Peacekeeper Enterprises, LLC.

http://www.peacekeeper.com/

Auto Tables for RAD Server

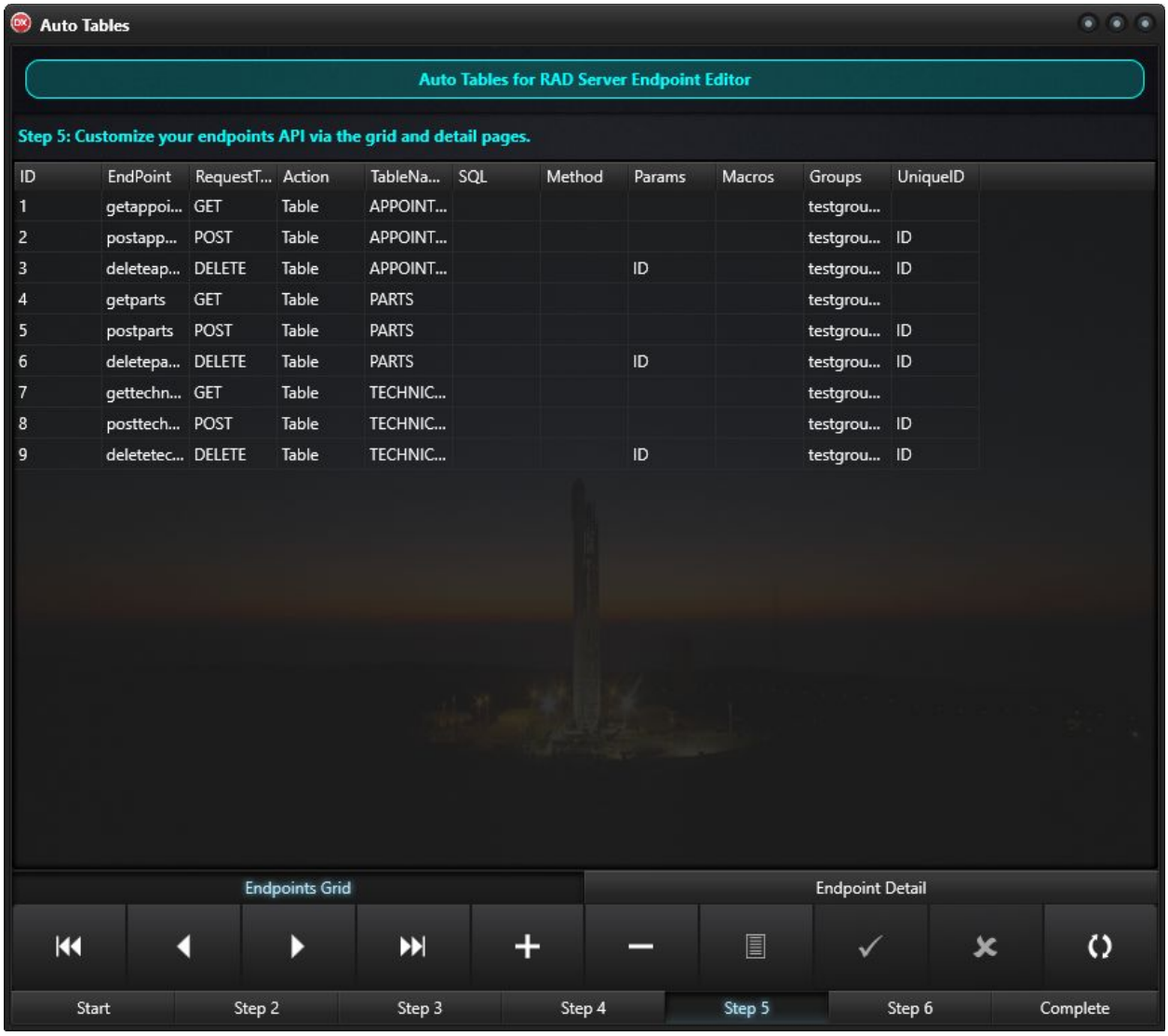
October 22, 2018

Product Overview

Auto Tables is an automatic low code REST API generator for RAD Server. The easy to use interface allows for the quick automatic configuration of a REST server with dynamic endpoints providing read, write, and delete access to your data. Database tables can be made available with enterprise permissions for over 30 different databases including databases such as MySQL, Microsoft SQL Server, and InterBase. Additionally, you can automatically generate endpoints for over 80 more data sources using the Embarcadero Enterprise Connectors.

A graphical Auto Tables for RAD Server Endpoint Editor is provided to help you set up, create, and edit your Auto Tables for RAD Server endpoints. Once you configure your endpoints you can either generate a new RAD Server project or you can save them out for loading into an existing Auto Tables for RAD Server ResourceModule. The configuration format is the standard FireDAC JSON.

Full source code for the Auto Tables RAD Server ResourceModule is provided so you control your REST API server. This gives you the freedom to enhance and modify the source code as needed for your own solutions.



License

Auto Tables for RAD Server Editor QuickStart

The TFDConnection component in the Editor needs to be setup to point at the database you want to generate the REST endpoints from. The existing TFDConnection will allow you to connect to an InterBase database. On Step 3 of the editor you can choose the InterBase database to connect to or you can

uncheck the Load InterBase Database check box to disable loading of the InterBase database file. If you uncheck the checkbox it will rely entirely on your settings in the TFDConnection component. You can safely delete and add a new TFDConnection component as long as you name your new component the same name which is FDConnection.You can use the Data Explorer in the RAD Studio IDE to create a connection to your database or you can set up a TFDConnection component manually. Be sure to include any needed TFDPhys*DriverLink components as well.

Once the FDConnection is setup you can compile the Editor and go through each step. Step 2 requires that you have a RAD Server instance running to be able to connect and get the list of RAD Server groups for permissions.

EndPoints

Auto Tables endpoints utilize a ResourceName for the root segment and then a single dynamic segment for the specific endpoint. The default ResourceName is v1. Parameters are passed in via the query string. An example endpoint would be:

/v1/myendpoint/?PageId=1

API Versioning

You can utilize the ResourceName segment to create different versions of your API. The default name is v1 for version 1 but you can change it to v2, v3, v4 etc. when you release different versions of the same API. By utilizing version numbers you can run different versions side by side and eventually depreciate older versions.

Request Types

Auto Tables makes the GET, POST, and DELETE HTTP/S request types available as endpoints in RAD Server.

GET

If RequestType is set to GET then a standard HTTP/HTTPS request with a query string coming in will trigger that request. GET Actions include Table, SQL, AggregateSQL, and Method.

- Table - TableName field will be used to return the full database table
- SQL - SQL field will be used to execute the SQL query and return the result
- AggregateSQL - SQL field will be used to execute the SQL query list and return the results
- Method - Method field will be used to execute the custom Method

POST

If you set Request Type to POST the defined endpoint will expect a FireDAC JSON format to be submitted in the body of the request. POST Actions include Table, SQL, AggregateSQL, and Method. The FireDAC JSON will be loaded into an TFDMemTable. If the auto increment field (ID) is zero Auto Tables will create new records with the submitted rows. If the auto increment field (ID) is greater than zero Auto Tables will edit and replace the existing record with that auto increment id using the data from the submitted row.

- Table - TableName field will be used to append or edit submitted records
- SQL - SQL field will be used to execute a SQL query and return the result
- AggregateSQL - SQL field will be used to execute a SQL query list and return the results
- Method - Method field will be used to execute the custom Method

DELETE

If you set Request Type to DELETE the defined endpoint will require an auto increment field (ID) parameter to be passed to it. A DELETE SQL statement will be executed to delete the auto increment id that was passed to the endpoint. DELETE Actions include Table, SQL, AggregateSQL, and Method.

Actions

There are a number of actions that can be taken when an Auto Tables endpoint is requested. These Actions include Table, SQL, AggregateSQL, and Method.

Table

Table will automatically return a database table from an endpoint via TFDTable as FireDAC JSON.

SQL

SQL will automatically execute the set SQL statement via TFDQuery and return FireDAC JSON when the endpoint is requested. The Params field can be used to define parameters such as PageId and Limit.

AggregateSQL

Aggregate SQL is a collection of SQL statements which are all executed and returned in a single response. Each SQL statement is executed separately and its result is added to an TFDMemTable. Once all of the SQL statements have been executed the TFDMemTable is returned from the endpoint as FireDAC JSON.

Method

Method actions allows you to define a custom method for each endpoint which will be requested. The custom methods must be defined in the Auto Tables for RAD Server ResourceModule code and have direct access to AContext, ARequest, and AResponse in RAD Server.

TableName

The TableName field is used by the GET, POST, and DELETE RequestTypes. When the RequestType is GET the TableName field is used to define the name of the table that will be returned via the endpoint when the Action is set to Table. When the RequestType is POST the

TableName field is used to determine on which database table to append or edit records. When the RequestType is DELETE the TableName field will be used as the database table on which the DELETE SQL statement will be executed against.

SQL

The SQL field is used to hold the SQL the SQL Action and the SQL list for the AggregateSQL Action. The SQL list for the AggregateSQL Action consists of name=value pairs where name is the name of the field that will be returned in the FireDAC JSON from the endpoint and value is the SQL statement used to fill the name field with a result. Any number of name=value pairs may be used in the AggregateSQL field limited only by the capacity of your database to execute all of the queries and the length of the SQL field itself. The SQL field accepts FireDAC style parameters in the format of :PageId which get resolved to the parameters listed in the Params field.

SQL Field Example For SQL Action

```
SELECT * FROM Log LIMIT :PageId,25
```

SQL Field Example For AggregateSQL Action

```
log_count=SELECT COUNT(*) FROM Log;
most_recent=SELECT last_modified as most_recent FROM Log LIMIT 0,1;
most_recent_error=SELECT message as most_recent_error FROM Log LIMIT 0,1;
```

Built In Default Params

There are a number of built in parameters you can use in your SQL statements. These include :UserId, :Username, :TenantId, and :Body. UserId returns the RAD Server UserId. Username returns the RAD Server Username. TenantId returns the current TenantId that is being passed in to RAD Server. And Body returns the contents of the POST body.

Method

The Method field is used to define the name of the custom method in the Auto Tables ResourceModule for RAD Server that will be executed via the endpoint when the Action is set to Method. You can create your own methods using the sample code below. Simply rename the customMethod() function as needed and utilize that name “customMethod” in the Method field.

```
function TAutoTablesResource.customMethod(AContext: TEndpointContext; ARequest: TEndpointRequest; AResponse: TEndpointResponse): TMemoryStream;

end;
```

Params

The Params field contains a comma delimited list of parameters that can be passed in to the query string on an endpoint. The parameters are made available or utilized by the various actions.

Reserved Params

The format parameter is reserved for Auto Tables to allow you to return one of 4 formats. You can pass ?format=CSV to return the data as a CSV file. You can pass ?format=XML to return the FireDAC results as XML. You can pass ?format=BINARY to return the FireDAC results as binary. And finally the default format is FireDAC JSON so any other value or an omitted format parameter will return FireDAC JSON.

Params for the SQL and AggregateSQL Actions

For the SQL and AggregateSQL action the parameters are available as parameters in the SQL query. The parameters are automatically inserted as properties and are quoted properly to handle escape characters. For example if a PageId parameter is defined in the Params field the

:PageId variable would be available for use in the SQL field such as “SELECT * FROM Log LIMIT :PageId,25”.

Params for Method Actions

The parameters defined in the Params field are directly available to you via the ARequest object in the custom methods.

Macros

The Macros field contains a comma delimited list of macros that can be passed in to the query string on an endpoint. The macros can be used in the SQL field and provide access to the FireDAC Macros functionality. Macros can contain SQL database fields and are directly substituted in the SQL string.

Macros for the SQL and AggregateSQL Actions

For the SQL and AggregateSQL action the macros are available as direct string replacement macros in the SQL query. It utilizes the !macro syntax provided in FireDAC. However, the macro strings are sanitized against the list of fields in the table.

Groups

Users and Groups are already built into RAD Server. You can have users in multiple different groups. The Groups field in Auto Tables for RAD Server should contain a comma delimited list of RAD Server Groups that should have access to this endpoint. A blank Groups field would allow users who are not logged into RAD Server to access the endpoint. Utilizing this feature you can create a wide variety of permissions for access to the endpoints.

UniqueID

The UniqueID field is used to hold the UniqueID field name for the POST and DELETE RequestTypes. Both RequestTypes utilize a primary key type field to know which field in the

table to use for adding, editing, and deleting records. The UniqueID field allows you to customize the name of that unique id field. By default the field name is ID.

Auto Tables for RAD Server Endpoint Editor

The Endpoint Editor allows you to automatically create, quickly customize, and easily edit all of your RAD Server endpoints. It is built in a wizard type interface with multiple easy to use steps. Everything you need to get your REST API server up and running fast is auto generated for you.

It will automatically generate a Delphi RAD Server project, a Delphi REST Client project to connect to RAD Server, and an OpenAPI (Swagger) specification file for your REST API. Additionally, you can copy the dataset component for the server endpoints to the clipboard or you can copy the REST endpoint connection components for the client to the clipboard.

Copy DataSet Component

You can copy the TFDMemTable with all of the endpoint configuration in it to the clipboard. It can then be easily pasted into an existing project.

Generated RAD Server Project

The automatically generated Auto Tables for RAD Server project has all the code already in it for handling the endpoints you have configured in the editor. The project is created from a template in the templates subdirectory. You can edit the template prior to generating your Auto Tables for RAD Server project and customize it as needed.

Copy REST Components

You can copy all of the REST components that needed to connect to the Auto Tables for RAD Server into the clipboard. They can then be easily pasted into an existing project.

Generated REST Client Project

The automatically generated Auto Tables for RAD Server REST Client project has all the code already in it for connecting to your Auto Tables for RAD Server endpoints that you have configured in the editor. The project is created from a template in the templates subdirectory. You can edit the template prior to generating your Auto Tables for RAD Server REST Client project and customize it as needed. Additionally, a Delphi SDK is also generated for the Auto Tables for RAD Server. Generation of SDKs for other languages can easily be added to the modular SDK generator.

Generated OpenAPI

In addition to the server project and client project that can be generated by editor it will also generate an OpenAPI specification document in JSON format. The OpenAPI file can be used to automatically generate SDKs or client projects using tools like OpenAPI-CodeGen or SwaggerHub to automatically create clients in a variety of languages like ActionScript, Ada, Apex, Bash, C#, C++, Clojure, Dart, Elixir, Elm, Eiffel, Erlang, Go, Groovy, Haskell, Java, Kotlin, Lua, Node.js, Objective-C, Perl, PHP, PowerShell, Python, R, Ruby, Rust, Scala, Swift, and Typescript.

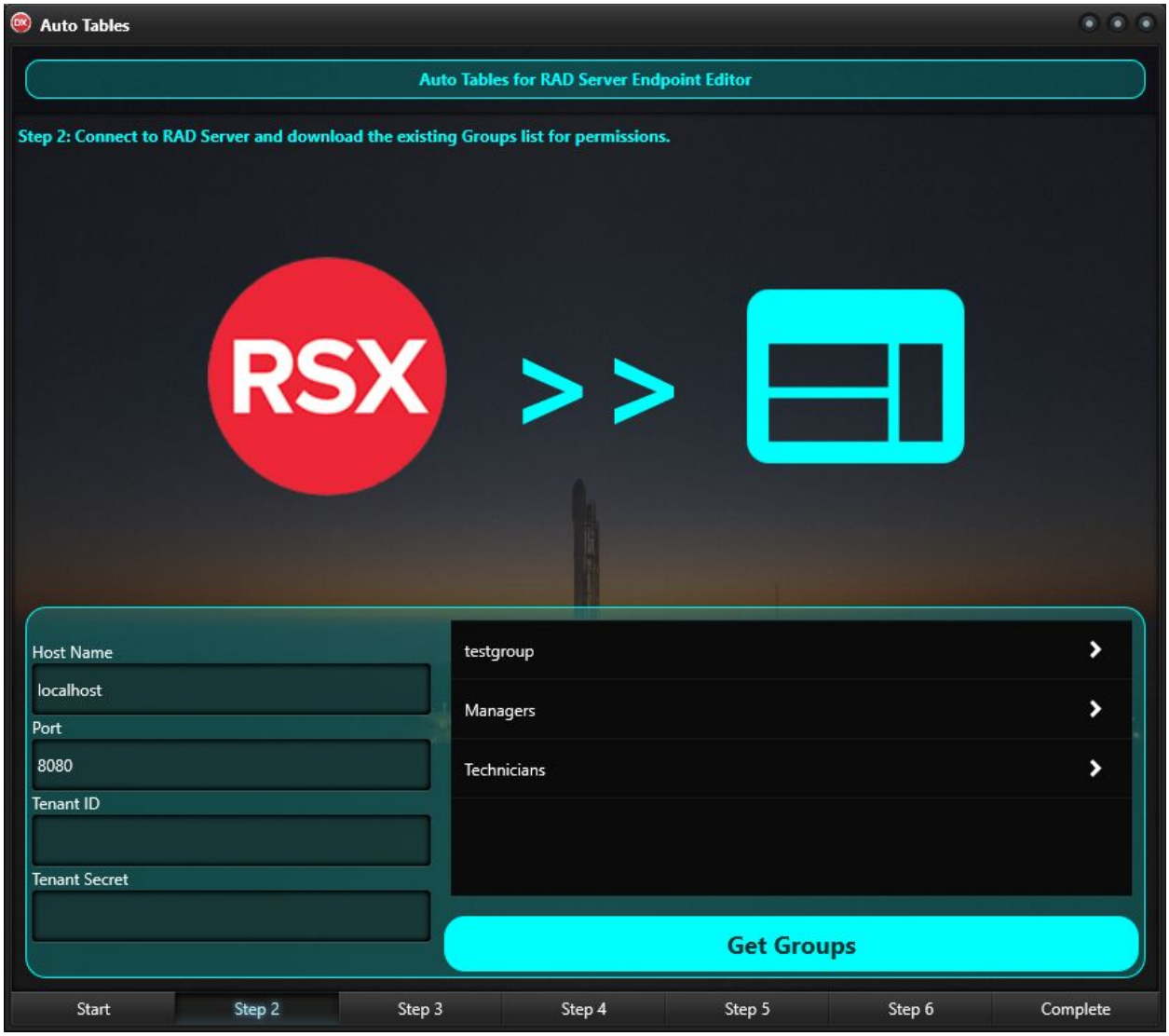
The OpenAPI spec generator has a number of different fields it needs filled out to generate a propert OpenAPI spec. These fields include Title, Version, Description, Terms of Service, Contact Name, Contact Email, Contact URL, License Name, and License URL.

Auto Tables Editor Walkthrough

Step 1

If you have an existing Auto Tables for RAD Server endpoints file you can load it on the first step. Otherwise you can skip to the next step.

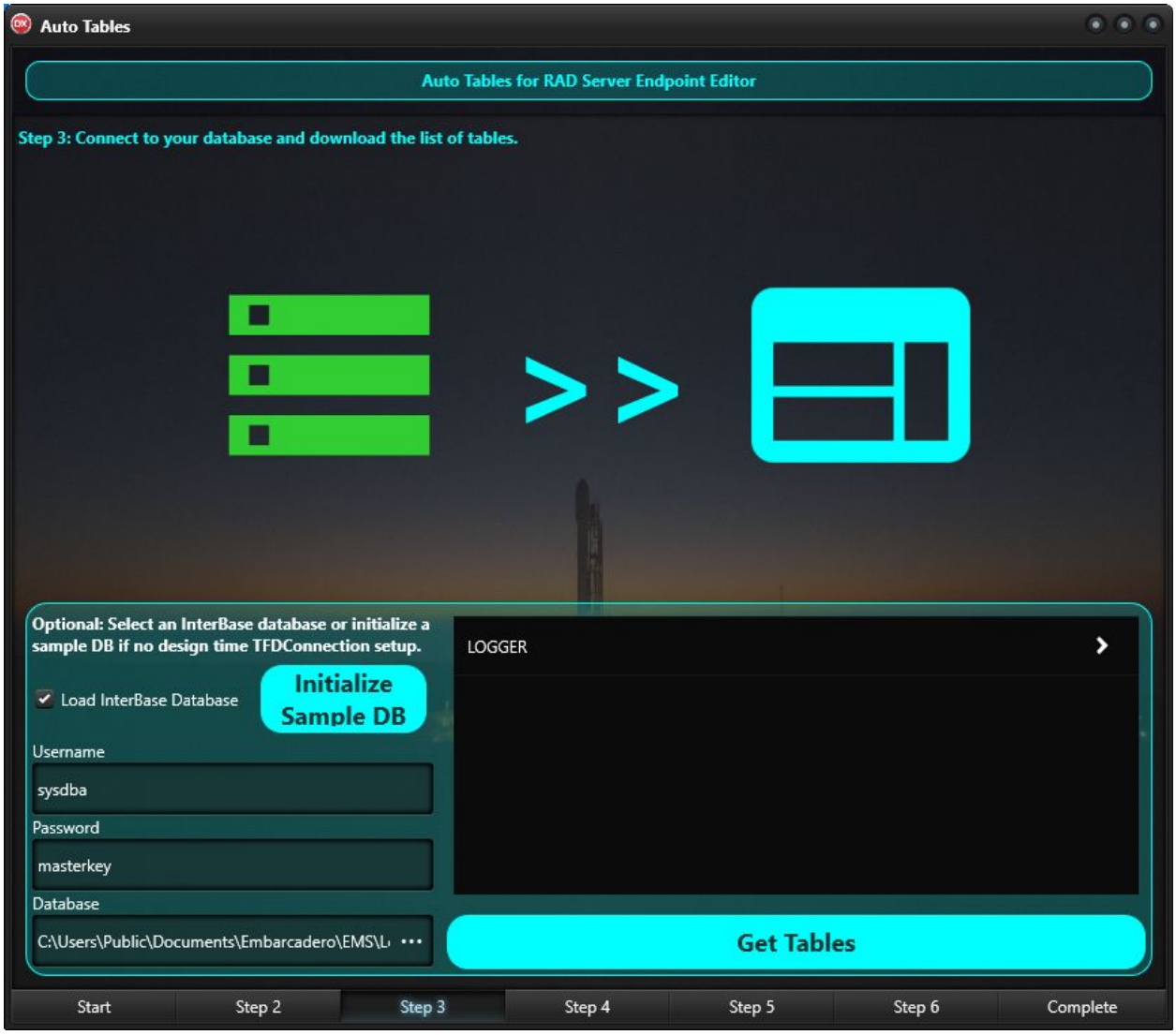
Step 2



An instance of RAD Server should be running in order to complete Step 2. The Editor will connect to your RAD Server instance and download the list of Groups. The Groups are used later to allow you to configure permissions. When ready you can click Get Groups to populate the

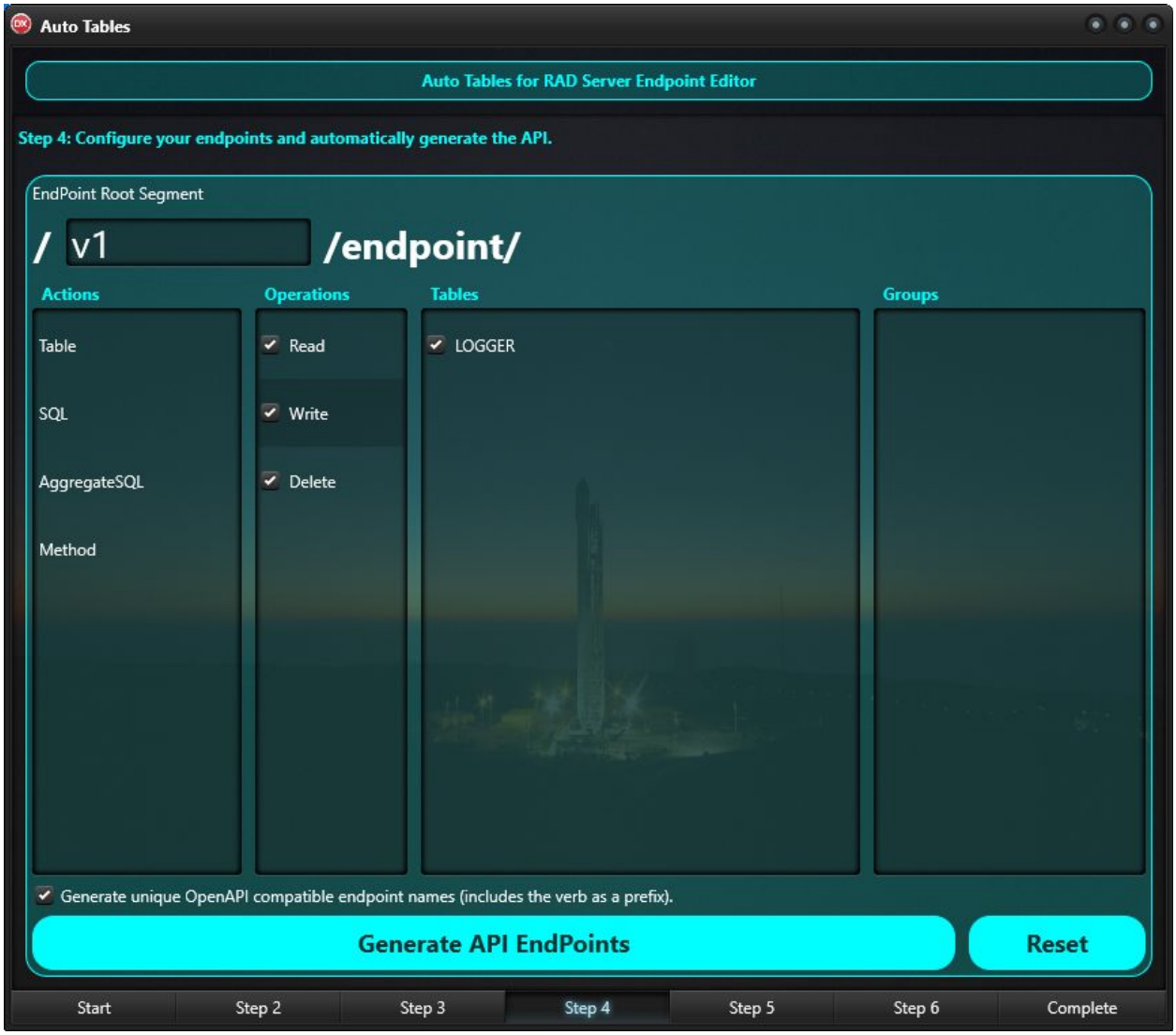
groups data automatically. You can skip this step if you want to configure permissions manually.

Step 3



In Step 3 you can either connect to an InterBase database or you can connect to your existing database that you set up at design time. If you are not using InterBase you should uncheck the checkbox. When ready you can click Get Tables to populate the tables data automatically.

Step 4



In Step 4 you can configure the root segment of your REST API endpoints. By default the root segment is set to v1. This allows you to create different versions of your REST API and run the different versions side by side. If you were creating version 2 you would set the root segment to v2.

In the Actions section you can set which Action you want to be used for the auto generated REST API. By default the Table Action is used which will operate on a standard Table.

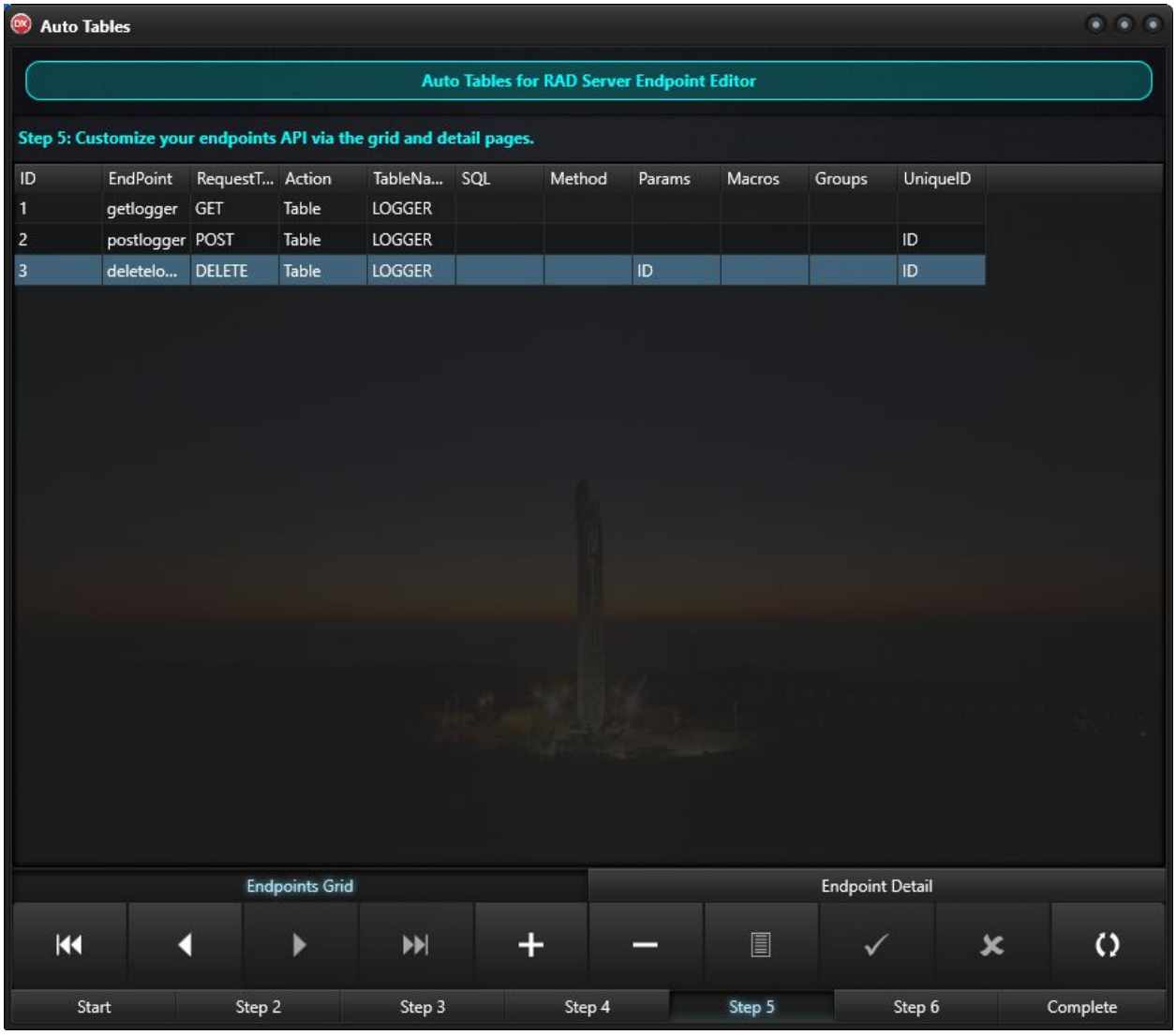
In the Operations section you can choose which Operations you want to be generated for those Actions. Operations include Read, Write, and Delete.

In the Tables section you can choose which tables you would like the REST API generated for. The Tables are populated in Step 3 of the editor.

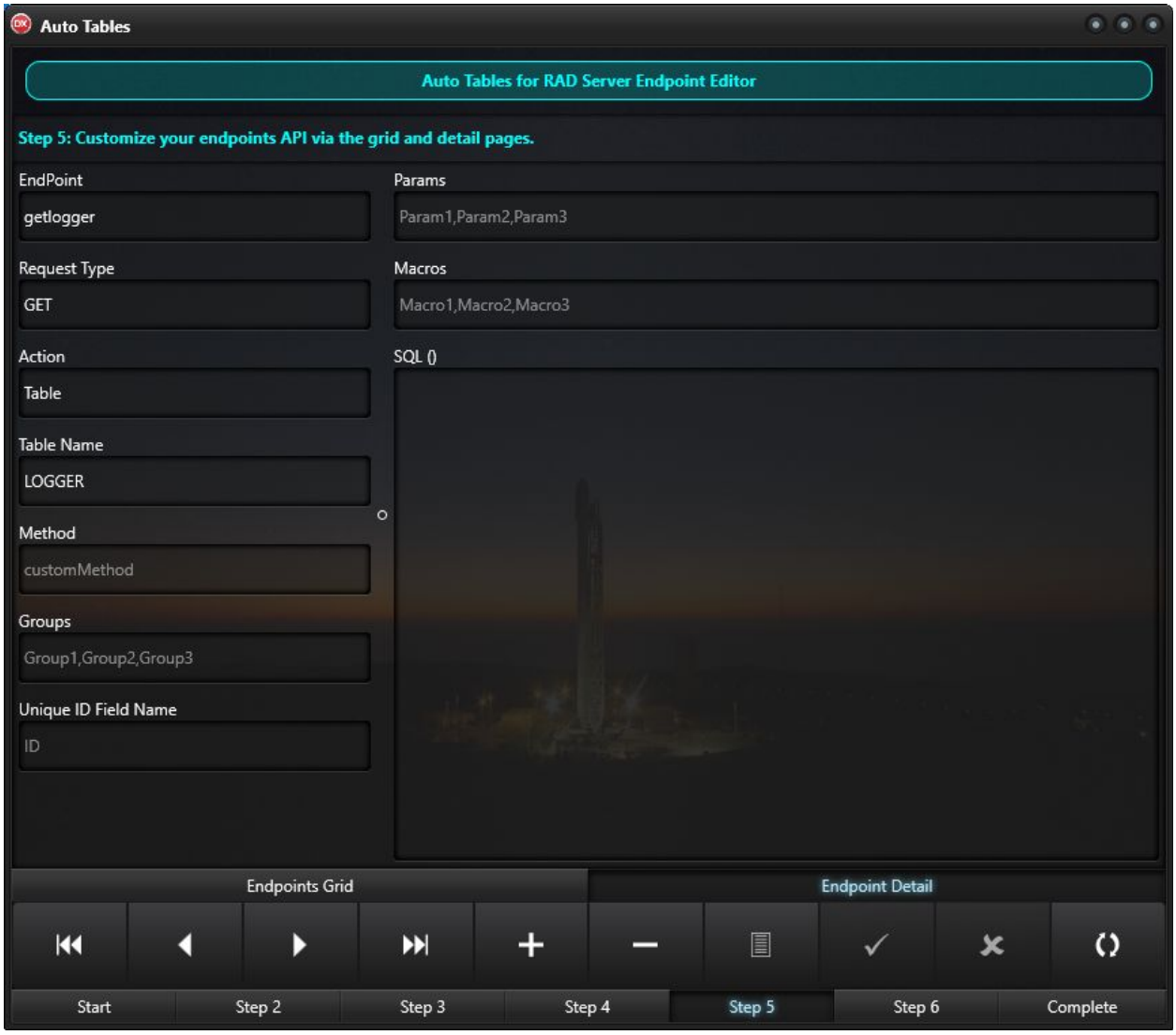
Finally, in the Groups section you can choose which Groups you want to have access to the REST API endpoints generated in this step. The Groups are populated in Step 2 of the editor.

All of the REST API endpoints can be configured after generation.

Step 5



In Step 5 you can add, edit, and delete REST API endpoints. If you auto generated endpoints in Step 4 you will see them here. You can manually edit the cells or you can select a row and then switch to the Endpoint Detail table for a full field editor of each REST API endpoint.

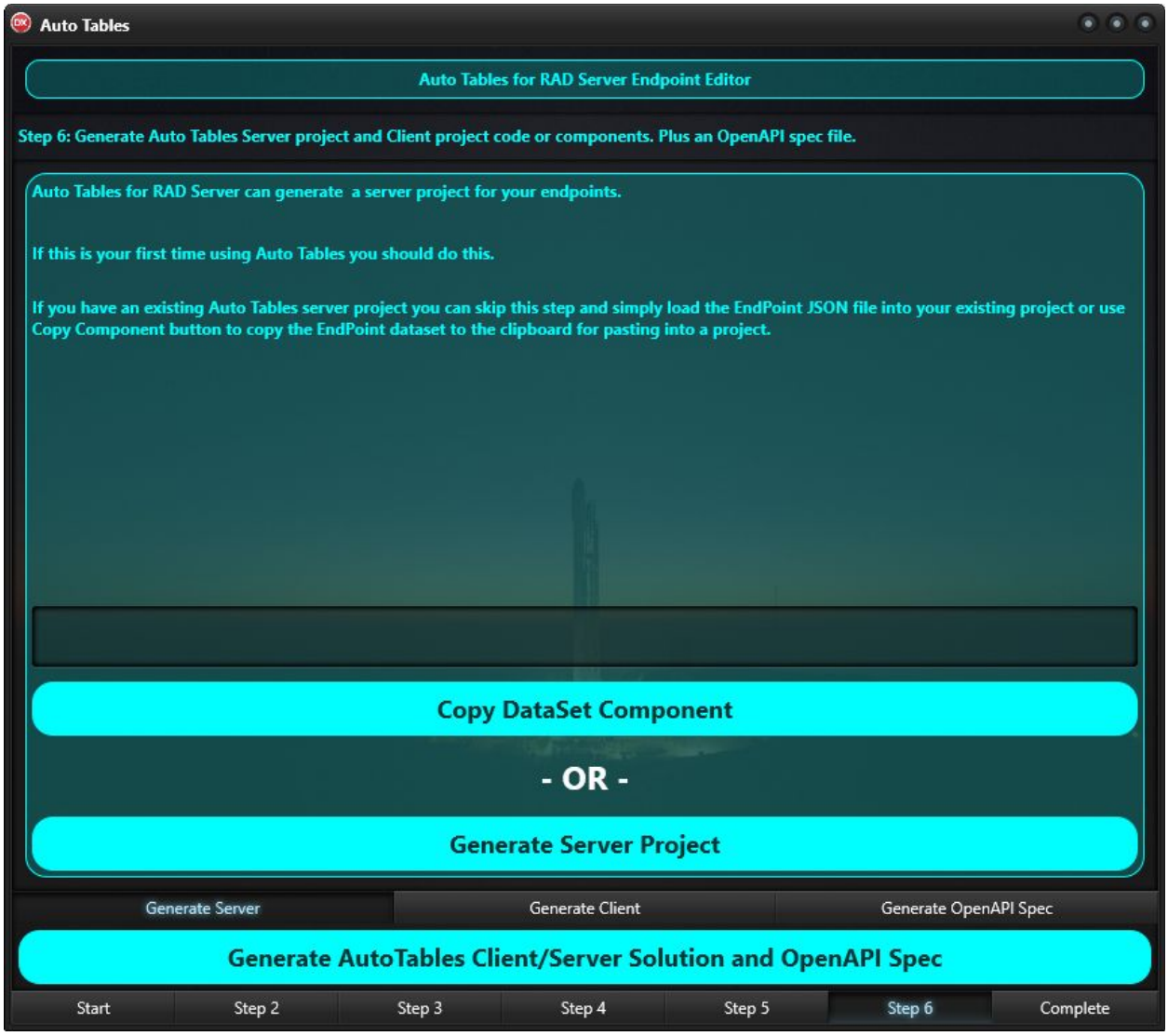


EndPoint is the name of the segment after the root segment that you want to use to call your endpoint. Request Type is the Operation that you want performed such as GET (read), POST (write), or DELETE (delete). Action is the type of Action you want to take place when the endpoint is called. Table Name is the name of the table you want the Action to be performed on. Custom Method is used for the name of the custom method in RAD Server if the Action is set to

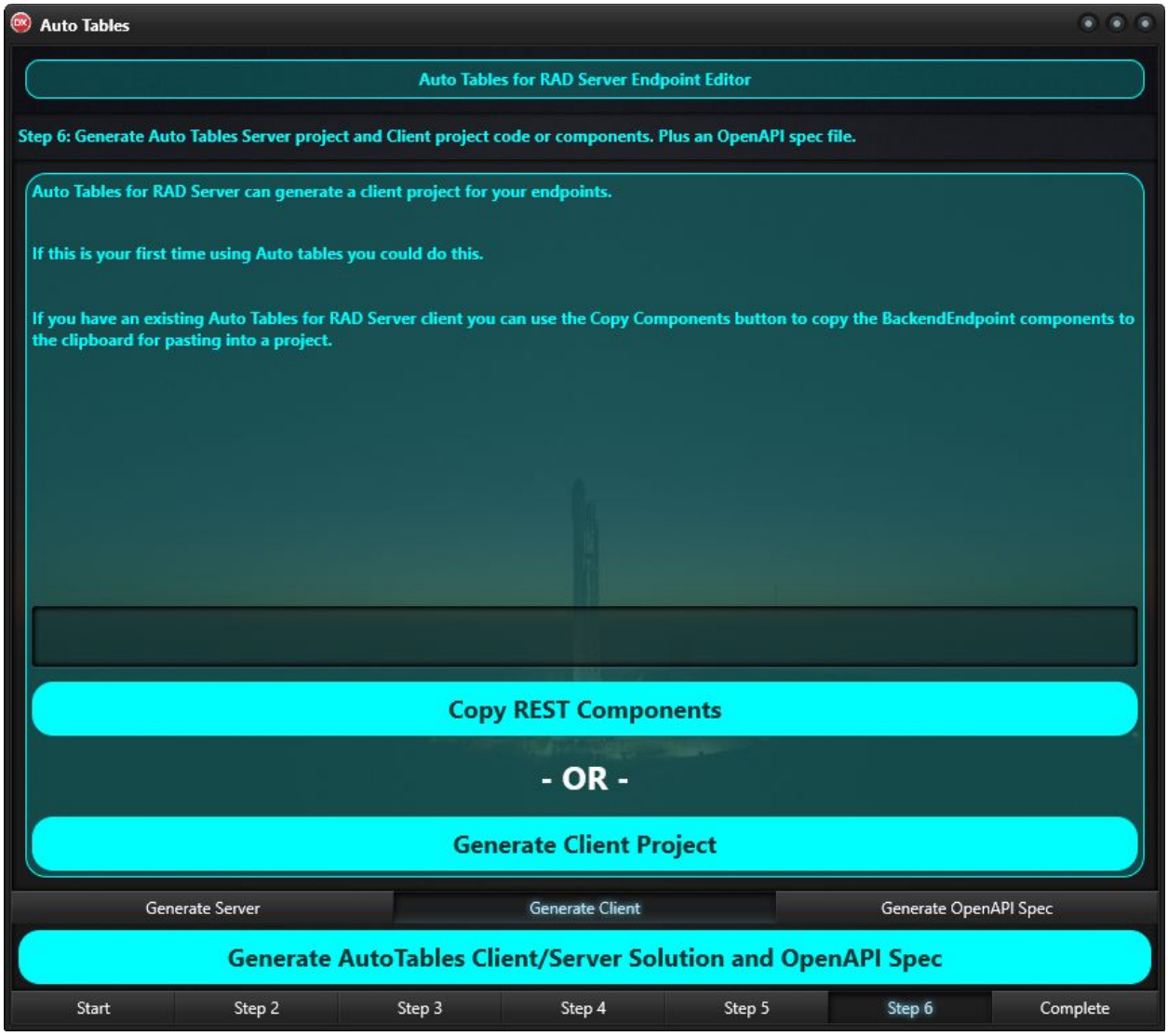
Method. Groups is a comma delimited list of groups you want to have access to the endpoint. Unique ID Field Name is the name of the ID field in your database for the table that the endpoint is operating on. Params is a comma delimited list of parameters that can be passed in on the query string to the endpoint. Params resolve to FireDAC params in the SQL query. Macros is a comma delimited list of macro names that can be passed in on the query string to the endpoint. Macros resolve to FireDAC macros in the SQL query. The SQL field is the custom SQL you want the endpoint to use if the Action is set to SQL or AggregateSQL.

Step 6

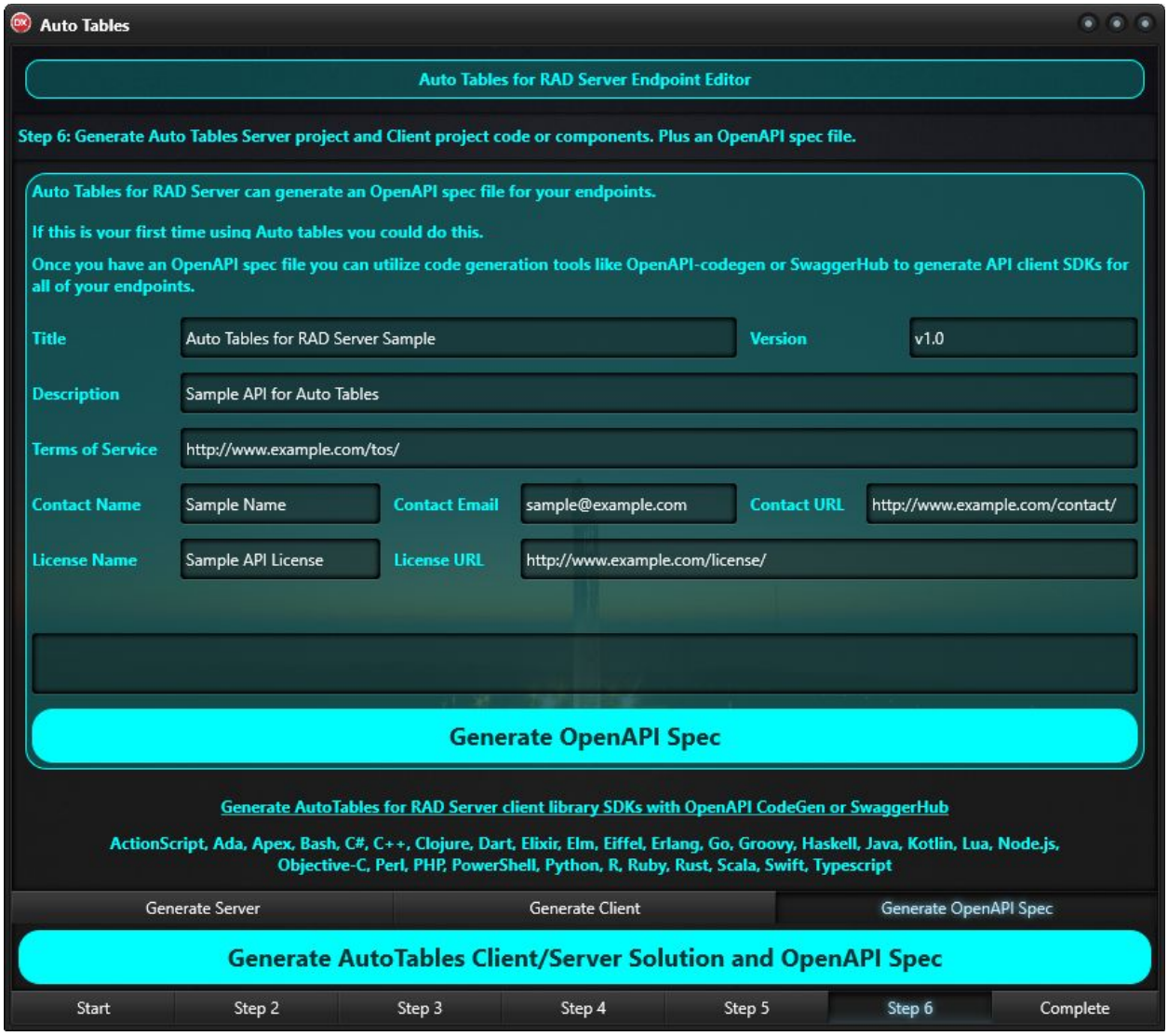
In Step 6 you can generate an entire solution for your AutoTables endpoints. The solution includes a RAD Server, a Delphi client library, a Delphi SDK, and an OpenAPI Spec file.



Clicking Generate Server Project will generate the RAD Server Delphi project for publishing the AutoTables endpoints for your project.



Clicking Generate Client Project will generate the Delphi client library and Delphi SDK for accessing the AutoTables endpoints for your project.

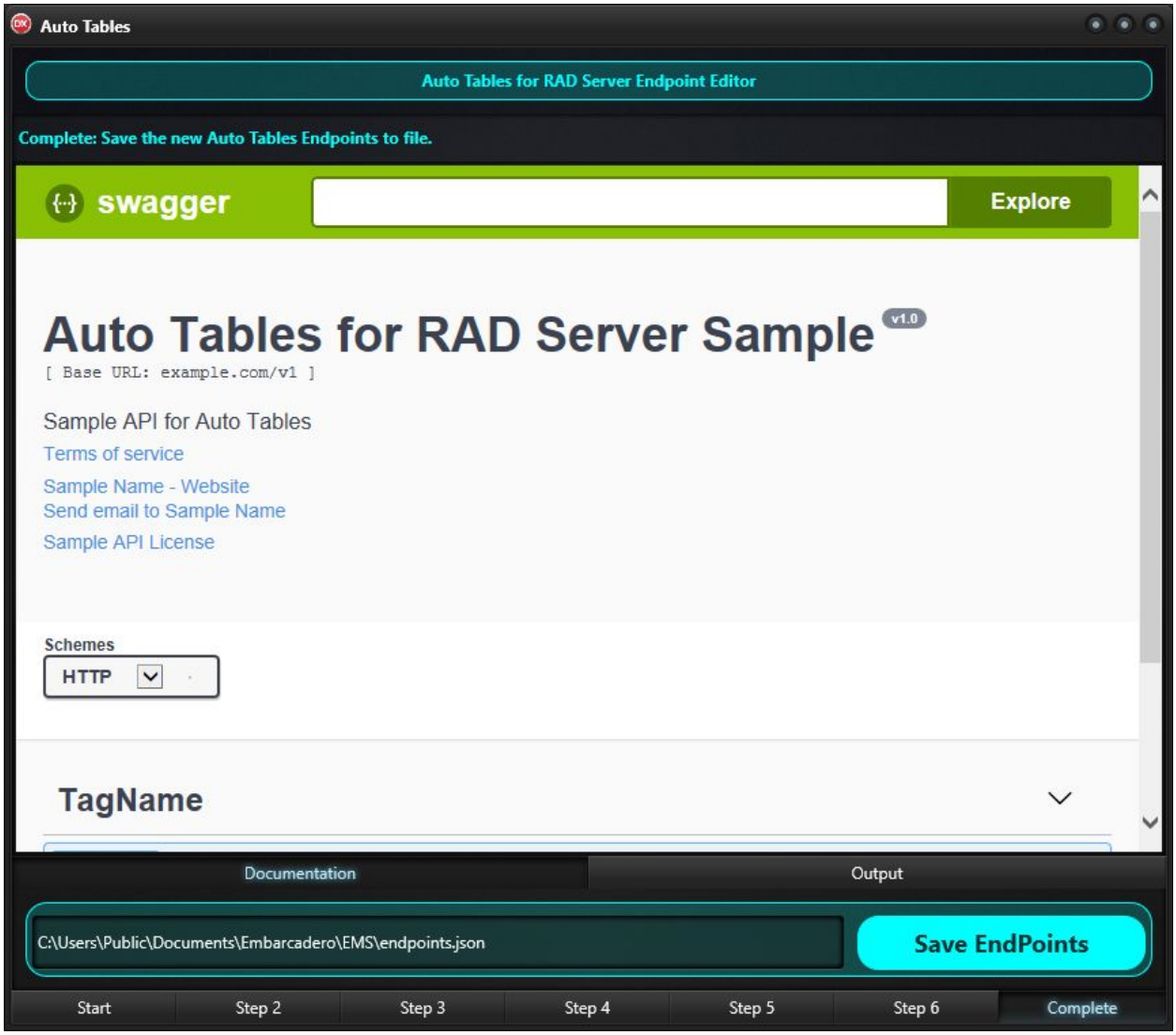


The OpenAPI spec generator has a number of different fields it needs filled out to generate a propert OpenAPI spec. These fields include Title, Version, Description, Terms of Service, Contact Name, Contact Email, Contact URL, License Name, and License URL. Clicking Generate OpenAPI

Spec will generate the JSON specification for your endpoints and open the folder where the file is saved.

Complete

In the Complete step on the Documentation tab you can see the Swagger-UI version of the documentation for your API.



On the Output tab you can see a list of the endpoints for your Auto Tables for RAD Server, you can copy the Client Components to clipboard for accessing those endpoints, and you can access the Custom Methods if any that your Auto Tables for RAD Server utilizes.

Lastly, if you used the editor to customize the endpoints for an existing Auto Tables for RAD Server installation you can save out the endpoints to a FireDAC JSON file for loading into your existing Auto Tables for RAD Server project.