LR.evalC('getRand');

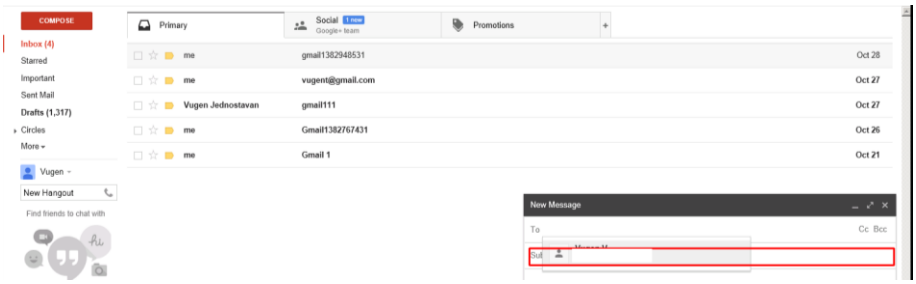


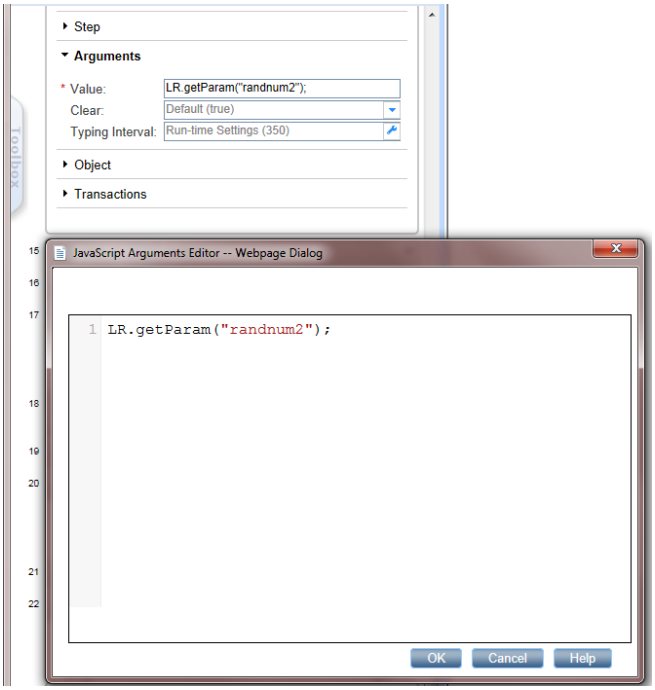
This is a very important step. By calling this function, we create a Subject value.

The *getRand()* function is defined in C-function.c file:

```
void getRand(){
typedef long time_t;
time_t t;
lr_param_sprintf("randnum2","bpm%d",time(&t));
}
```

Type LR.getParam('randnum2'); in Subject Field

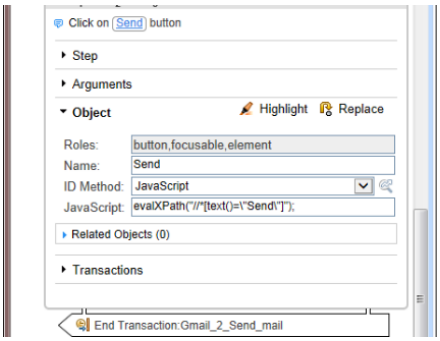




In this step, we acquire the value from the parameter:

```
LR.getParam("randnum2");
```

Click Send

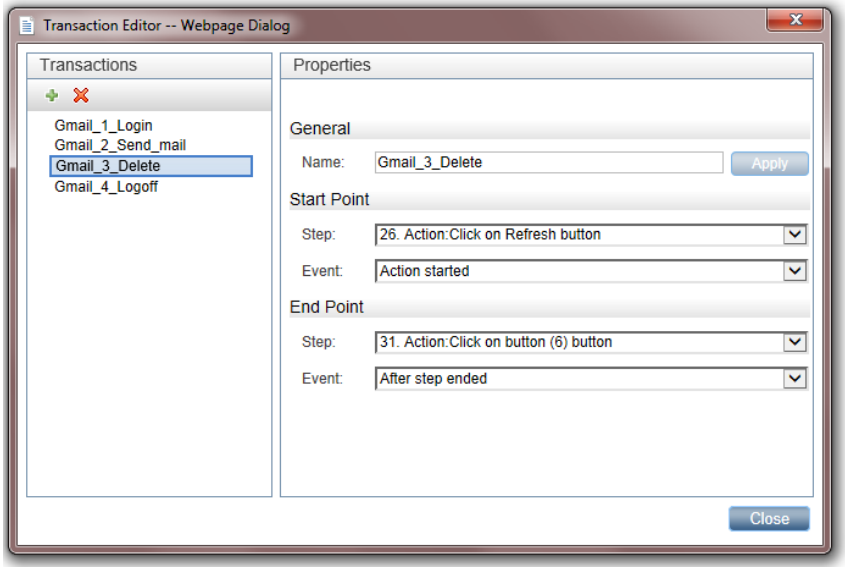


Click **Send**. The Send button is easy to recognize so there was no need for any additional corrections.

The ID Method is JavaScript:

```
evalXPath("//*[text()=\"Send\"]");
```

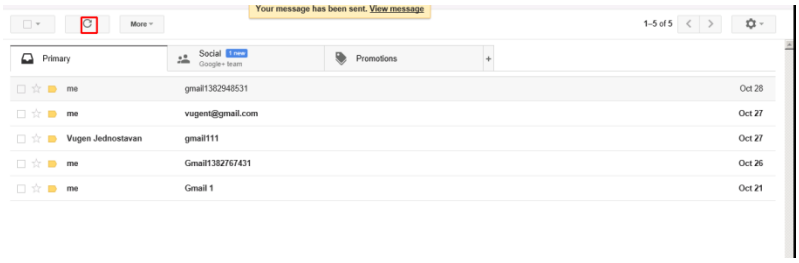
Transaction — Gmail_3_Delete

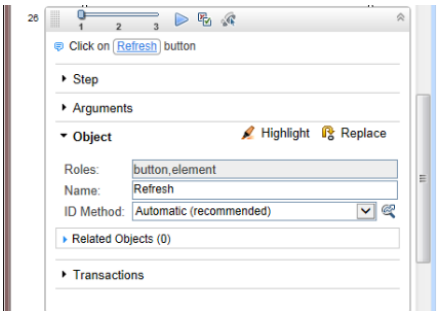


The name of this transaction is “Gmail_3_Delete”.

This transaction contains all the steps necessary for checking and deleting an email that was sent in the current interaction.

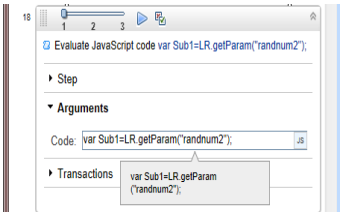
Click Refresh





If the email still has not arrived, repeat the previous step (**Click Settings**) and this step (**Click Refresh**).

Evaluate JavaScript code var Sub=LR.getParam('randnum2')

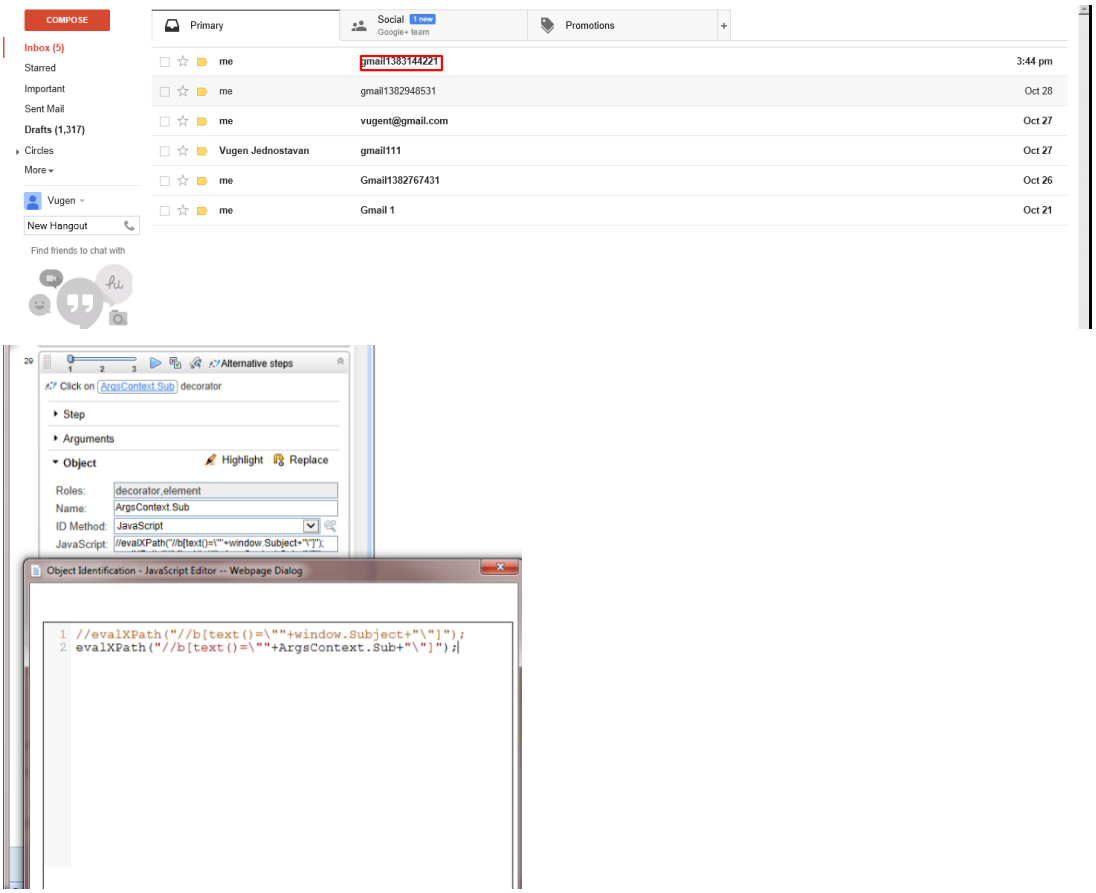


Create the *Sub1* variable and get its value from the parameter *randum2*:

```
var Sub=LR.getParam("randnum2");
```

This variable is used to check (identify) and delete a specific account. The life span of this variable is within the page. When you change the page, you must re-evaluate this JavaScript if you need a value from the *randum2* parameter.

Click ArgsContext.Sub Decorator



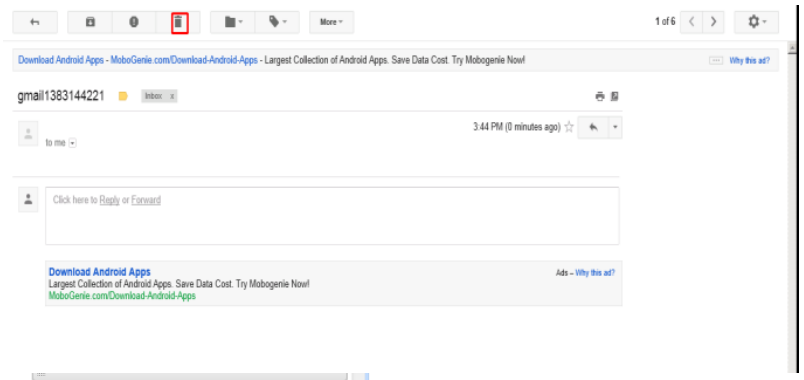
To identify an object, the name must be set as *ArgsContext.Sub.*

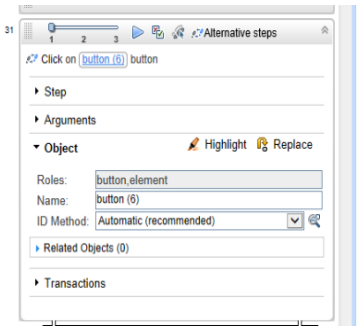
The ID Method is JavaScript:

```
evalXPath("//span[text()=\""+ArgsContext.Sub+"\"]");
```

The specific email is found.

Click Button (6)

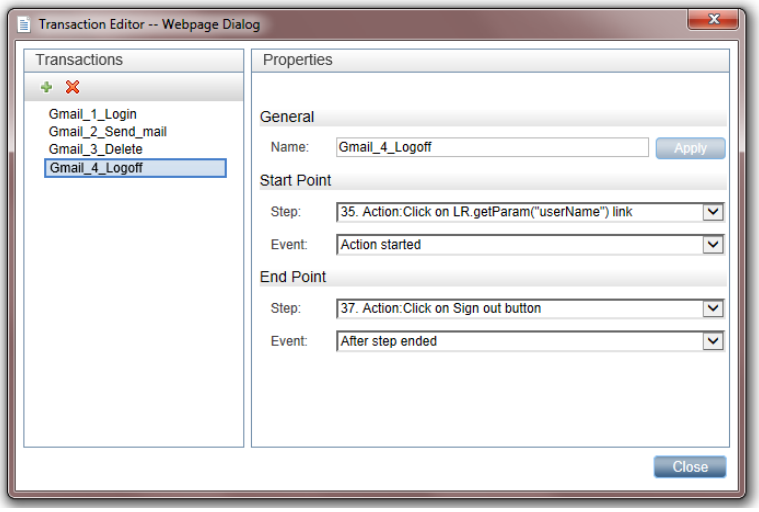




The **Delete** button (button(6)) is easy to recognize.

This step completes Gmail_3_Delete mail.

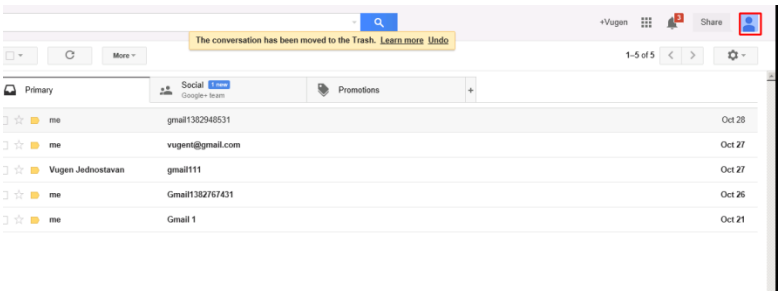
Transaction — Gmail_4_Logoff

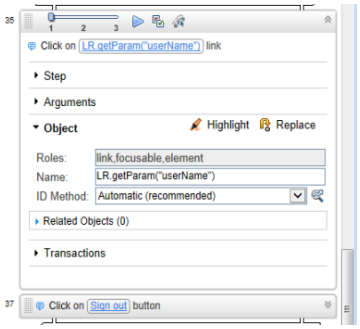


The name of this transaction is “Gmail_4_Logoff”.

This transaction contains all the steps necessary for logging off.

Click LR.getParam('userName')

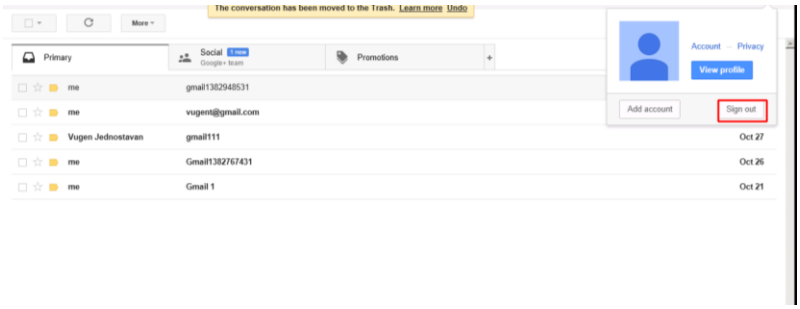


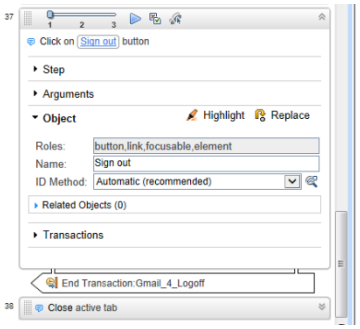


We used the same parameter as we did for the Login:

```
LR.getParam("userName")
```

Click Sign out





By clicking **Sign out**, we end this transaction and script.

All we need to do now is close the active tab.

Chapter 5: Cloud CRM Provider – Salesforce Transaction Flow

This case study uses Salesforce cloud service. The case study uses one flow/script with the following transactions:

1. **Login** – Open a login page, enter your credentials, and confirm that the first page loads.
2. **Create item (account)** – Open a new account with a unique account name.
3. **Delete item (account)** – Check for this account, select it and delete it.
4. **Logoff** – Log off from the web application.

The main purpose of running this script is to confirm that the application's processes run successfully. If successful, the transactions complete without any errors.

You can view a sample Salesforce script by downloading the following file: http://support.openview.hp.com/selfsolve/document/KM00658288/binary/BPM_Monitoring_Solutions-SalesForce_Trial.zip

General Flow

Salesforce is a cloud CRM (Customer Relationship Management).

A sequence of transactions creates the business process flow. A transaction is a unit that is measurable by availability and performance. Therefore, we group activities that perform a specific transaction.

To describe and validate these transactions, logically group the transactions into the following steps:

1. Login.
   a. Navigate to https://login.salesforce.com.
   b. Enter your credentials and confirm that the first page loads.
2. Create a new account.
   a. Click **New Account**.
   b. Enter a unique account name.
   c. Save the account.
3. Delete the account.
   a. Check if the account was created
   b. Select the account.
   c. Delete the account.
4. Logoff
   a. Log off from the web application

The following sections describe how to create these transactions.

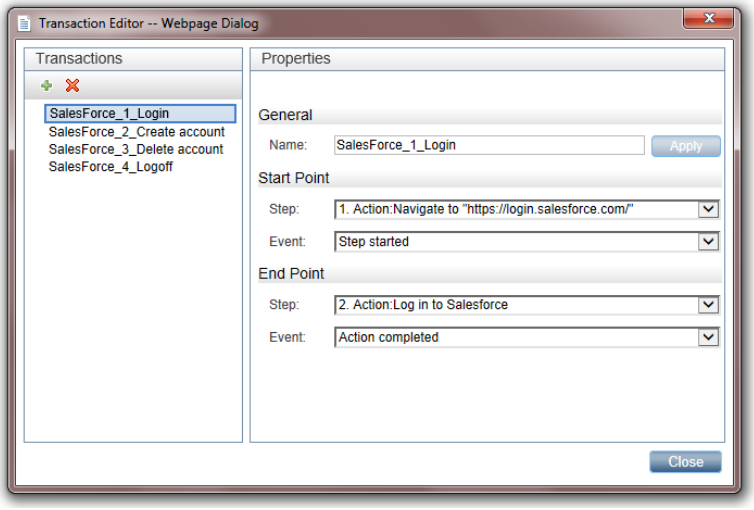
A good approach, before using this tool, is to know what we want to achieve. Create a blueprint as follows:

1. Run the Salesforce web application in Internet Explorer 9. Perform the steps of the flow and create annotations.
2. Determine the boundaries that form the transactions.
3. Select which values to assign to the parameters.
4. Select what values to protect and encrypt.

Record all the information you collect ,since it will come in handy later.

We recommend you start using the script. Refer to sections in this document for assistance.

Transaction SalesForce_1_Login

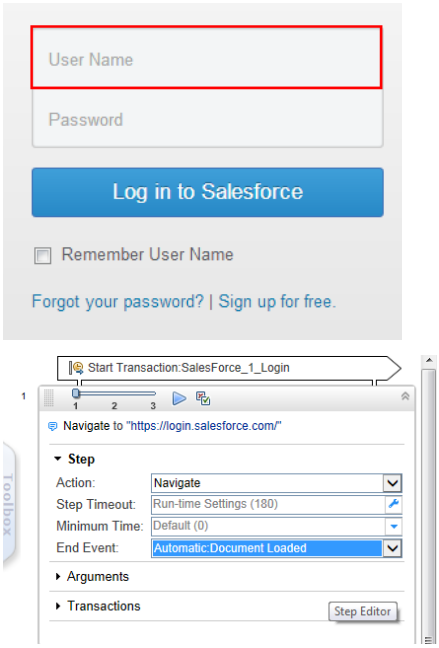


Define the name of the transaction. In this example, we call the transaction "SalesForce_1_Login". Naming conventions should provide a clear view of the application, the transaction order, and create an overview of the business processes covered by this transaction.

Define the steps and events for the starting and ending points of the transaction. The most convenient event for the start step is **Action started**. The most convenient event for the end step is **After step ended**.

This transaction contains all the steps necessary for logging into the Salesforce application.

Navigate to 'https://login.salesforce.com/'

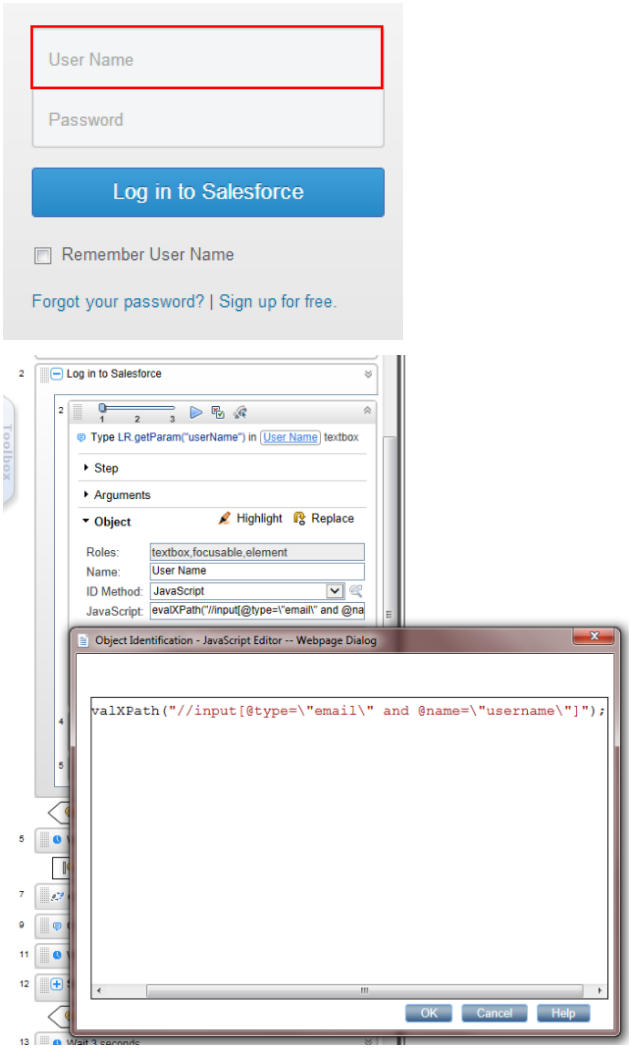


In the Navigation step, specify the URL.

While recording, VuGen automatically selects the appropriate End Event, in this case **Document downloaded**. The step ends when the process of loading a document completes. In most cases, you can freely select the default values.

If the pages load slowly, you can use a Wait step to make sure the page loads. For information on Wait steps, see *Meaning of Wait Step* on page 89.

Type LR.getParam('userName') in User Name Textbox



This step is actually a sub-step of Log in to SalesForce.

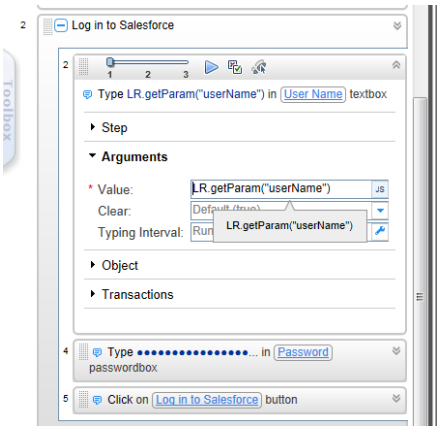
You can either select the **Automatic** or **JavaScript** ID method.

```
evalXPath("//input[@type=\"email\" and @name=\"username\"]");
```

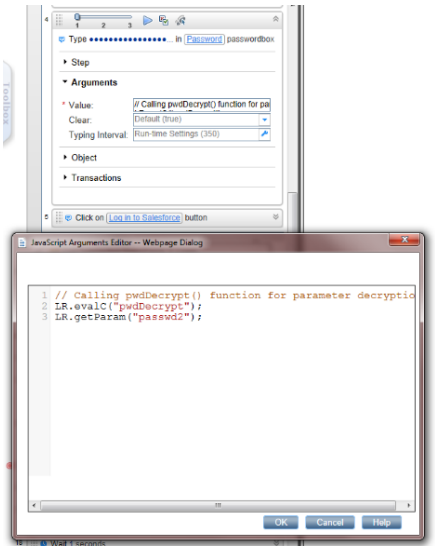
There are cases when you will change the ID Method, as we will see later.

Type your username in the **User Name** textbox.

Since we are using parameters, you can enter the *LR.getParam("userName");* command in the **Value** field and the result is the defined username.


2
Log in to Salesforce
2
1 2 3
Type LR.getParam("userName") in User Name textbox
Step
Arguments
* Value:
LR.getParam("userName")
JS
Clear:
Typing Interval:
LR.getParam("userName")
Object
Transactions
4
Type ••••••••••••••••... in Password passwordbox
5
Click on Log in to Salesforce button

Type ********** in Password Textbox



Type a password in the **Password** textbox. To display an encrypted password (so no one can see it), type the following:

```
// Calling pwdDecrypt() function for parameter decryption
LR.evalC("pwdDecrypt");
// This is a parameter with a decrypted value
LR.getParam("passwd2");
```

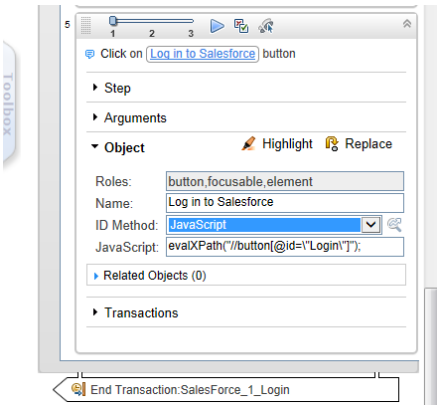
The pwdDecrypt() function is defined in C-function.c file:

```
void pwdDecrypt(){

lr_save_string(lr_decrypt(lr_eval_string("{passwd}")),"passwd2");
/*If you need to get some value from external parameter file*/
lr_output_message(lr_eval_string("passwd2"));
}
```

`passwd` is the parameter defined for external use (BSM – EUM) and is encrypted.

`passwd2` is the parameter seen only within this script instance.

Click Log in to Salesforce



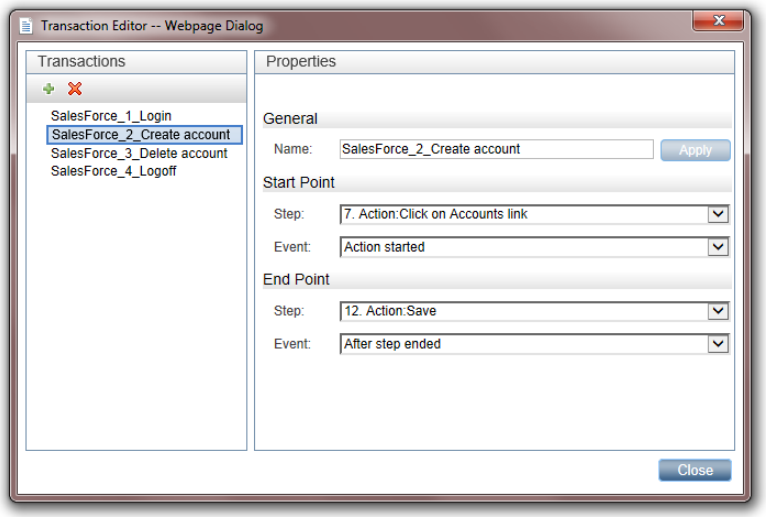
This is defined as the End step for transaction “SalesForce_1_Login”.

The JavaScript ID Method used is:

```
evalXPath("//button[@id=\"Login\"]");
```

The End Event Step is set to automatic, and in this case is **Step Synchronous network completed**. The step ends when all HTTP requests have completed excluding requests initiated by XMLHttpRequest.

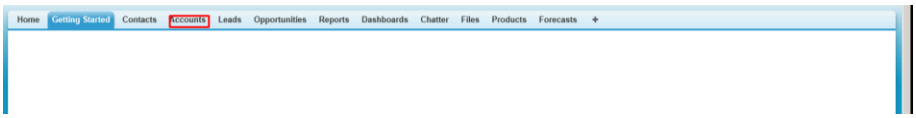
Transaction SalesForce_2_Create account

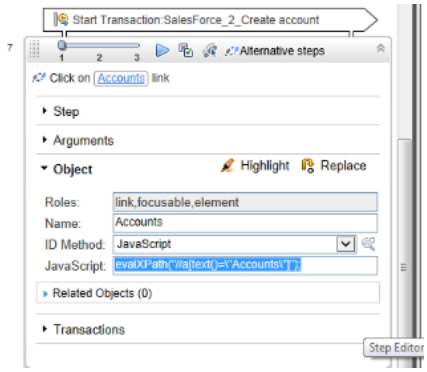


The name of this transaction is “SalesFoerce_2_Create account”.

This transaction contains all the steps necessary for creating an account.

Click Accounts



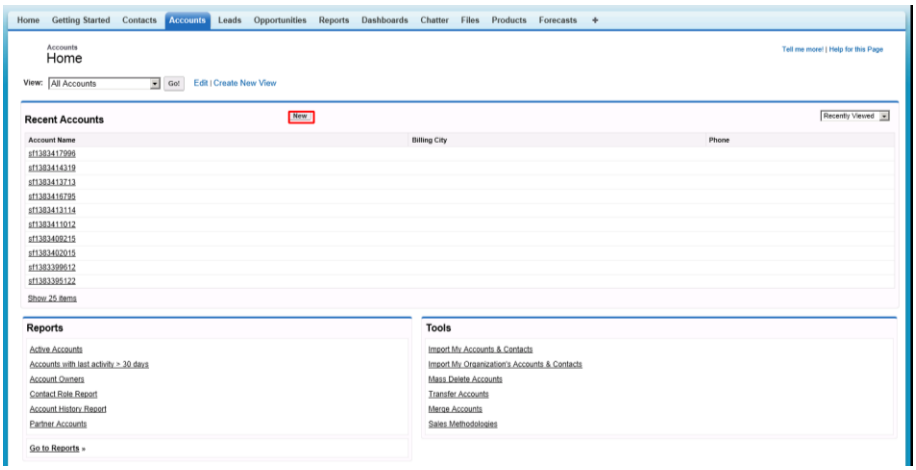


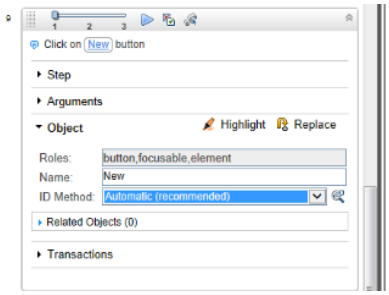
When you click **Account**, a form appears for listing accounts. By running the script multiple times, we can confirm that this element is recognized with the Automatic ID method.

You can also use the **JavaScript** ID method:

```
evalXPath("//a[text()=\"Accounts\"]");
```

Click New



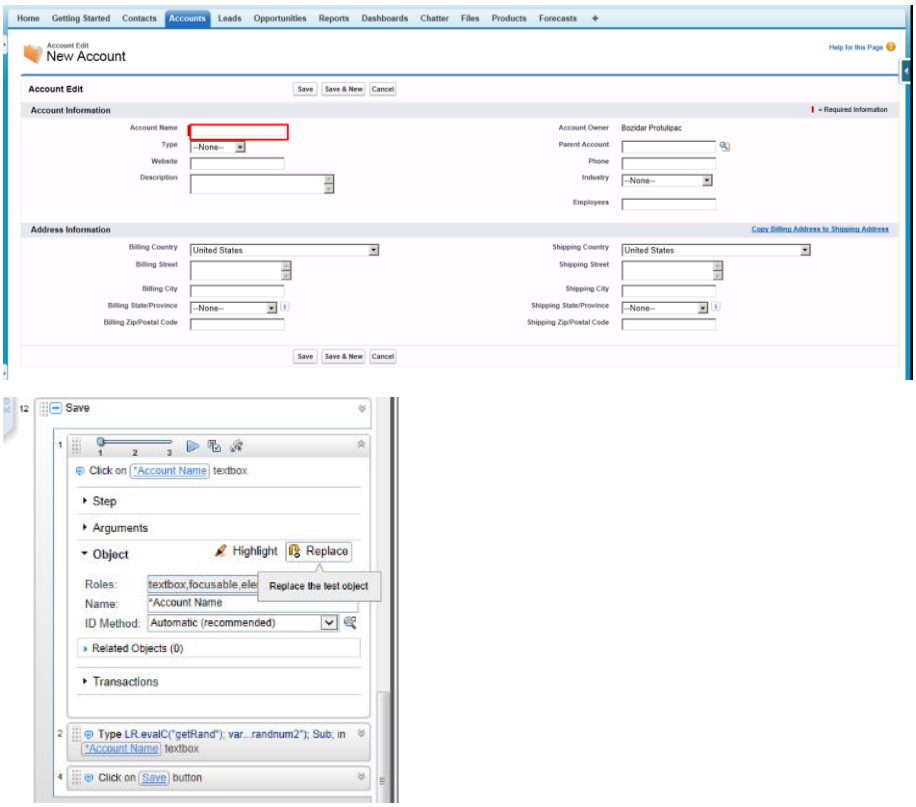


When you click **New**, a form appears for creating a new account. By running the script multiple times we can confirm that this element is recognized with the Automatic ID method.

You can also use the **JavaScript** ID method:

```
evalXPath("//input[@type=\"button\" and @name=\"new\" and @value=\"
New \"]");
```

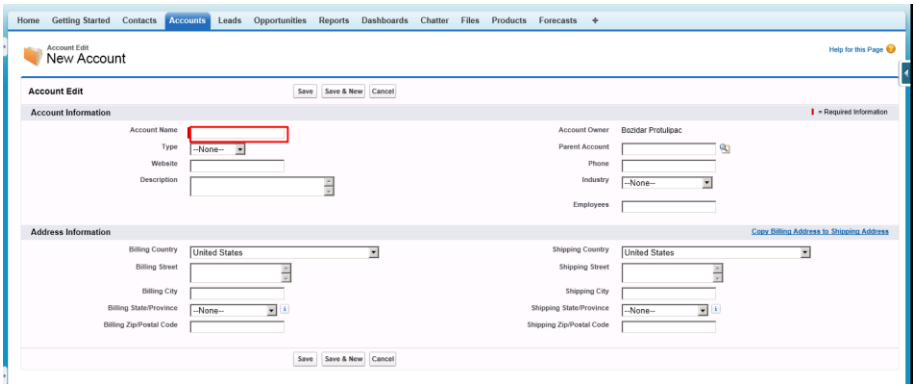
Click Account Name Textbox

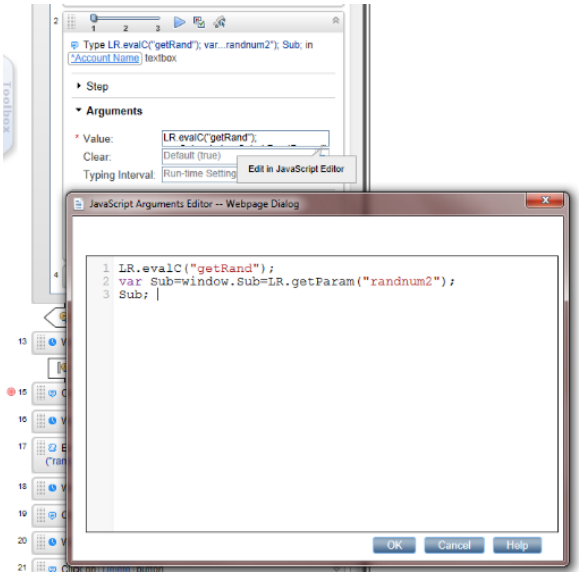


Click **Account Name** textbox.

To make this script robust, we use parameters. Therefore, if the username or password is changed at a later time, you do not need to change the script, only the parameter's value. The username is in the form of an email address.

Type LR.evalC('getRand'); var...randnum2'); Sub; in Account Name Textbox





By calling this function, we create a Subject value.

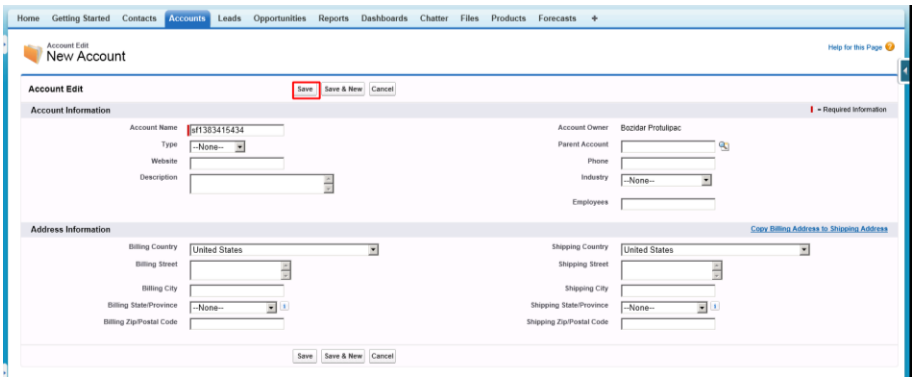
The getRand() function is defined in C-function.c file:

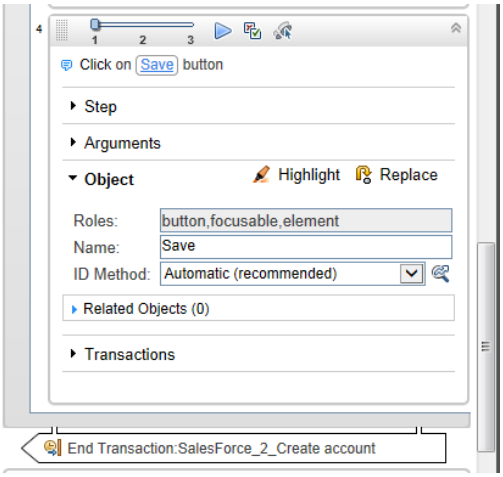
```
void getRand(){
typedef long time_t;
time_t t;
lr_param_sprintf("randnum2","sf%d",time(&t));
```

In this step, we acquire the value from the parameter:

```
LR.getParam("randnum2");
```

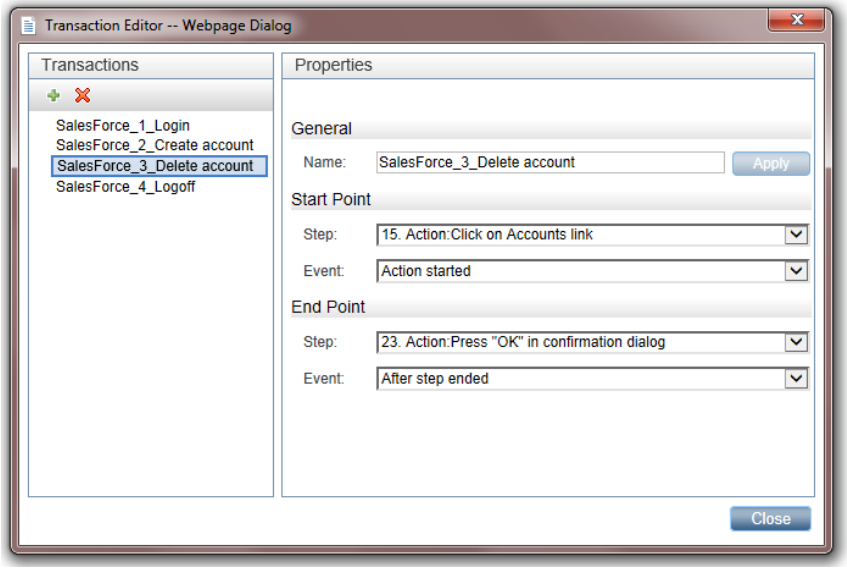
Click Save





Click **Save**. The **Save** button is easily recognized so there is no need for corrections.

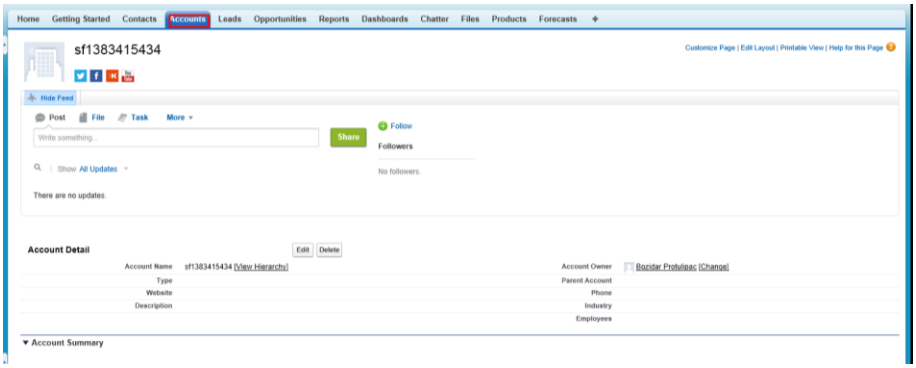
Transaction SalesForce_3_Delete account

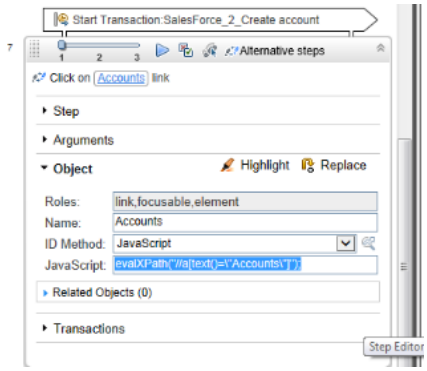


The name of this transaction is “SalesForce_3_Delete account”.

This transaction contains all the steps necessary for checking and deleting an account that was created in the current interaction.

Click Accounts





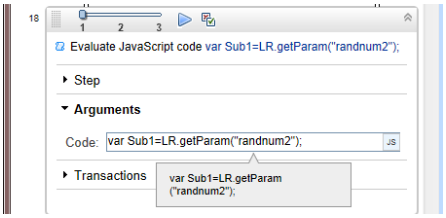
Click **Account** to open a form that lists the accounts.

By running the script multiple times, we can confirm that this element is recognized with the Automatic ID method.

You can also use the JavaScript ID Method:

```
evalXPath("//a[text()=\"Accounts\"]");
```

Evaluate JavaScript code var Sub1=LR.getParam('randnum2')

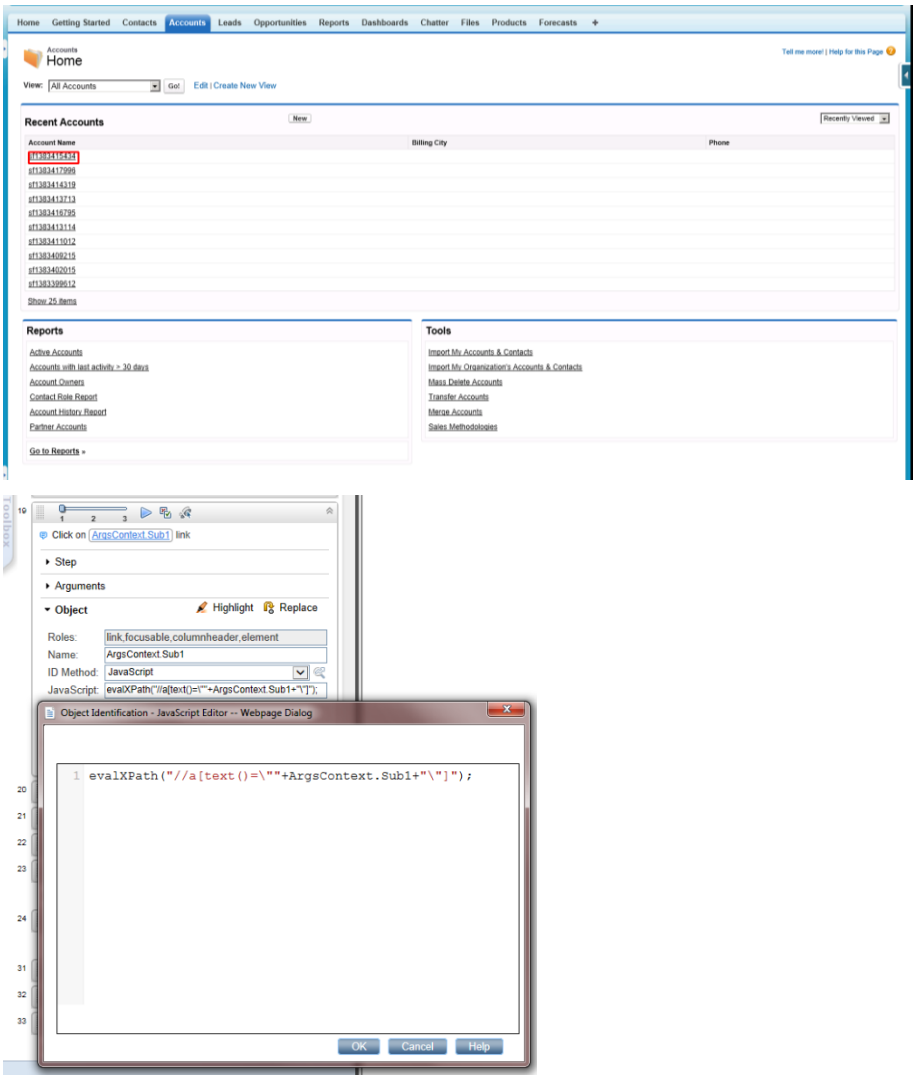


Create the *Sub1* variable and get its value from the parameter *randum2*:

```
var Sub1=LR.getParam("randnum2");
```

This variable is used to check (identify) and delete specific email sent within this interaction. The life span of this variable is within the page. When you change the page you must evaluate this Java script again if you need a value from the *randum2* parameter.

Click ArgsContext.Sub1



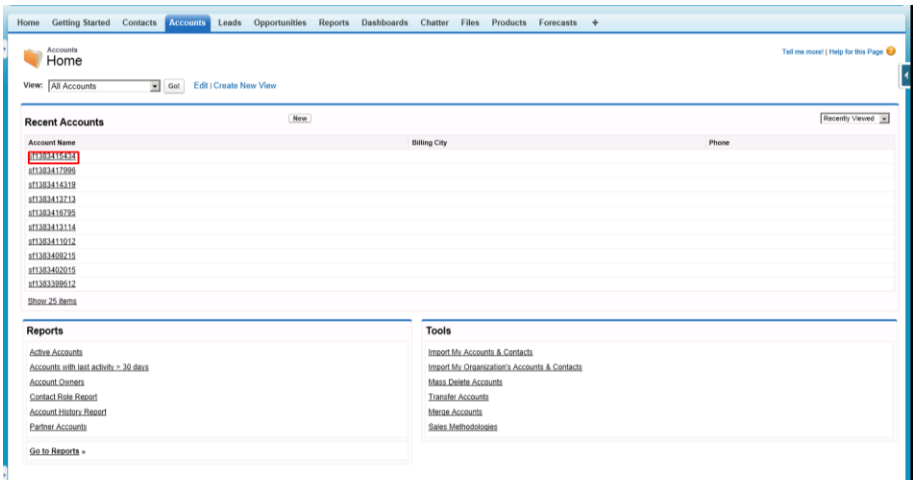
To identify an object, the name must be configured as *ArgsContext.Sub1* .

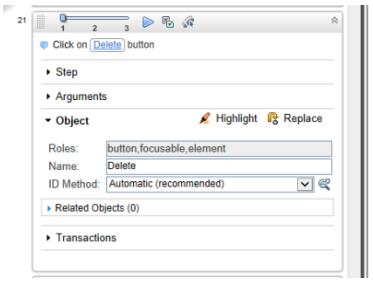
The ID Method is JavaScript:

```
evalXPath("//a[text()=\""+ArgsContext.Sub1+"\"]");
```

The specific account is found.

Click Delete

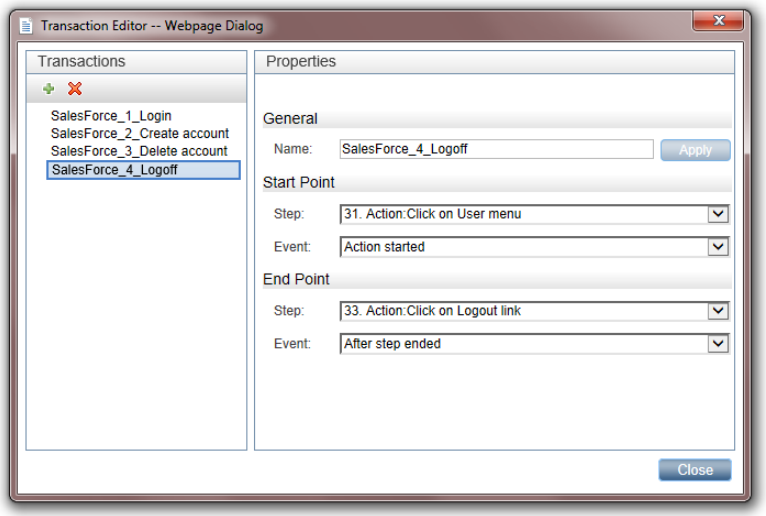




The **Delete** button is easy to recognize.

This step completes SalesForce_3_delete account.

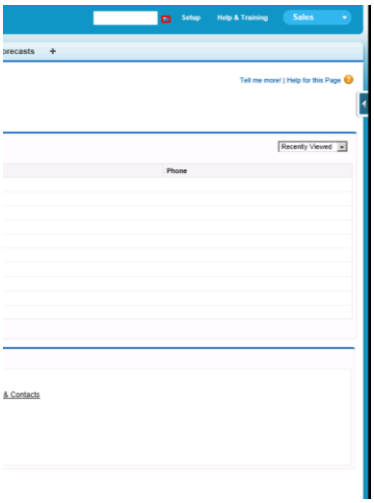
Transaction SalesForce_4_Logoff

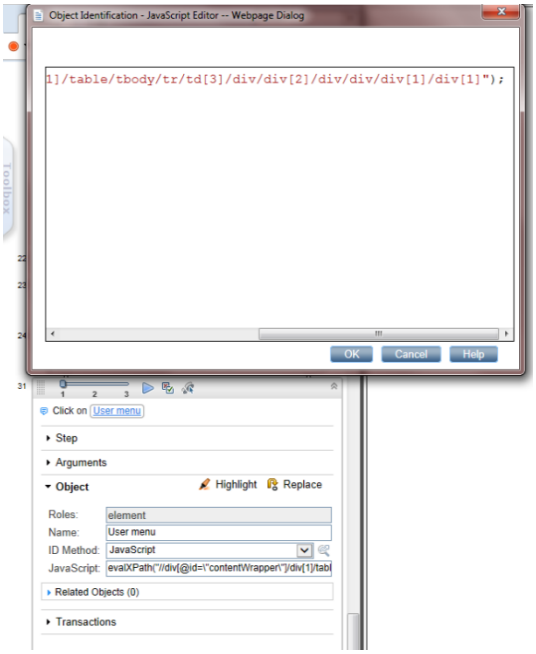


The name of this transaction is “SalesForce_4_Logoff”.

This transaction contains all the steps necessary for logging off.

Click User Menu





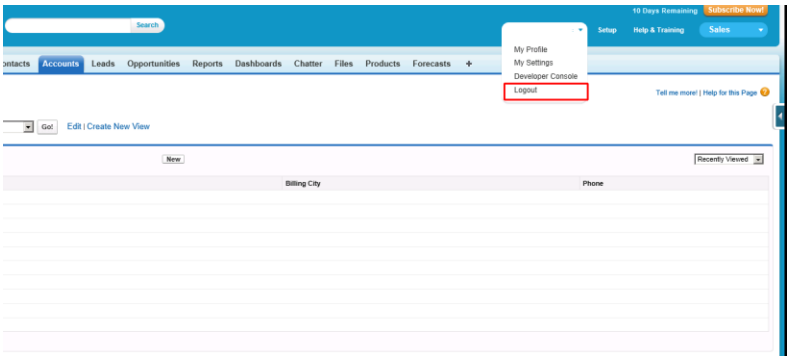
When you click the arrow next to the user name, a dropdown menu appears.

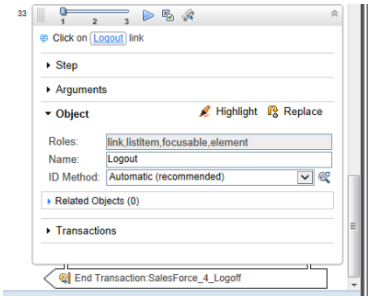
The ID Method is JavaScript:

```
evalXPath("//div[@id=\"contentWrapper\"]/div[1]/table/tbody/tr/td[3]/
div/div[2]/div/div/div[1]/div[1]");
```

You might want to use Firebug here.

Click Logout





Click **Logout** to end this transaction and script.

The optional JavaScript ID Method is: `evalXPath("//a[text()=\"Logout\"]");`

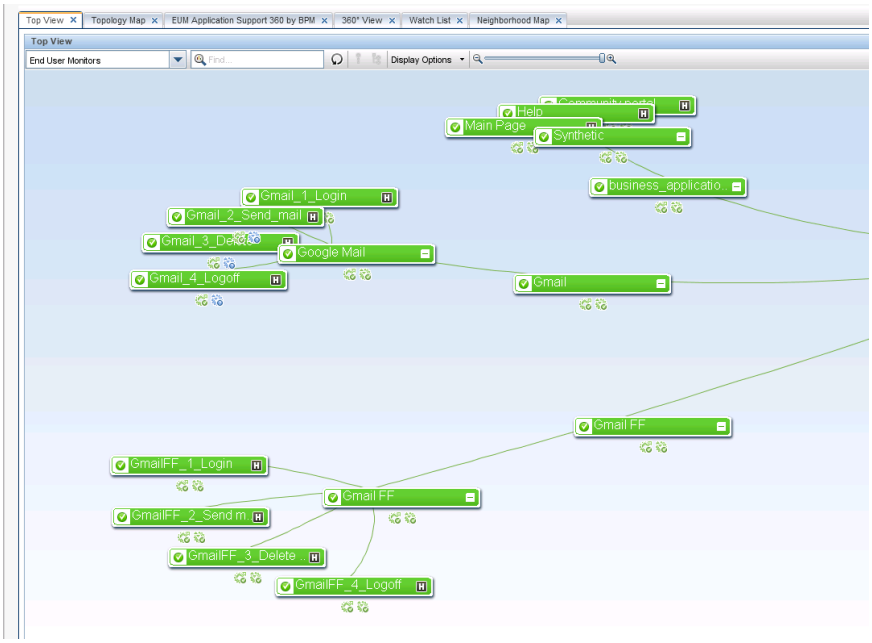
Chapter 6: Using Your Scripts to Monitor Your Cloud Service Provider

Isolating Performance Issues with HP BPM

BSM Service Health's Top View enables you to see the business performance of your system components at a glance. The CI bars in the component provide a visual representation of real-time IT performance metrics mapped onto business applications, based on the hierarchy structure defined for each view. The connecting lines between the bars define the relationships between the CIs.

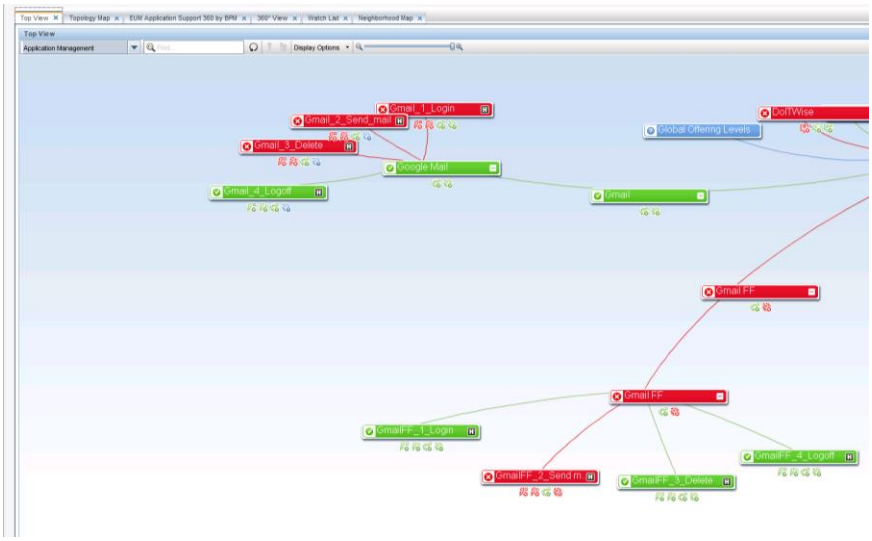
You can isolate performance issues with the scripts you create.

The following image is an *End User Monitors* view in Top View. It displays of a Gmail script with no performance issues.



Top View – End User MonitorsView

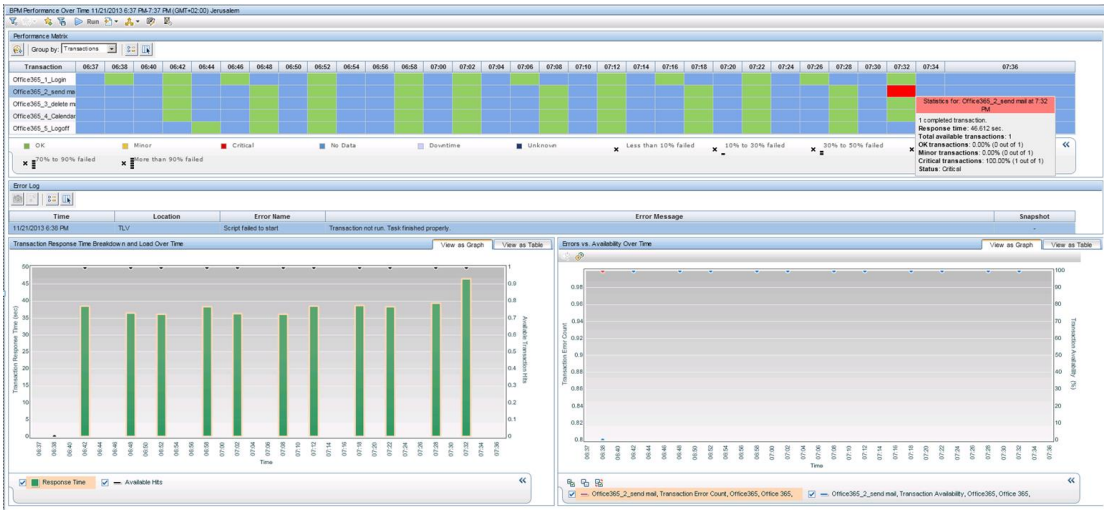
The following image is a Top View display of a Gmail script with “Send a mail” performance issue.



Top View – Performance Issues

From the Top View display you can drill down into the BPM Performance Over Time report to investigate a performance problem.

The *BPM Performance Over Time* report displays a matrix of the distribution of average transaction response times—organized by transaction or location—over a specified period of time. Additional components display different aspects of a selected transaction or of the transactions for a selected location. The report gives you the ability to investigate in more detail transaction hits for selected transactions or locations.

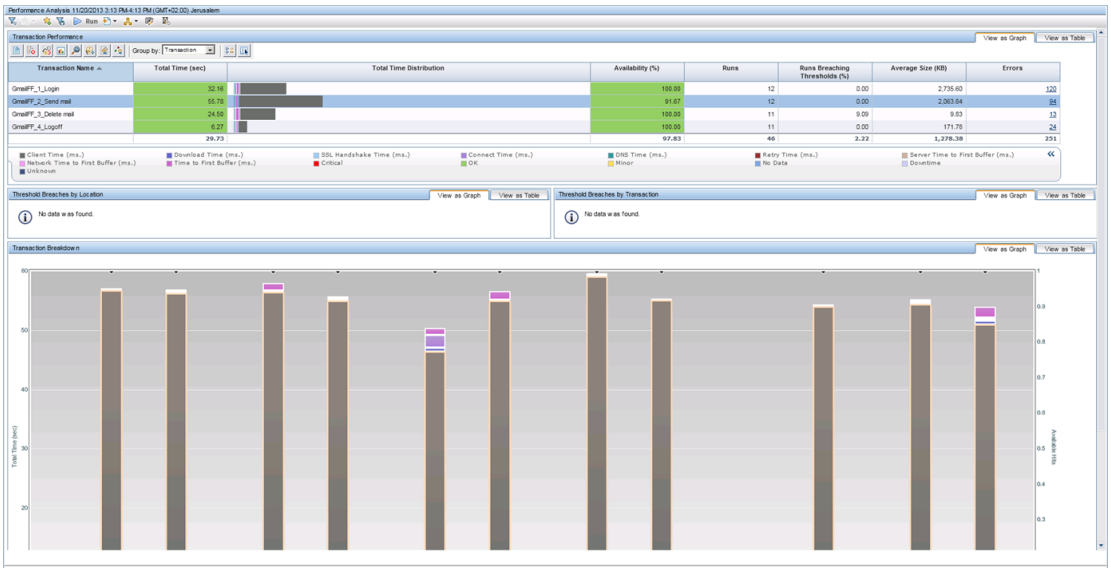


BPM Performance Over Time Report

From the BPM Performance Over Time report you can drill down to display the Breakdown Over Time component.

The *Breakdown Over Time* component helps you determine whether poor transaction response times are caused by network or server problems, or by client delays, and enables

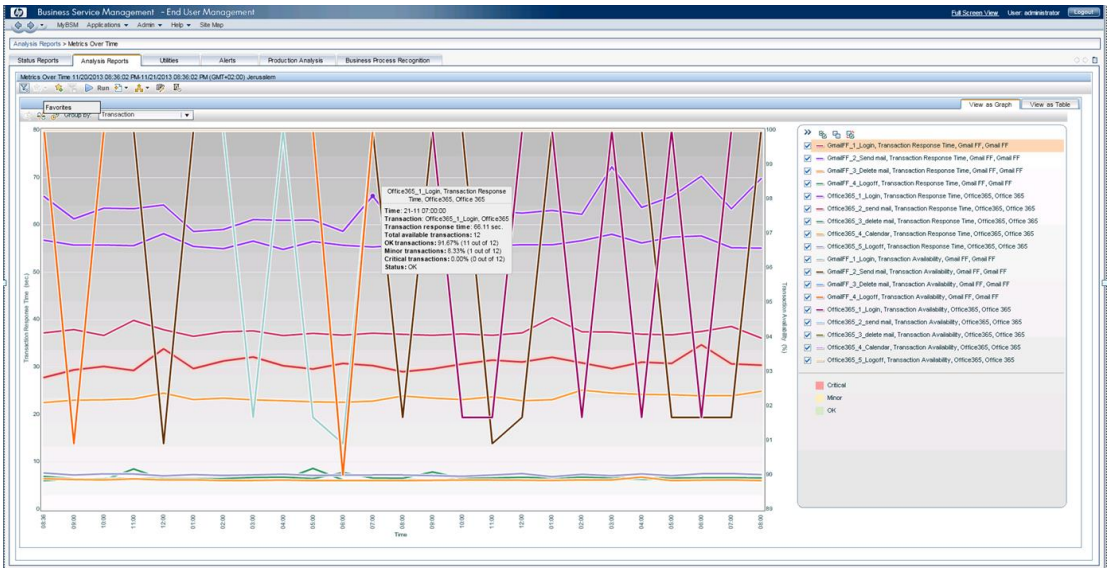
you to pinpoint exactly when the problems are occurring. Using the time range selector and active filters, you can highlight the exact time and source of a poorly performing transaction.



End User Management – Breakdown Over Time

You can also drill down to display the *Metrics Over Time* report.

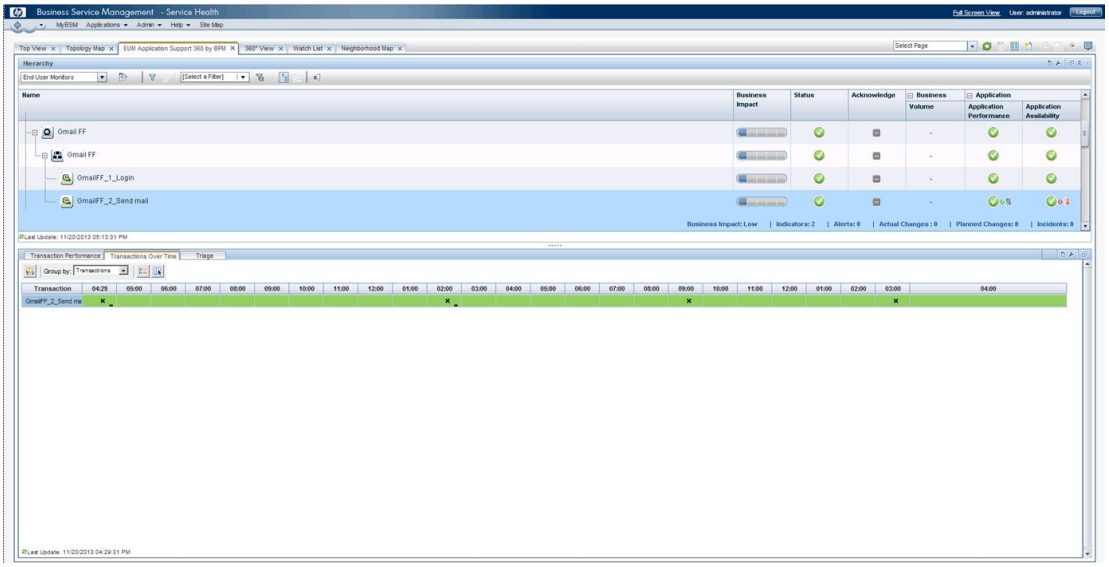
The Metrics Over Time report enables you to select one or more metrics for selected applications and to view their behavior over a period of time. By selecting multiple metrics, you can compare their behavior to discover possible correlations between them. You can also view data for up to four different time comparisons.



End User Management – Metrics Over Time

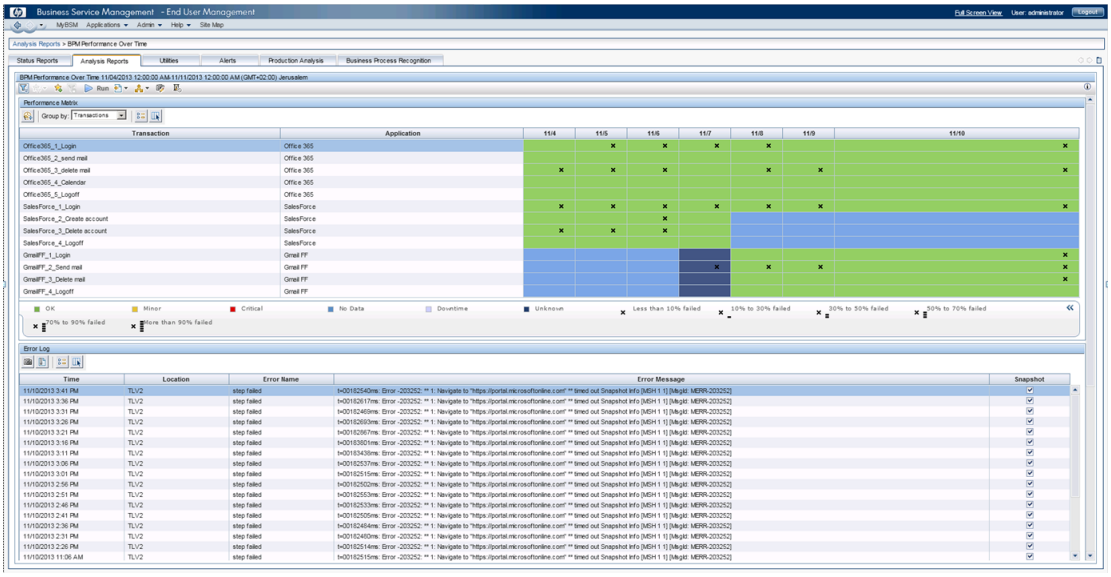
Isolating Availability Issues with HP BPM

The *End User Monitor* view in Service Health's *Hierarchy* component displays the hierarchy of the CIs in a view, and the real-time status of each of the KPIs assigned to the CIs. KPIs are grouped into domains; you can collapse a domain and display the worst status of all the KPIs in a domain. You can also filter the display to only show CIs with KPIs of a specific status, or specific CI types.



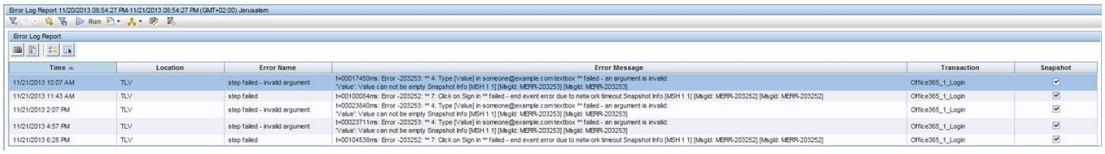
Service Health Hierarchy – End User Monitor View

From the Service Health Hierarchy – End User Monitor View you can drill down into the *BPM Performance Over Time – Performance Matrix*.



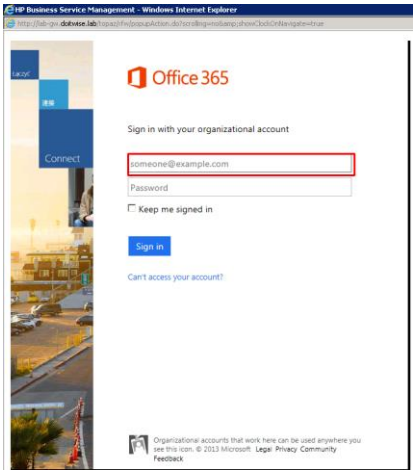
BPM Performance Over Time – Performance Matrix

The *Error Log Report* displays a list of errors that occurred in the selected BPM transactions, during a given time period.



Error Log Report

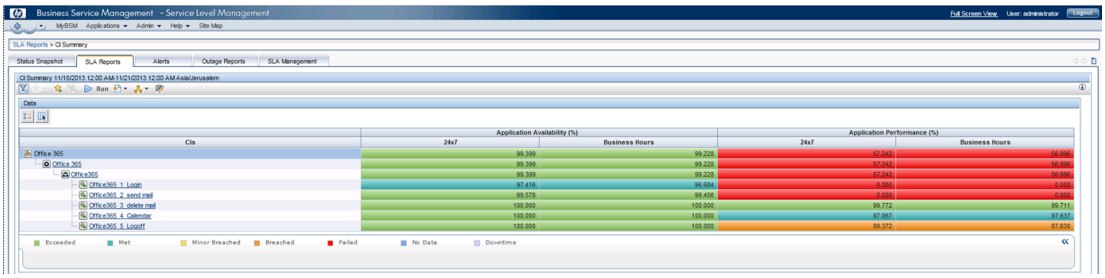
Creating a snapshot of an error as it appears when the error occurs saves time when opening support cases and provides additional information to aid in troubleshooting issues.



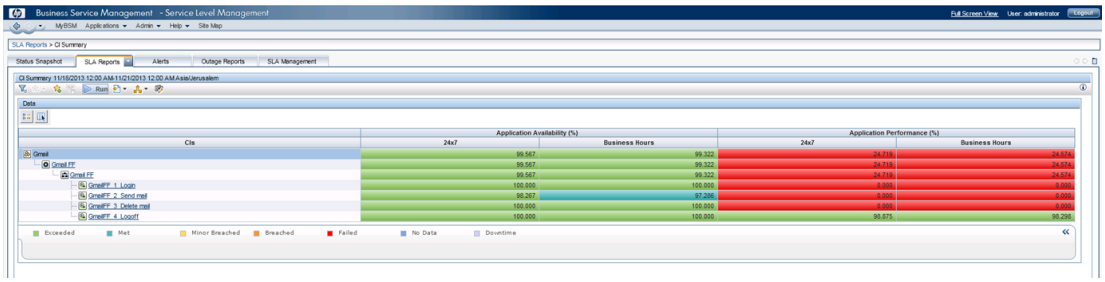
Defining, Tracking, and Reporting on SLAs

You can define, track, and report on service level agreements from a business perspective.

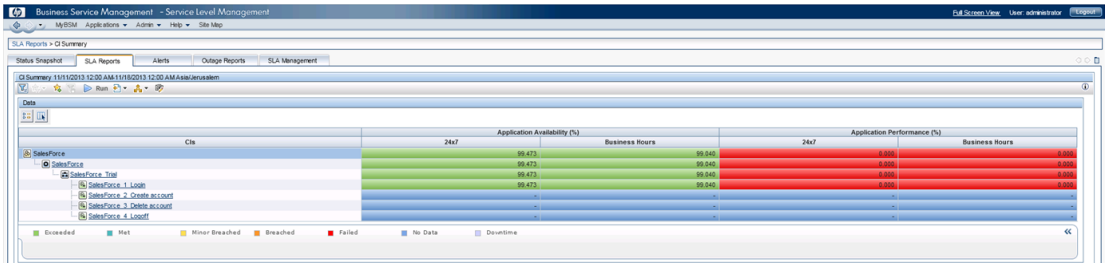
As shown in the images below, you can view the status of your cloud service provider's SLAs and OLAs.



SLA Report for Cloud Email Provider – Office 365



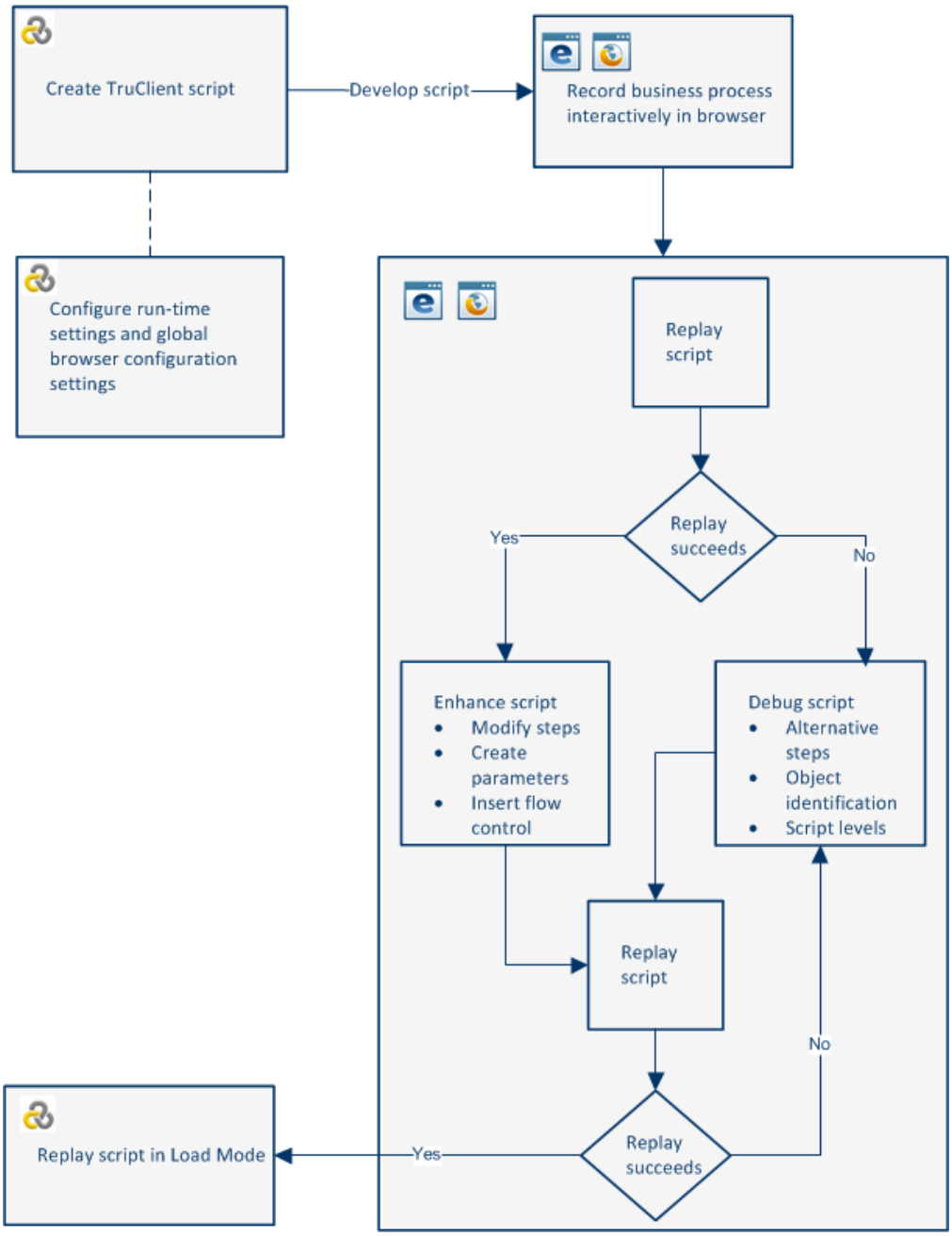
SLA Report for Cloud Email Provider – Gmail



SLA Report for Cloud CRM Provider – Salesforce

Appendix A: About TruClient for IE Protocol

The Ajax TruClient protocol interactively records scripts as you navigate through your business process. This enables TruClient to easily record and replay dynamic, complex web-based applications and create user friendly scripts. Scripts are created in real-time and steps can be seen in the TruClient sidebar as they are performed.



Ajax TruClient Workflow

The Ajax TruClient for IE protocol is designed to work with applications running in IE9 in standard mode only. When you need a deeper insight, use Firebug Lite (see *Use Firebug Lite* on page 86).

Based on LoadRunner's innovative Ajax TruClient technology, Mobile TruClient enables you to test web applications designed for mobile devices. With this protocol you can:

- Simulate various mobile browsers.
- Develop scripts that are recorded on the user level making them clear and easily maintained.

Mobile TruClient Workflow

Appendix B: Time to Value

There are a few concepts that can speed things up when creating Office 365 scripts.

Use Firebug Lite

Firebug Lite is a third-party utility that provides many valuable development tools. You can edit, debug, and monitor CSS, HTML, and JavaScript live in any web page. You can access this utility by selecting **Advanced > FireBug Lite** or by pressing **F12** while in TruClient's IE9 browser.

If other default ID methods fail during the replay, you can use Inspect mode in Firebug Lite to select the evalXPath() that will choose the correct object based on the HTML. This is useful when it comes to tricky objects.

Standard Steps

When you look at the script, you will notice that some steps repeat often. We can group these steps as follows:

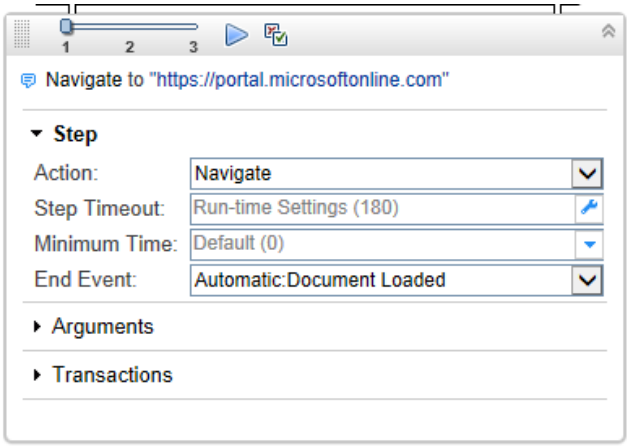
- Navigate (navigate to a URL)
- Click Object (Link, Textbox, Button, or Decorator)
- Type in Text Object
- Evaluate JavaScript

Navigate

Keep in mind when navigating to a defined URL, the End Event value defines when this step finishes. An End Event can be one of the following:

- **Action Completed** – The step ends when its action is completed. An example of an action is clicking a button.
- **DOM load** – The step ends when the process of loading a document completes.
- **DOM content loaded** – The step ends when the HTML parsing of the document completes.
- **Step network completed** – The step ends when all HTTP requests have completed including requests initiated by XMLHttpRequest.
- **Step synchronous network completed** – The step ends when all HTTP requests have completed, excluding requests initiated by XMLHttpRequest.
- **Dialog opened** – The step ends when a dialog box is opened.

We recommend that you set the End Event to **Automatic**, as seen in this script.



Automatic End Event Setting for Navigate Step

Click Object (Link, Textbox, Button, or Decorator)

Two things are important for the Click on Object step:

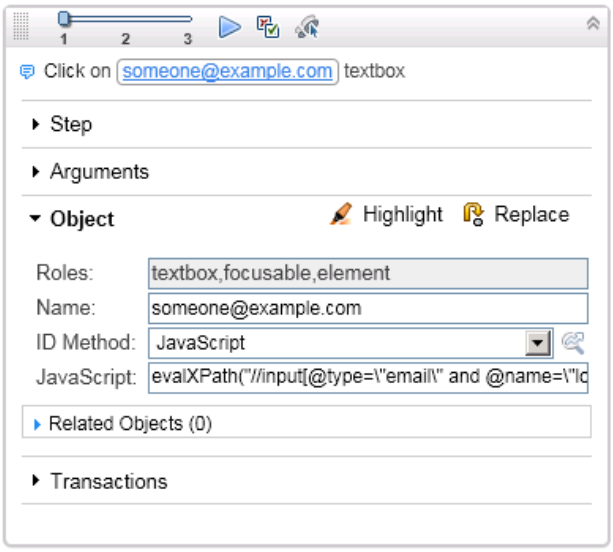
- Arguments
- Object settings

The X and Y coordinates are defined relative to the object in the Arguments settings. If the coordinates are left empty, the click is executed in the middle of the object.

If the object is not recognized with the ID Method set to automatic, you will need to use the JavaScript ID Method. It relies on finding the required element or string in the HTML code. When you select the JavaScript ID Method, you can get:

- Straight forward syntax:
  `evalXPath("//input[@type=\"email\" and @name=\"login\"]");`
- Suspicious syntax:
  `evalXPath("/html/body/div[5]/div/div[2]/div[3]/div/div[1]/div[6]/div/div/div[3]/div[2]/div[1]/div[7]/div/div/input");`

When confronted with suspicious syntax, use Firebug Lite as described in *Use Firebug Lite* on page 86.

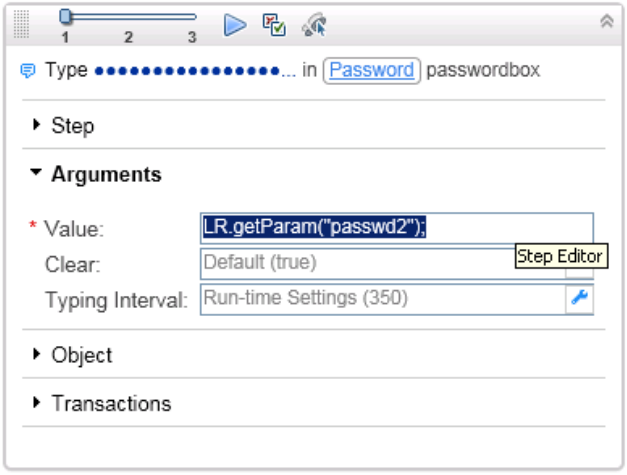


Straight Forward JavaScript Syntax for ID Method

Type in Text Object

Argument values can be plain text or a parameter. Examples are:

- **Plain text** – "this is plain text typed in a text box"
- **Getting a value from a parameter** – LR.getParam("passwd2");



Getting a Value from a Parameter – LR.getParam("passwd2")

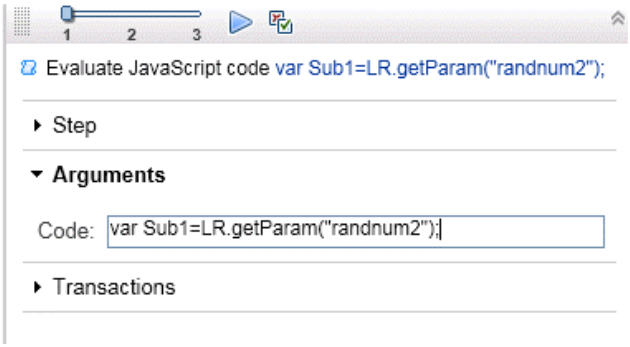
Evaluate JavaScript

By evaluating JavaScript, you can call a function, for example:

```
LR.evalC("getRand");
```

Or define a variable that will be used later:

```
var Sub1=LR.getParam("randnum2");
```

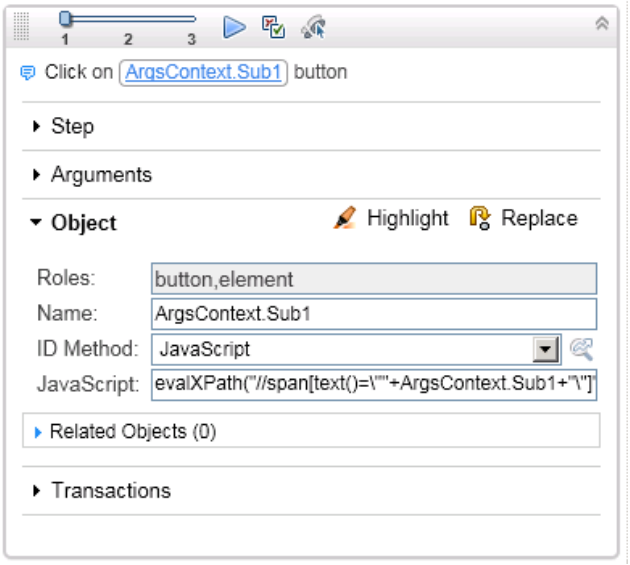


JavaScript Evaluation Step

Variables and ID Methods – How to Combine

To use a defined variable to find a specific and unique object:

1. Record the step by clicking that object.
2. To make a script more robust, make those changes:
   - Change Object name to: `ArgsContext.Sub1`
   - Change JavaScript ID Method to:
     `evalXPath("//span[text()=\""+ArgsContext.Sub1+"\"]");`



Meaning of Wait Step

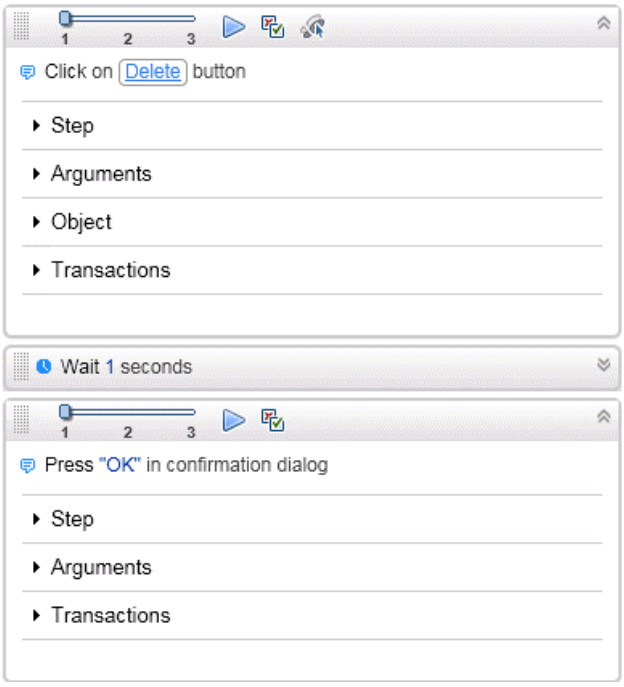
A Wait step is useful when you want to wait for a very heavy page to load and you do not want to timeout while waiting. Just put a Wait step after this load step and enter the number of seconds to wait.

Another example of when to use a Wait step is when a dialog box is in the script. This dialog box can be any of the following:

- Alert
- Authentication

- Confirmation
- Prompt
- Prompt Password

Put a Wait step between the step that caused the dialog box to open and the dialog box itself. In this way, you give the dialog box step time to complete.



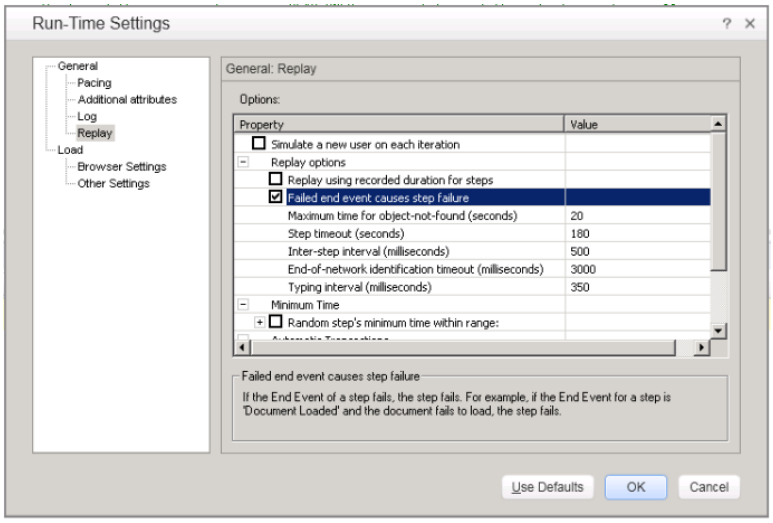
Wait Step before Dialog Box Step

Appendix C: Script Validation Concept

Validation of the script within the VuGen domain, is performed by replaying the script.

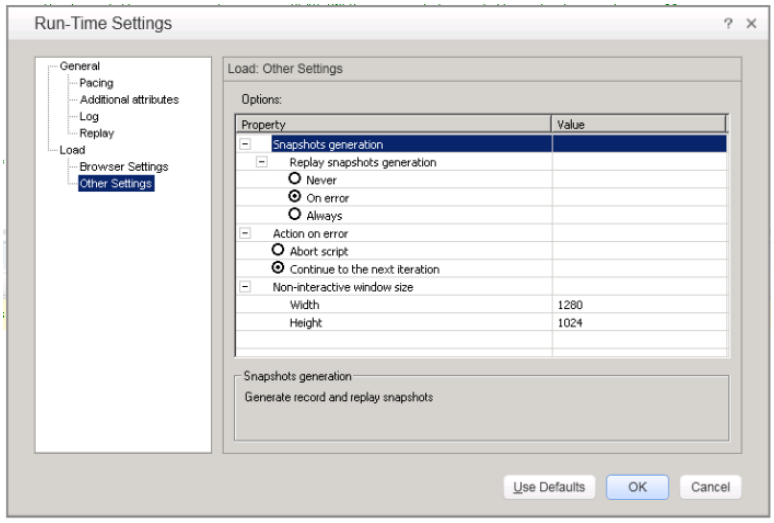
You can replay the script in the TruClient for IE browser or the main VuGen window. What is important is to define the run-time settings:

- **Pacing** – Define the number of iterations to be used for testing. Test the script multiple times to see how it behaves.
- **Log** – When troubleshooting, use the extended log to locate problems quickly.
- **Replay** – Select the **Failed end event causes step failure** option



Failed End Event Causes Step Failures Option

- **Browser settings** – If a proxy is used, configure it for testing purposes.
- **Other settings** – The most valuable setting is to generate a snapshot of the error.



Generate a Snapshot of the Error