```
<cfcomponent output="false" extends="taffy3_1_0.core.api">

  <cfset this.sessionManagement = true />
  <cfset this.sessionTimeout = createTimeSpan(0,3,0,0) />
  <cfset this.clientManagement = true />
  <cfset this.mappings[ "/taffy" ] = expandPath('./taffy3_1_0') />
  <cfset this.rootPath = getDirectoryFromPath(getCurrentTemplatePath()) />
  <cfset this.mappings[ "/common" ] = #this.rootPath.ReplaceFirst("([^\\\/]+[\\\/]){3}$", "")#
 & 'common'/>
  <cfset this.mappings[ "/resources" ] = expandPath('./resources') />
  <cfset this.mappings[ "/cfc" ] = expandPath('./cfc') />
  <cfset this.mappings[ "/cfm" ] =  expandPath('./cfm') />
  <cfset this.mappings[ "/root" ] = this.rootPath />
  <cfset this.serialization.preservecaseforstructkey = false >

  <cfset this.name = hash(getCurrentTemplatePath()) />
  <cfset this.clientManagement = true />
  <cfset this.serialization.serializequeryas = "struct">

<cffunction name="OnApplicationStart" access="public" output="false">
  <cfset application.rootPath = this.rootPath>
  <!--- Get common application variables --->
  <cfinvoke component="common.cfc.commonApplicationVars" method="setCommonApplicationVars">
  <!--- Get application variables --->
  <cfinvoke component="cfc.applicationVars" method="setApplicationVars">

  <cfinclude template="/common/cfm/onApplicationStart.cfm" >

  <cfinvoke component="cfc.applicationVars" method="setApplicationTables">

  <cfscript>
    application.objArrayCollection = createObject("component","common.cfc.ArrayCollection");
    application.objAdAuth = createObject("component","common.cfc.authenticateAd");

    //application.jsonUtil = createObject("component", "common.cfc.jsonUtil");
    var local = {};
    application.serializer = new common.cfc.JsonSerializer()
          .asInteger( "ID" )
          .asString("FNAME")
          .asString("MNAME")
          .asString("LNAME")

        ;

    return super.onApplicationStart();
  </cfscript>
</cffunction>

<cffunction name="OnSessionStart" access="public" output="false" returntype="void">

</cffunction>


<cffunction name="OnRequestStart" access="public" output="false">
  <cfif isdefined("url.reload")>
    <cfinvoke method="OnApplicationEnd">
    <cfinvoke method="OnSessionEnd">
```

```
    <cfinvoke method="OnApplicationStart">
    <cfinvoke method="OnSessionStart">
  </cfif>
  <cfscript>
    variables.framework = {
      serializer = "testSerializer",
      reloadKey = "reload",
      reloadPassword = "true",
      reloadOnEveryRequest = false,
      allowCrossDomain = "https://thecompany.sharepoint.com,https://formso365.nintex.com ",
      exceptionLogAdapterConfig = {
        emailFrom = "api-error@test.com",
        emailTo = application.errorEmail,
        emailSubj = "Exception Trapped in " &application.displayName & ' - ' &
application.environment,
        emailType = "html"
      },
      disableDashboard = false,
      environments = {
        PROD = {
            disableDashboard = false,
            reloadOnEveryRequest = false
        }
      }
      //,serializer = "JsonUtilSerializer"
    };

    return super.onRequestStart();
  </cfscript>
</cffunction>

<cffunction name="OnRequestEnd" access="public" output="true" >
  <cfif isdefined("url.debug")><cfinclude template="/common/cfm/dumpVars.cfm"></cfif>
</cffunction>

<cffunction name="OnSessionEnd">
  <cfset structClear(session)>
</cffunction>

<cffunction name="OnApplicationEnd">
  <cfset structClear(application)>
</cffunction>

<cffunction name="getEnvironment" returntype="String">
  <cfreturn #application.environment#>
</cffunction>

<cfscript>
  // this function is called after the request has been parsed and all request details are
known
  function onTaffyRequest(verb, cfc, requestArguments, mimeExt){
    var local = {};
    local.arrError = [];

    local.auth = getBasicAuthCredentials();
    if (len(trim(local.auth.username)) && len(trim(local.auth.password))){
      if (isdefined("session.#local.auth.username#_#application.hlq#.username") &&
```

```
    session["#local.auth.username#_#application.hlq#.username"] eq local.auth.username){
112         //do nothing user is authed
113       } else {
114         local.user =
    application.objAdAuth.authLogin(user=local.auth.username,pass=local.auth.password);
115         if (local.user.found){
116           session["#local.auth.username#_#application.hlq#.username"] = local.auth.username;
117           session["#local.auth.username#_#application.hlq#.userid"] = local.user;
118           session["#local.auth.username#_#application.hlq#.roles"] = '';
119           if (isdefined("application.#local.auth.username#_roles")){
120             session["#local.auth.username#_#application.hlq#.roles"] =
    application["#local.auth.username#_roles"];
121           }
122         } else {
123           return rep("[Not Authorized - User/Pass not found]").withStatus(401, "Not Authorized
    - User/Pass not found");
124         }
125       }
126     } else if (arguments.cfc eq 'userRoles') {
127         return true;
128     } else {
129       //unauthorized because they haven't included their username/pass
130       //arrayappend(local.arrError, 'Not Authorized - User/Pass not supplied ');
131       //return rep(local.arrError);
132     }
133
134     if (isdefined("application.#arguments.cfc#_#arguments.verb#_actionRoles") &&
    len(application["#arguments.cfc#_#arguments.verb#_actionRoles"])){
135       local.authorized = false;
136       for (local.i=1; local.i LTE
    listLen(application["#arguments.cfc#_#arguments.verb#_actionRoles"]);local.i=local.i+1) {
137         if
    (listfindnocase(application["#arguments.cfc#_#arguments.verb#_actionRoles"],listGetAt(applicat
    ion["#arguments.cfc#_#arguments.verb#_actionRoles"],local.i))){
138           local.authorized = true;
139         }
140       }
141       if (not local.authorized){
142         return rep("[Not Authorized to method]").withStatus(401, "Not Authorized to method");
143       }
144     }
145
146     /* savecontent variable="local.result" {
147
148             writeDump( local );
149             writeDump( session );
150             writeDump( arguments );
151             writeDump( application );
152
153         };
154     sendEmail(body=local.result); */
155
156     //if(not structKeyExists(arguments.requestArguments, "apiKey")){
157       //unauthorized because they haven't included their API key
158       //return noData().withStatus(401);
159     //}
160
```

```
161
162     //api key found
163     return true;
164   }
165 </cfscript>
166 </cfcomponent>
```