Metro EDGE Integration Guide

Contents

[Overview 2](#_Toc452538743)

[Definitions 2](#_Toc452538744)

[Assumptions 2](#_Toc452538745)

[To Embed an EDGE Widget in a third party application 3](#_Toc452538746)

[Step-by-step flow 3](#_Toc452538747)

[To Allow calls from a third party server to an EDGE API 5](#_Toc452538748)

[Step-by-step flow 5](#_Toc452538749)

[Open Issues 8](#_Toc452538750)

# Overview

The purpose of this document is to describe the steps necessary for integration a third party:

* To call REST APIs on the EDGE service layer.
* To embed an EDGE widget in a third party website.

# Definitions

1. **EDGE:** A service layer and UI Widgets provider for various activities that can be performed on the MeroPCS system.
2. **Third Party Backend/Server**: A server that can perform API calls to the EDGE service layer using RESTful http calls.
3. **Third Party Website:** A website that is hosted by a third party entity and can perform activities on the third party backend server. This website has the ability of adding cookie to a user’s browser session.
4. **Embedding Application:**  The third party application that wants to embed a widget that is provided by EDGE.
5. **XSRF-Token**: a specialized token-based mechanism that is transferred between a browser and a web server. The server sends this token as an HTTP cookie, and the browser sends this token back as an HTTP Header. This mechanism is designed to prevent cross-site request forgery attacks. For more information, please read the following article: <https://en.wikipedia.org/wiki/Cross-site_request_forgery>

# Assumptions

1. The EDGE service layer and the third party website will reside under the same domain, e.g. metro.com. This is because all browsers block JavaScript calls from one domain to another domain (a.k.a cross-site scripting attacks). Such configuration can be done at the load-balancer level and be fairly transparent to the actual hosted applications.
2. An alternative to assumption #1 will be to add reverse proxy or load balancer on the host site, to map the URLs to the correct server.
3. This document does not discuss the possibility of using a single sign-on (SSO) server as a means of implementing widgets on third party websites since this solution such a solution can only be provided by MetroPCS and currently does not exist.
4. The solution assumes that the embedding application will initiate a session with the EDGE service layer on behalf of the client/browser. The session has a predefined timeout (currently 30 minutes). A session is required in order to verify that a logged in customer can only perform the activities related to his billing account number.
5. EDGE implements enhanced security to prevent Cross-Site Request Forgery, using XSRF Tokens. It is assumed that the embedding application will support this feature fully. XSRF Tokens are only needed for HTTP requests other than GET.
   1. *Note:* XSRF Tokens are returned in the form of an HTTP cookie once a successful authentication has been performed. It is the responsibility of the client to send that token back in subsequent calls in the form of an HTTP Header. The token and the session id are tied together, so whenever a session expires, that token is considered to be expired as well. Once a new session / login is performed, a new token will be generated and sent from the server to the client in the form of a cookie.
6. All communications between the third party server and the EDGE service must be done over a TLS/SSL connection.
7. EDGE application and third party application use the same key for session id and XSRF token

# To Embed an EDGE Widget in a third party application

## Step-by-step flow

Prerequisite: Third party backend server contains a mapping of its users to a predefined set of proxy users running on EDGE. These proxy users loosely map to the various security profiles that are configured in EDGE.

1. User logs in to the third-party website
2. Third party backend server authenticates the user according to its own existing mechanism.
3. Third party backend server maps the user to one of the predefined proxy users.
4. Third party backend server calls /user URL on the EDGE server using the proxy username and password. The proxy username and password are sent to the server using HTTP Basic Authentication protocol. This means that it is sent as part of the HTTP Header ‘Authorization’. The http spec defines the value for this header as ‘Basic <encrypted data>’, where the encrypted data is the string username:password encoded in Base64. In order to make this call secure, the activity must be done over a TLS/SSL connection.

Here is an example REST call to /user:

**Request headers:**

Accept:application/json, text/plain, \*/\*

Accept-Encoding:gzip, deflate, sdch

Accept-Language:en-US,en;q=0.8,he;q=0.6

Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW1l

Connection:keep-alive

Host:demo.com:7090

Referer:http://demo.com:7090/aot/

User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_11\_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36

WidgetName:USER

**Response headers:**

HTTP/1.1 200 OK

X-Content-Type-Options: nosniff

X-XSS-Protection: 1; mode=block

Cache-Control: no-cache, no-store, max-age=0, must-revalidate

Pragma: no-cache

Expires: 0

X-Frame-Options: DENY

Server:

X-Application-Context: gateway-server:7090

X-Frame-Options: DENY

Cache-Control: no-cache, no-store, max-age=0, must-revalidate

X-Content-Type-Options: nosniff

Pragma: no-cache

Expires: 0

X-XSS-Protection: 1; mode=block

X-Application-Context: aot-server:7091

Date: Mon, 30 May 2016 14:01:23 GMT

Content-Type: application/json;charset=UTF-8

Transfer-Encoding: chunked

**Cookie response set:**

Cookie:XSRF-TOKEN=3a5fd395-4d81-4b44-8bac-f38634a8b8f8; SESSION=a60feaf6-972e-4934-8335-5e8b4d991549

**Response:**

{

"name": "51504",

"roles": [

"ROLE\_SecurityRoles\_canBYOD",

"ROLE\_SecurityRoles\_canRefreshGSMDevice",

"ROLE\_SecurityRoles\_canRefreshDeviceWithLRN",

"ROLE\_Payments\_SetPayment",

"ROLE\_SecurityRoles\_canAssignAffinity",

"ROLE\_Bill\_GetFinancialStatement",

"ROLE\_SecurityRoles\_canCreateRolesReport",

"ROLE\_SecurityRoles\_VPCheckbookCreditAllowed",

"ROLE\_admin",

"ROLE\_Bill\_GetBillSummary",

"ROLE\_SecurityRoles\_selectWarehouse",

"ROLE\_ALL",

"ROLE\_SecurityRoles\_selectRouting",

"ROLE\_SecurityRoles\_ignoreSignatureObtained",

"ROLE\_UamsAdmin"

],

"userFullName": "David Jones",

"clientNgidleTimeout": 1800

}

1. EDGE server verifies the proxy username and password in Amdocs Security Manager. If authenticated successfully, EDGE returns the data on the user, along with a session id and XSRF Token in the response as a cookie.
2. Third party backend server saves the session id and XSRF token on the browser in the form of a cookie.
3. When the third party website wants to display embedded content, he will link to an HTML page hosted on the third party website that will contain references to the EDGE service layer’s widget (javascript and css). The widget will be embedded in the third party website, again – provided both servers are hosted behind the same domain.
4. The third party website may change minor styling settings such as font being used, colors being used, background color etc., in order to match their website’s look and feel. However, various styling configurations cannot be changed by the third party website since it will impact the functionality of the widget (for example, changing font size might cause and overflow of the widget outside the window bounds). Such core styling changes can only be performed by the EDGE development team upon request.
5. Here is an example for an HTML document that hosts a website from EDGE for Reset Bill Cycle. Please note that demo.com:8080 is a placeholder that should be substituted with the correct domain name.

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<!-- Import angular js -->

<script src=*"http://demo.com:8080/aot/vendor/js/angular.js"*></script>

<!-- Import bootstrap CSS -->

<link rel=*"stylesheet"* href=*"http://demo.com:8080/aot/css/bootstrap.css"*>

<!-- Import EDGE custom CSS -->

<link rel=*"stylesheet"* type=*"text/css"* href=*"http://demo.com:8080/widgets/styles/style.css"*>

<!-- Import Widget javascript files (App, Directive, Service and Controller) -->

<script src=*"http://demo.com:8080/aot/vendor/js/ui-bootstrap-tpls.js"*></script>

<script src=*"http://demo.com:8080/widgets/js/widgets/widget-reset-cycle/script/resetBillCycleApp.js"*></script>

<script src=*"http://demo.com:8080/widgets/js/widgets/widget-reset-cycle/script/resetBillCycleDirective.js"*></script>

<script src=*"http://demo.com:8080/widgets/js/widgets/widget-reset-cycle/script/resetBillCycleService.js"*></script>

<script src=*"http://demo.com:8080/widgets/js/widgets/widget-reset-cycle/script/resetBillCycleController.js"*></script>

</head>

<!-- Body -->

<body ng-app=*"resetBillCycle"*>

<!-- embed the widget -->

<reset-bill-cycle-directive></reset-bill-cycle-directive>

</body>

</html>