WHITE PAPER

www.progress.com

ACCESSING THE PROGRESS® OPENEDGE® APPSERVER FROM PROGRESS® ROLLBASE® USING OBJECT SCRIPT

BY EDSEL GARCIA, PRINCIPAL SOFTWARE ENGINEER, PROGRESS OPENEDGE DEVELOPMENT

PROGRESS

TABLE OF CONTENTS

INTRODUCTION

Progress® Rollbase® provides a simple way to create a web-based, multi-tenanted, customizable application that satisfies critical business needs without investing months in development. Progress® OpenEdge® users can take advantage of this rapid application development by accessing existing Progress OpenEdge data from Progress Rollbase.

Progress Rollbase provides scripting capabilities on both the client-side and the server-side. Rollbase scripting capabilities can be used to access the Progress OpenEdge Application Server (AppServer). Using a Script Component on the client-side, you can use the Progress OpenEdge Mobile JavaScript Data Object (JSDO). This process is described more completely in the Progress companion whitepaper, "Accessing the Progress OpenEdge AppServer from Progress Rollbase Using the JSDO." If you would rather access Progress OpenEdge from a Rollbase Server, Progress Rollbase provides a powerful business logic framework that enables automatic workflow by defining the business logic via Triggers. JavaScript code can be called from these Object Script Triggers.

Object Script Triggers are associated with Rollbase Objects. OpenEdge tables can be imported into Progress Rollbase as objects using External Tables. Depending on the application requirements, you can create an Object Script Trigger for an OpenEdge table, and from it call the `rbv _ api.sendJSONRequest()` to call exposed REST operations in OpenEdge services (both Mobile services and REST services). External tables will provide direct access for CRUD (create, read, update and delete) operations to the Progress OpenEdge database.

This technical whitepaper describes how to use the `sendJSONRequest()` API from Object Script on the server-side in Progress Rollbase to call REST operations in OpenEdge Services. These operations can invoke business logic in the Progress OpenEdge AppServer.

COMPONENTS OF A PROGRESS ROLLBASE/OPENEDGE APPSERVER ENVIRONMENT

Rollbase can run in a hosted cloud or private cloud environment. The private cloud requires a Java-based application server. Apache Tomcat is used for development and runtime. In both cases, Rollbase access to OpenEdge only requires that the OpenEdge REST services be exposed on a host or port that is accessible from the Rollbase server.

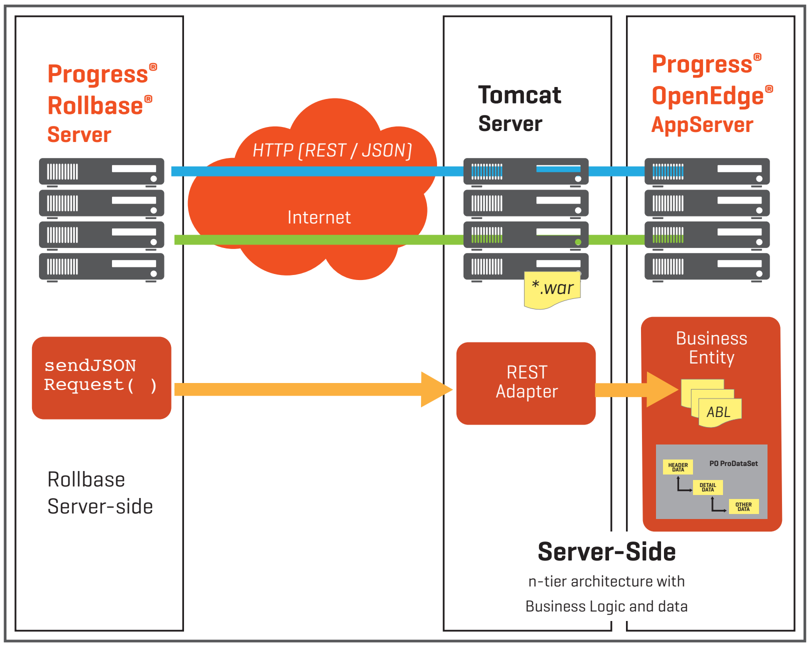


Figure 1. Components of a private cloud environment running Progress Rollbase.

As shown in Figure 1, private cloud components include:

- **Progress Rollbase Server:** A Rollbase Server in a hosted cloud or private cloud environment. The Web server for the REST Adapter and Mobile Service can be the same Tomcat server used by the Rollbase Server.
- **Rollbase server-side:** Rollbase Triggers execute on the Rollbase Server. The `rbv _ api. sendJSONRequest() API` runs from an Object Script Trigger on the server-side. In this scenario, Rollbase is the client to the OpenEdge REST Adapter / AppServer.
- **Tomcat Server (Web server for REST adapter and Mobile service):** The HTTP request sent by `send JSONRequest()` is received by the REST adapter hosted by the Web server.
- **Mobile service:** A Java Web application built using Progress Developers Studio for OpenEdge and deployed as a WAR (Web Application Archive) file to the Tomcat server that provides access to the OpenEdge AppServer via HTTP / HTTPS. Refer to Progress OpenEdge documentation for best practices deploying mobile services for internal accessibility.

- **OpenEdge AppServer:** An application server that can access the OpenEdge database anywhere on the network. OpenEdge Mobile uses the REST adapter component to communicate with OpenEdge AppServer and execute Progress OpenEdge Advanced Business Language (ABL) logic in a Business Entity class.
- **Business Entity:** An ABL class that implements the methods (CRUD operations and user-defined operations) available via the mobile service. This class accesses the OpenEdge database and executes the ABL logic.

CONFIGURATION

In order to use the `sendJSONRequest()` API from Rollbase, the following components must be present in your environment:

- Mobile Service deployed to a Tomcat Server
- OpenEdge AppServer and Business Entity class

Note that any REST Service can be accessed via this method. For the purpose of this whitepaper, we will be focusing on OpenEdge Mobile Services as an exemplary class.

MOBILE SERVICE

To access the OpenEdge AppServer, you must build a mobile service using Progress Developer Studio for OpenEdge and deploy the WAR file for the mobile service to a Java-based application server. For a private cloud environment, you can make use of the same instance of Tomcat as Rollbase, or a separate instance. For a hosted cloud environment, the WAR file needs to be deployed to its own Tomcat instance. Note that during development, the Tomcat instance included with Progress Developers Studio forOpenEdge can be used.

PROGRESS OPENEDGE APPSERVER AND BUSINESS ENTITY CLASS

The OpenEdge AppServer can be anywhere on the network. OpenEdge AppServer is configured to connect to the OpenEdge database. The Business Entity class must be available to the OpenEdge AppServer to execute the ABL logic.

CALLING INVOKE OPERATIONS IN AN OPENEDGE MOBILE SERVICE

OpenEdge Services provide access to Business Entities running on the AppServer. The Business Entity class provides the implementation for the methods (CRUD operations and user-defined operations—also called INVOKE operations) that define the entity (also called a resource). These methods are exposed via HTTP/REST. Figure 2 provides information on the HTTP request for calling an INVOKE operation. For additional information about how URLs are used in OpenEdge Mobile services, refer to “URLs for accessing Mobile Web applications, services, and resources” in *Progress OpenEdge Mobile Applications*.

HTTP VERB	HTTP REQUEST
PUT	**OpenEdge Use Case: INVOKE** **URL:** URL to resource / <method name> **Body:** JSON payload that corresponds to the parameters expected by the method. **Returns:** JSON data that represents the OUTPUT parameters and teh returned value of the method.
Note: The HTTP response for this request can also return a string with an error message. If an operation returns an ABL Error, the JSON data contains an `_retVal` and `_errors` properties.	

Figure 2. Calling INVOKE operations via HTTP.

USING SENDJSONREQUEST() TO CALL AN OPENEDGE SERVICE

The server-side API `rbv_api.sendJSONRequest()` can be used to send HTTP requests to a REST Service that accepts and returns JSON data. In particular, it can be used to call OpenEdge Mobile services and OpenEdge REST services so that Rollbase applications can execute OpenEdge Business Logic.

The `rbv_api.sendJSONRequest()` accepts several parameters and returns a string that corresponds to the body of the HTTP request. Server-side APIs in Rollbase can be used to access Rollbase objects.

The code in Figure 3 provides an example of calling an INVOKE operation on an OpenEdgeMobile service from an Object Script Trigger:

```
01 var CustNum = 3;
02 try {
03     var result = rbv _ api.
sendJSONRequest(
04
05 "http://oemobiledemo.progress.com/
CustomerService/rest/CustomerService/
Customer/"+CustNum+"/GetOrders",
06         null,
07         "PUT",
08         null,
09         "", "",
10         null);
11 }
12 catch(e) {
13     rbv _ api.println('Exception: ' +
e.message);
14 }
15
16 rbv _ api.println(result);
17
18 var jsonObject = rbv _ api.
stringToJson(result);
19
20 rbv _ api.println(jsonObject.response.
eOrder.eOrder.length);
```

Figure 3. Example of code to call an INVOKE operation on an OpenEdge Mobile service.

LINES 01 – 14:

The example sets a variable called `CustNum` to 3. This variable will be used as parameter to the INVOKE operation.

The code calls the `rbv_api.sendJSONRequest()` API passing the following parameters:

- **url:** The URL to INVOKE operation for `GetOrders` which includes the Customer number as a parameter.
- **data:** `Null` is used since the operation does not expect parameter in the body of the request.
- **method:** `PUT` is specified to perform an INVOKE operation.
- **contentType:** `Null` is used since the default content-type is application/json.
- **username/password:** Empty string `("", "",)` is specified for the username and password to indicate an anonymous connection.
- **headers:** Null is used since the HTTP headers do not need to be overwritten.

The response of the request is a string that corresponds to the body of the HTTP response; in this case, JSON data representing the output parameters. Multiple output parameters can be returned for an INVOKE operation. The JSON data includes a "response" property which then includes the parameters–in this case, the JSON representation for a dataset called "`eOrder`."

Exceptions can be captured by using a catch block. The example uses property `e.message` to obtain the message corresponding to the exception. A property's name and stack can also be queried.

LINE 16:

As an example, the text of the response is printed using the `rbv_api.println()`.

LINE 18:

The `rbv_api.stringToJson()` is used to convert the response of the request to a JSON (JavaScript) object so that the data in the response can be handled from JavaScript.

LINE 21:

The `rbv_api.println()` is used to print the number of order records in the eOrder table of the `eOrder` dataset.

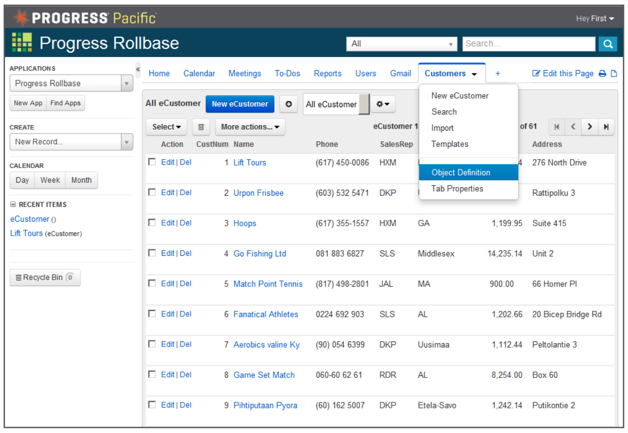
NOTES:

1. The Object Script Trigger can call APIs, such as `rbv _ api.getFieldValue()` and `rbv _ api.setFieldValue()`, to access the Rollbase Objects. For example, these APIs can obtain the values to be used as input parameters when invoking an operation or to update Rollbase Objects that would present the result of the operation.
2. The APIs `rbv _ api.stringToJson()` and `rbv _ api.jsonToString()` can be used to parse a string into a JavaScript object and serialize a JavaScript object into a string respectively.
3. The size of the JSON data that `rbv _ api.stringToJson()` can process depends on the system-level `FormulaSize` property. This property, defined in the `shared.properties` file, can be increased if large responses are expected to be returned and processed.

STEPS TO CALL A SENDJSONREQUEST() FROM AN OBJECT SCRIPT TRIGGER

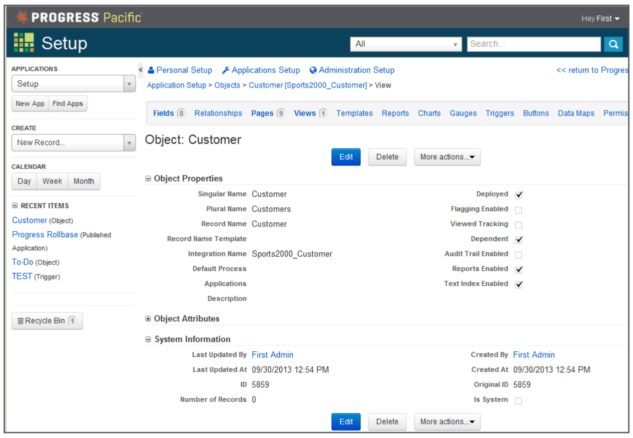
Follow these steps to call a `sendJSONRequest()` from an Object Script Trigger:

1. Select the desired Object from the Rollbase home page.
2. Select **Triggers** from the Object page.



Action	CustNum	Name	Phone	SalesRep			Address
Edit | Del	1	Lift Tours	(617) 450-0086	HXM			276 North Drive
Edit | Del	2	Urpon Frisbee	(603) 532 5471	DKP			Rattipolku 3
Edit | Del	3	Hoops	(617) 355-1557	HXM	GA	1,199.95	Suite 415
Edit | Del	4	Go Fishing Ltd	081 883 6827	SLS	Middlesex	14,235.14	Unit 2
Edit | Del	5	Match Point Tennis	(817) 498-2801	JAL	MA	900.00	66 Homer Pl
Edit | Del	6	Fanatical Athletes	0224 692 903	SLS	AL	1,202.66	20 Bicep Bridge Rd
Edit | Del	7	Aerobics valine Ky	(90) 054 6399	DKP	Uusimaa	1,112.44	Peltolantie 3
Edit | Del	8	Game Set Match	060-60 62 61	RDR	AL	8,254.00	Box 60
Edit | Del	9	Pihtiputaan Pyora	(60) 162 5007	DKP	Etela-Savo	1,242.14	Putikontie 2

Figure 4. Progress Rollbase home page



Application Setup > Objects > Customer [Sports2000_Customer] > View

Object: Customer

Object Properties

Singular Name	Customer	Deployed	✓
Plural Name	Customers	Flagging Enabled	☐
Record Name	Customer	Viewed Tracking	☐
Record Name Template		Dependent	✓
Integration Name	Sports2000_Customer	Audit Trail Enabled	☐
Default Process		Reports Enabled	✓
Applications		Text Index Enabled	✓
Description			

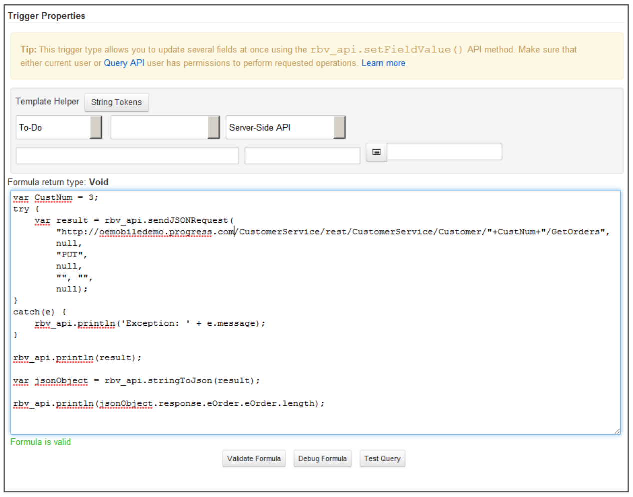
Object Attributes

System Information

Last Updated By	First Admin	Created By	First Admin
Last Updated At	09/30/2013 12:54 PM	Created At	09/30/2013 12:54 PM
ID	5859	Original ID	5859
Number of Records	0	Is System	☐

Figure 5. Customer Object selected from the Rollbase home page

3. Select **New Trigger**.
4. Specify **Object Script** as the **Trigger Type**.
5. Select **Next** and specify the **Trigger Name** and the **Integration Name**.
6. In the **Trigger Properties** section, enter the JavaScript code to call `sendJSONRequest()`. This code would look similar to that shown in Figure 6. Refer to example 1 in Appendix A for a working example.
7. Use the buttons **Validate Formula** and **Debug Formula** to debug the JavaScript code to ensure that it is working correctly. You can use `rbv _ api.println()` to output specific values while debugging.
8. Select **Save** to save your changes. You can run the Trigger from the Rollbase interface or from JavaScript using the `rbf _ runTrigger()` API from the client-side or the `rbv _ api.runTrigger()` API from the server-side.



Trigger Properties

Tip: This trigger type allows you to update several fields at once using the `rbv_api.setFieldValue()` API method. Make sure that either current user or Query API user has permissions to perform requested operations. Learn more

Formula return type: **Void**

```
var CustNum = 3;
try {
    var result = rbv_api.sendJSONRequest(
        "http://oemobiledemo.progress.com/CustomerService/rest/CustomerService/Customer/"+CustNum+"/GetOrders",
        null,
        "PUT",
        null,
        "", "",
        null);
}
catch(e) {
    rbv_api.println('Exception: ' + e.message);
}

rbv_api.println(result);

var jsonObject = rbv_api.stringToJson(result);

rbv_api.println(jsonObject.response.eOrder.eOrder.length);
```

Formula is valid

Figure 6. Object Script Trigger with code to call INVOKE operation GetOrders.

BENEFITS

Using the Progress Rollbase platform, you can quickly create an application that calls Progress OpenEdge data and services. These applications can extend your customer's experience of working with your critical business applications outside of the office—boosting both your reach and your reputation without draining your resources.

PROGRESS SUPPORT SERVICES

Progress offers worldwide technical support and professional services around the clock and around the globe tailored to your business' individualized needs. Our self-service, global support centers and tiered-level service agreements combine to protect the investments you have made in your technology. For more information, visit progress.com/support-and-services.

ABOUT THE AUTHOR

Edsel Garcia is a Principal Engineer in the Progress OpenEdge Engineering group. Edsel has a long history of first-hand experience using Progress products, starting as a customer and application developer about 23 years ago. During his 16-year tenure at Progress, Edsel has been a member of Customer Support, Solution Engineering development, the Tooling development team, the OpenEdge Architect product development team, the OpenEdge Management team, and the Core Client team as a member of the OpenEdge Mobile development initiative.

ABOUT PROGRESS ROLLBASE

Progress Rollbase is a cloud platform for development and delivery of software as a service (SaaS) business applications using point & click, drag & drop tools in a standard browser with a minimal amount of code. Progress Rollbase delivers on the promise of rapid application development (RAD), making application creation much faster than traditional software development methods. Progress Rollbase is offered as both a hosted service (Progress Rollbase Hosted Cloud) and as an installable product (Progress Rollbase Private Cloud) that can be deployed on any cloud infrastructure or on-premises. Progress Rollbase is designed for enterprises as well as independent software vendors (ISVs) with a complete system for tenant and subscriber management, provisioning, application development, publishing and deployment. Progress Rollbase provides ISVs and resellers with complete white label capabilities making it easy to deploy applications under any brand or identity. For more information about the Progress Rollbase platform, please visit www.progress.com/rollbase.

PROGRESS

APPENDIX A: EXAMPLES CALLING AN INVOKE OPERATION IN A PROGRESS OPENEDGE MOBILE SERVICE

EXAMPLE 1: CALLING AN INVOKE OPERATION ON AN APPLICATION SERVER USING SENDJSONREQUEST()

The following example demonstrates how to call an INVOKE operation on an OpenEdge Mobile Service using sendJSONRequest() from an Object Script Trigger in Progress Rollbase. The example returns JSON data representing parameters that include a ProDataSet.

```
var CustNum = 3;

try {
    var result = rbv_api.sendJSONRequest(
        "http://oemobiledemo.progress.com/CustomerService" +
        "/rest/CustomerService/Customer/” + CustNum + “/GetOrders",
        null, "PUT",
        null, null, null,
        null);
}
catch(e) {
    rbv_api.println('Exception: ' + e.message);
}

rbv_api.println(result);

var jsonObject = rbv_api.stringToJson(result);
rbv_api.println(jsonObject.response.eOrder.eOrder.length);
```

EXAMPLE 2: CALLING AN INVOKE OPERATION ON AN APPLICATION SERVER USING SENDJSONREQUEST()

The following example demonstrates how to call an INVOKE operation on an OpenEdge Mobile Service using sendJSONRequest() from an Object Script Trigger in Rollbase. This example returns JSON data representing parameters that include a return value.

```
try {
    var result = rbv_api.sendJSONRequest(
        "http://oemobiledemo.progress.com/CustOrderService" +
       "/rest/CustOrderService/CustOrder/CheckCredit",
        null, "PUT",
        null, null, null,
        null);
}
catch(e) {
    rbv_api.println('Exception: ' + e.message);
}

rbv_api.println(result);

var jsonObject = rbv_api.stringToJson(result);
rbv_api.println(jsonObject.response._retVal);
```

APPENDIX B: EXAMPLES INVOKING CRUD OPERATIONS IN AN OPENEDGE MOBILE SERVICE

OPENEDGE SERVICES PROVIDE ACCESS TO BUSINESS ENTITIES RUNNING ON THE APPSERVER

The Business Entity class provides the implementation for the methods (CRUD operations and user-defined operations – also called INVOKE operations) that define the entity (also called a resource). In addition to INVOKE operations, the sendJSONRequest() API can be used to send GET, POST and PUT requests to perform READ, CREATE and UPDATE operations.Figure B1 provides a summary format of the HTTP requests for CRUD operations.

HTTP VERB	HTTP REQUEST
PUT	**OpenEdge Use Case: INVOKE** **URL:** URI to resource / <method name> **Body:** JSON payload that corresponds to the parameters expected by the method. **Returns:** JSON data that represents the OUTPUT parameters and teh returned value of the method.
POST	**OpenEdge Use Case: CREATE** **URL:** URI to resource **Body:** JSON payload representing the new record. **Returns:** JSON data with the new record. The returned record can include changes done on the application server; for example, performed by a database trigger.
GET	**OpenEdge Use Case: READ** **URL:** URI to resource **Query String:** Optional filter parameter. **Returns:** JSON data representing the records.
DELETE	**OpenEdge Use Case: DELETE** **URL:** URI to resource **Body:** JSON payload representing the record to delete. **Returns:** JSON data indicating the record to delete. Note: Not supported in this release. Use an INVOKE operation to call a method to delete the records.
Notes: 1. The HTTP response for these requests can also return a string with an error message. If an operation returns an ABL Error, the JSON data contains `an _ retVal` and `_ errors` properties. 2. It is recommended to primarily access business logic by calling INVOKE operations. For CRUD operations we recommend using the External Table feature in Rollbase.	

Figure 7. HTTP requests for CRUD operations.

EXAMPLE 1: CALLING A CREATE OPERATION ON THE OPENEDGE APPSERVER USING SENDJSONREQUEST()

The following example demonstrates how to call a CREATE operation on an OpenEdge Mobile Service using `sendJSONRequest()` from an Object Script Trigger in Rollbase. The operation expects JSON data to represent the record to be created, and returns JSON data that corresponds to the newly created record as returned from the server. It can include changes made by database Triggers or additional business logic.

Notice that `CustNum` is specified as 0. The code in the Business Entity excludes `CustNum` when saving the values, so that values generated using the CREATE database trigger are used.

```
try {
    var response = rbv_api.sendJSONRequest(
        "http://oemobiledemo.progress.com/CustomerService" +
        "/rest/CustomerService/Customer",
        {"eCustomer":[
            {"CustNum":"0","Name":"TEST","Address":"",
             "Phone":"","SalesRep":"","Balance":"0",
             "State":""}]},
        "POST",
        null, null, null,
        null);
}
catch(e) {
    rbv_api.println('Exception: ' + e.message);
}

rbv_api.println(response);

var dataSet = rbv_api.stringToJson(response);
rbv_api.println(dataSet.dsCustomer.eCustomer[0].CustNum);
```

EXAMPLE 2: ON THE OPENEDGE APPSERVER USING SENDJSONREQUEST()

The following example demonstrates how to call a READ operation on an OpenEdge Mobile service using `sendJSONRequest()` from an Object Script Trigger in Rollbase. The operation returns JSON data that represents the records of the ProDataSet.

```
try {
    var response = rbv_api.sendJSONRequest(
        "http://oemobiledemo.progress.com/CustomerService" +
        "/rest/CustomerService/Customer",
        {"eCustomer":[
            {"CustNum":"0","Name":"TEST","Address":"",
             "Phone":"","SalesRep":"",”Balance":"0",
             "State":""}]},
        "POST",
        null, null, null,
        null);
}
catch(e)  {
        rbv_api.println('Exception:  ' + e.message);
}

rbv_api.println(response);

var dataSet = rbv_api.stringToJson(response);
rbv_api.println(dataSet.dsCustomer.eCustomer[0].CustNum);
```

EXAMPLE 3: CALLING AN UPDATE OPERATION ON OPENEDGE APPSERVER USING SENDJSONREQUEST()

The following example demonstrates how to call an UPDATE operation on an OpenEdge Mobile service using `sendJSONRequest()` from an Object Script Trigger in Rollbase. The operation expects JSON data to represent the record being updated.

`CustNum` is specified. The Operation returns JSON data that corresponds to the record as returned from the server which can include changes made by database triggers or additional business logic.

```
try {
    var response = rbv_api.sendJSONRequest(
        "http://oemobiledemo.progress.com/CustomerService" +
       "/rest/CustomerService/Customer",
       null, "GET",
        null, null, null,
        null);


}
catch(e) {
    rbv_api.println('Exception: ' + e.message);
}


rbv_api.println(response);


var dataSet = rbv_api.stringToJson(response);
rbv_api.println(dataSet.dsCustomer.eCustomer.length);
```

EXAMPLE 4: CALLING AN INVOKE OPERATION ON THE APPLICATION SERVER USING SENDJSONREQUEST() TO DELETE RECORDS

The following example demonstrates how to perform a delete on an OpenEdge Mobile service using

`sendJSONRequest()` from an Object Script Trigger in Rollbase. The operation expects JSON data to represent the record to be deleted.

`CustNum` is specified. The operation returns an empty payload. An INVOKE operation is used to delete the records.

```
try {
    var response = rbv_api.sendJSONRequest(
        "http://oemobiledemo.progress.com/CustomerService" +
        "/rest/CustomerService/Customer/DeleteCustomer",
       {"request":
           {"eCustomer":[{"CustNum":"3225","Name":"TEST","Address":"",
            "Phone":"","SalesRep":"","Balance":"0",
            "State":""}]}},
        "PUT",
       null, null, null,
        null);
}
catch(e) {
    rbv_api.println('Exception: ' + e.message);
}

rbv_api.println(response);
```

EXAMPLE 5: CALLING A READ OPERATION WITH A FILTER USING SENDJSONREQUEST()

The following example demonstrates calling a READ operation on an OpenEdge Mobile service using `sendJSONRequest()` from an Object Script Trigger in Rollbase. The operation returns JSON data that represents the records of the ProDataSet. Parameters are passed in the query string of the URL.

```
try {
    var response = rbv_api.sendJSONRequest(
        "http://oemobiledemo.progress.com/CustomerService" +
        "/rest/CustomerService/Customer/DeleteCustomer",
        {"request":
            {"eCustomer":[{"CustNum":"3225”,"Name":"TEST","Address":"",
             "Phone":"","SalesRep":"",”Balance":"0",
             "State":""}]}},
        "PUT",
        null, null, null,
        null);
}
catch(e) {
    rbv_api.println('Exception: ' + e.message);
}

rbv_api.println(response);
```