Salesforce Functions

Salesforce functions provide login and session information and lookups for Salesforce instances.

On This Page

- GetSalesforceTimestamp
- LoginToSalesforceAndGetTimeStamp
- SalesforceLogin
- SetSalesforceSession
- SfCacheLookup
- SfLookup
- SfLookupAll
- SfLookupAllToFile

Search in This Topic

Related Topics

- Cloud Studio
- Functions
- Salesforce
- Scripts

Last updated: May 24, 2019

GetSalesforceTimestamp

Declaration

```
string GetSalesforceTimestamp(string url, string sessionId[, string
timeZoneId])
```

Syntax

```
GetSalesforceTimestamp(<url>, <sessionId>[, <timeZoneId>])
```

Required Parameters

- **`url`:** The URL to use when calling Salesforce, from calling `SalesforceLogin()`
- **`sessionId`:** The Salesforce session ID, from calling `SalesforceLogin()`

Optional Parameters

- **`timeZoneId`:** A time zone ID to use, as described under Time Zone Codes. The default is the time zone of the Salesforce endpoint identified by the Salesforce session ID.

Description

Retrieves the current system time from Salesforce.

The timestamp is returned in the format `yyyy-mm-dd HH:MM:SS`, using your Salesforce endpoint's time zone setting by default. You must have logged into Salesforce using the `SalesforceLogin()` function before using this function. The function returns a null value if the call fails. Use the `GetLastError()`function to retrieve the error message in that case.

The optional third argument can be used to set the time zone to use. The time zone of your Salesforce endpoint is used by default if a time zone is not provided. The time zone argument must be a time zone recognized by Java's TimeZone class.

In a common scenario, the `SalesforceLogin()` function is called first. The URL and session ID are then available in these global variables:

- `$Salesforce.ServerUrl`
- `$Salesforce.SessionId`

As an alternative to this function, see also the `LoginToSalesforceAndGetTimestamp()` function.

Examples

```
// Start by logging into Salesforce
if(!SalesforceLogin("..."), RaiseError(GetLastError()));
timestamp = GetSalesforceTimestamp($Salesforce.ServerUrl,
    $Salesforce.SessionId);
// Check for any errors before proceeding further
if(IsNull(timestamp), RaiseError(GetLastError()));

// Retrieving the timestamp in UTC:
timestamp = GetSalesforceTimestamp($Salesforce.ServerUrl,
    $Salesforce.SessionId, "UTC");

// Retrieving the timestamp in Pacific Standard Time:
timestamp = GetSalesforceTimestamp($Salesforce.ServerUrl,
    $Salesforce.SessionId, "PST");
```

[Back to Top]

LoginToSalesforceAndGetTimeStamp

Declaration

```
string LoginToSalesforceAndGetTimestamp(string salesforceOrg[, string
timeZoneId])
```

Syntax

```
LoginToSalesforceAndGetTimestamp(<salesforceOrg>[, <timeZoneId>])
```

Required Parameters

- **`salesforceOrg:`** A string reference path to a Salesforce connection in the current project

Optional Parameters

- **`timeZoneId:`** A time zone ID to use, as described under Time Zone Codes. The default is the time zone of the Salesforce endpoint.

Description

Logs in to Salesforce using a Salesforce endpoint and retrieves the current system time from Salesforce.

The Salesforce endpoint used in this function call must be defined as a Salesforce connection in the current project. For more information, see the instructions on inserting endpoints under the Endpoints section in Jitterbit Script.

The login call is made using the credentials in the specified Salesforce endpoint. The timestamp is returned in the format `yyyy-mm-dd HH:MM:SS`, using your Salesforce endpoint's time zone setting by default. The function returns a null value if the call fails. Use the `GetLastError()` function to retrieve the error message in that case.

The optional argument can be used to set the time zone to use. The time zone of your Salesforce endpoint is used by default if a time zone is not provided. The time zone argument must be a time zone recognized by Java's TimeZone class.

Once this function has been called, the Salesforce URL and session ID are available in these global variables:

- `$Salesforce.ServerUrl`
- `$Salesforce.SessionId`

As an alternative to this function, see also the `GetSalesforceTimestamp()` function.

Examples

```
// Logs in to Salesforce and retrieves the timestamp
timestamp = LoginToSalesforceAndGetTimestamp("<TAG>endpoint:salesforce
/Salesforce</TAG>");

// Check for any errors before proceeding further
if(IsNull(timestamp), RaiseError(GetLastError()));

// Retrieving the timestamp in UTC:
timestamp = LoginToSalesforceAndGetTimestamp("<TAG>endpoint:salesforce
/Salesforce</TAG>", "UTC");

// Retrieving the timestamp in Pacific Standard Time:
timestamp = LoginToSalesforceAndGetTimestamp("<TAG>endpoint:salesforce
/Salesforce</TAG>", "PST");
```

[Back to Top]

SalesforceLogin

Declaration

```
bool SalesforceLogin(string salesforceOrg)
```

Syntax

```
SalesforceLogin(<salesforceOrg>)
```

Required Parameters

- **`salesforceOrg`:** A string reference path to a Salesforce connection in the current project

Description

Logs into Salesforce, using the specified Salesforce endpoint.

The Salesforce endpoint used in this function call must be defined as a Salesforce connection in the current project. For more information, see the instructions on inserting endpoints under the Endpoints section in Jitterbit Script.

After a successful login, these global variables will have been set and can be used in subsequent scripts or mappings:

- `$Salesforce.SessionID`: The Salesforce session ID
- `$Salesforce.ServerURL`: The URL to use in subsequent calls to Salesforce in the same session
- `$Salesforce.UserID`: The ID of the Salesforce user

The function returns true if the login was successful and false if the login failed. Use the `GetLastError()` function to retrieve the error message in that case.

Examples

```
// Logs in to Salesforce
result = SalesforceLogin("<TAG>endpoint:salesforce/Salesforce</TAG>");

// Check for any errors before proceeding further
if(!result, RaiseError(GetLastError()));
```

[Back to Top]

SetSalesforceSession

Declaration

```
void SetSalesforceSession(string salesforceOrg, string sessionId, string
serverURL)
```

Syntax

```
SetSalesforceSession(<salesforceOrg>, <sessionId>, <serverURL>)
```

Required Parameters

- **`salesforceOrg`:** A string reference path to a Salesforce connection in the current project
- **`sessionID`:** A Salesforce session ID
- **`serverURL`:** A Salesforce server URL

Description

Sets Salesforce session information for the specified Salesforce endpoint. Use this function if you have an existing Salesforce session ID and server URL. Calling this function will disable the automatic Salesforce login and instead use the provided session information.

The Salesforce endpoint used in this function call must be defined as a Salesforce connection in the current project. For more information, see the instructions on inserting endpoints under the Endpoints section in Jitterbit Script.

After a successful function call, these global variables will have been set and can be used in subsequent scripts or mappings:

- `$Salesforce.SessionID`: The Salesforce session ID.
- `$Salesforce.ServerURL`: The URL to use in subsequent calls to Salesforce in the same session.

This function does not validate the input; it only fails if either the session ID or the server URL are empty or the referenced Salesforce endpoint does not exist. If either the session ID or server URL is invalid, subsequent Salesforce operations will fail.

Use the `Eval()` function to catch errors, calling the `GetLastError()` function to retrieve the error message.

> **WARNING:** It is a known issue that the `SetSalesforceSession` function does not work correctly and should not be used. If used, the script may or may not generate an error. Even if the script does not generate an error, the function will not work correctly. As the function doesn't work, the built-in session handling of the Salesforce connector will be used.

Examples

```
// Sets the Salesforce session information
sfOrg = "<TAG>endpoint:salesforce/Salesforce</TAG>";
sfSessionID = "00D5...SE";
sfURL = "https://example.my.salesforce.com/services/Soap/u/39.0/...";
// Logs in to Salesforce
Eval(SetSalesforceSession(sfOrg, sfSessionID, sfURL),
    RaiseError("Failed to set Salesforce session: " + GetLastError()));
```

[Back to Top]

SfCacheLookup

Declaration

```
string SfCacheLookup(string salesforceOrg, string soql)
```

Syntax

```
SfCacheLookup(<salesforceOrg>, <soql>)
```

Required Parameters

- **`salesforceOrg:`** A string reference path to a Salesforce connection in the current project
- **`soql:`** A query written in Salesforce Object Query Language (SOQL)

Description

Logs into Salesforce (if necessary) and retrieves the result of the query from Salesforce. Only the value from the first field of the first record is returned.

The Salesforce endpoint used in this function call must be defined as a Salesforce connection in the current project. For more information, see the instructions on inserting endpoints under the Endpoints section in Jitterbit Script.

Values are cached so that subsequent calls with the same exact parameters (Salesforce endpoint and SOQL) do not trigger a call to Salesforce. Salesforce is called only the first time.

The function returns `null` if the login fails, the query returns no records, or the API fails. Use the `GetLastError()` function to retrieve the error message in that case.

Examples

```
// Logs in to Salesforce and retrieves the first result of a query
// If the query is cached, it uses the cached value
myId = SFCacheLookup("<TAG>endpoint:salesforce/Salesforce</TAG>",
    "SELECT Id FROM Account WHERE Name='My Account'");

// Checks for any errors before proceeding
if(IsNull(myId), RaiseError(GetLastError()));
```

[Back to Top]

SfLookup

Declaration

```
string SfLookup(string salesforceOrg, string soql)
```

Syntax

```
SfLookup(<salesforceOrg>, <soql>)
```

Required Parameters

- **`salesforceOrg:`** A string reference path to a Salesforce connection in the current project
- **`soql:`** A query written in Salesforce Object Query Language (SOQL)

Description

Logs into Salesforce (if necessary) and retrieves the result of the query from Salesforce. Only the value from the first field of the first record is returned.

The Salesforce endpoint used in this function call must be defined as a Salesforce connection in the current project. For more information, see the instructions on inserting endpoints under the Endpoints section in Jitterbit Script.

The function returns `null` if the login fails, the query returns no records, or the API fails. Use the `GetLastError()` function to retrieve the error message in that case.

See also the `SFLookupAll()` and `SFLookupAllToFile()` functions.

Examples

```
// Logs in to Salesforce and retrieves the first result of a query
myId = SFLookup("<TAG>endpoint:salesforce/Salesforce</TAG>",
    "SELECT Id FROM Account WHERE Name='My Account'");

// Checks for any errors before proceeding
if(IsNull(myId), RaiseError(GetLastError()));
```

[Back to Top]

SfLookupAll

Declaration

```
array_2D SfLookupAll(string salesforceOrg, string soql)
```

Syntax

```
SfLookupAll(<salesforceOrg>, <soql>)
```

Required Parameters

- **`salesforceOrg:`** A string reference path to a Salesforce connection in the current project
- **`soql:`** A query written in Salesforce Object Query Language (SOQL)

Description

Logs into Salesforce (if necessary) and retrieves the result of the query from Salesforce. The returned array is two-dimenional; an array of records, with each record an array of named fields.

The Salesforce endpoint used in this function call must be defined as a Salesforce connection in the current project. For more information, see the instructions on inserting endpoints under the Endpoints section in Jitterbit Script.

The function returns `null` if the login fails, the query returns no records, or the API fails. Use the `GetLastError()` function to retrieve the error message in that case.

There are limitations if a relationship query is used:

- Only immediate relationship can be retrieved. The query cannot include a grandchild relationship.
- For each query record, each child cannot have multiple records.
- In the query statement, the fields under the same child should be grouped together.

See also the `SFLookup()` and `SFLookupAllToFile()` functions.

Examples

Example 1

```
// Logs in to Salesforce and retrieves the results of a query
records = SFLookupAll("<TAG>endpoint:salesforce/Salesforce</TAG>",
    "SELECT Id,Name FROM Account");
firstId = records[0][0];
firstAccountName = records[0]["name"];
```

Example 2

```
// Logs in to Salesforce and retrieves the results of a query
soql = "SELECT Id,Name,CreatedBy.ContactId,
    CreatedBy.FirstName,CreatedBy.LastName FROM Account";
rs = SfLookupAll("<TAG>endpoint:salesforce/Salesforce</TAG>", soql);
firstId = rs[0][0];
firstAccountName = rs[0]["name"];
lastName = rs[0]["CreatedBy.LastName"];
// or using an index:
last_name = rs[0][5];
```

[Back to Top]

SfLookupAllToFile

Declaration

```
int SfLookupAllToFile(string salesforceOrg, string soql, string targetId)
```

Syntax

```
SfLookupAllToFile(<salesforceOrg>, <soql>, <targetId>)
```

Required Parameters

- **`salesforceOrg:`** A string reference path to a Salesforce connection in the current project
- **`soql:`** A query written in Salesforce Object Query Language (SOQL)
- **`targetID:`** A string activity associated with a file-type endpoint in the current project

Description

Logs into Salesforce (if necessary) and writes the results from Salesforce of the query to a CSV file. The function returns the number of records retrieved.

The login call is made using the credentials in the specified Salesforce endpoint. The Salesforce endpoint used in this function call must be defined as a Salesforce connection in the current project.

The target used in this function call must be defined as an activity associated with a file-type endpoint in the current project. These include configured file share, FTP, HTTP, local storage, and temporary storage activities.

For more information, see the instructions on inserting endpoints under the Endpoints section in Jitterbit Script.

The function returns `null` if the login fails, the query returns no records, or the API fails. Use the `GetLastError()` function to retrieve the error message in that case.

See also the `SFLookup()` and `SFLookupAll()` functions.

Examples

Example 1

```
// Logs in to Salesforce,
// retrieves the results of a query,
// and writes the results to a target
nrec = SFLookupAllToFile("<TAG>endpoint:salesforce/Salesforce</TAG>",
    "SELECT Id,Name FROM Account",
    "<TAG>activity:/...</TAG>");
```

Example 2

```
// Logs in to Salesforce,
// retrieves the results of a query
// specified in a local variable,
// and writes the results to a target
soql = "SELECT Id,Name,CreatedBy.ContactId,
    CreatedBy.FirstName,CreatedBy.LastName FROM Account";
nrec = SfLookupAllToFile("<TAG>endpoint:salesforce/Salesforce</TAG>",
    soql, "<TAG>activity:/...</TAG>");
```

[Back to Top]