Developer Guide: REST API Applications

Sybase Unwired Platform 2.2 SP04

DOCUMENT ID: DC01858-01-0224-01
LAST REVISED: May 2013

Contents

REST Services Applications

The Sybase® Unwired Platform REST Services API enables standard HTTP client applications running in any platform to access Sybase Unwired Platform REST services.

Unwired Server provides REST services so that applications can be built as any standard HTTP application to leverage Unwired Server for security and push features among others.

The client application should first register an application connection with device information, such as device type, push information and password capability and so on. After registering, application can retrieve and update the application connection settings through REST API. You can enable or disable the push notification only after registering.

Note: The application connection can be deleted through REST API, when it is not in use. In this case, the data stored by application in the custom fields of the application connection will be lost.

The application can download resources (such as meta-data files, multimedia files, and so on.) through resource bundles service, in order to initialize client application.

Then, the application can access SAP® Gateway through proxy service, and receive native push notification triggered by gateway if push properties are configured and enabled. If application supports can also upload the business transactions for tracing using BTX service.

This development approach supports:

- Registration (creating an application connection)
- Authentication
- Native push notification
- Configuration

You build your client applications using third party developer tools, and without using any Sybase Mobile SDK or native client libraries. Sybase Unwired Platform provides management and monitoring of the applications, and support for native push notification such as Apple Push Notification Service, BlackBerry Enterprise Server push, or Google Cloud Messaging.

Optionally, you may use certain Sybase Unwired Platform native client libraries to enable complex application capabilities, such as OData parsing and building, caching, or data vault management. For more information, see *Developer Guide: OData SDK*. You may also use third-party JavaScript framework and helper libraries.

REST SDK Library

(Applicable to iOS and Android platforms) The REST SDK libraries enable consumption of Sybase Unwired Platform REST services with pure HTTPS connectivity. The REST SDK provides simplified APIs for registration, exchange settings between client and server, and end-to-end tracing. The SDK also supports native push notifications.

The different components of the REST SDK are implemented as static runtime libraries and each component can be used independently.

The REST SDK supports the following functionalities:

- Initialize client connection
- User on-boarding or registration
- Setting exchange such as setting configurations on SCC
- Customization resource bundle
- Security configurations - Basic authentication, SSO certificate authentication, anonymous access, network edge (SiteMinder authentication - always asynchronous), non-network edge, single SSL and mutual SSL authentication
- End-to-end tracing
- Native notification support for iOS and Android platforms
- CAPTCHA support (asynchronous only)
- Non network edge authentication support for iOS and Android platforms

Feature differences between On-Premise and Cloud versions

Differences between On-Premise and Cloud functionality
The table provides the list of features supported in On-Premise and/or Cloud versions.

Feature Support	On-Premise	Cloud
Notification of Customization resource bundle	No	Yes
Registration is required	No	Yes
CAPTCHA support (asynchronous)	No	Yes
BTX upload	Yes	No
Resource bundles configuration at application connection level	Yes (only at application connection level)	Yes
Mutual SSL (between client and Sybase Unwired Platform server)	Yes	No
Network edge authentication	Yes	No
HTTP **Note:** The default connectivity is HTTPS.	Yes	No

Feature Support	On-Premise	Cloud
End-to-end tracing	Yes	No

Documentation Roadmap for Sybase Unwired Platform

Sybase® Unwired Platform documents are available for administrative and mobile development user roles. Some administrative documents are also used in the development and test environment; some documents are used by all users.

See *Documentation Roadmap* in *Fundamentals* for document descriptions by user role.

Check the Product Documentation Web site regularly for updates: *http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&amp;c=firsttab&amp;a=0&amp;p=categories*, then navigate to the most current version.

Prerequisites

Review the prerequisites for developing an HTTP client application to access Sybase Unwired Platform REST services.

Application Requirements

Ensure that application meets the necessary requirements for authentication, user names, cookies, and client libraries.

Authentication

Set up your HTTP client to use one of four types of authentication.

- **Basic authentication –** Provide a username and password to login. This method is available when connecting through HTTP and one-way HTTPS.
- **SSO token –** Provide an SSO token to login. This method is available when connecting through HTTP and HTTPS and a token validation service is available and configured.
- **X.509 Mutual authentication –** Provide a client certificate to login. This method is available only through HTTPS port configured with enabling mutual authentication. For more information on mutual authentication, see *Single Sign-on for SAP* in *Security*.
- **X.509 Mutual authentication through intermediary –** Provide a forwarded client certificate to login; the forwarded certificate is provided by the SSL_CLIENT_CERT header. The SSL_CLIENT_CERT header contains a forwarded PEN-encoded client certificate. This method is available only through an appropriately configured HTTPS listener. The certificate forwarder must have the "SUP Impersonator" role to be authorized for this type of login. The certificate of the actual "SUP Impersonator" user cannot be used as a regular user certificate.

  Intermediary refers to the relay server or an apache reverse proxy. The requests are sent on mutual SSL connection from client to intermediary and then intermediary to SUP. From client to intermediary, client certificate is used on transport level. From intermediary to SUP, the impersonator certificate is used on transport level. The client certificate is forwarded as SSL_CLIENT_CERT HTTP request header.

A request targeting a resource that requires authentication fails with error code 401 unless X. 509 mutual authentication is used to establish a valid authenticated request context (The certificate validation login module must exist in the security configuration used to authenticate the request and the certificate validation must be successful and sufficient. If that is not the case, regular username/password or token authentication is attempted and the request is authenticated equal to a non X.509 authenticated connection). After three subsequent failed authentication attempts using the same X-SUP-SESSID, the fourth attempt fails with a 403 error code.

Authentication Challenges

Your application must be able to respond to standard HTTP authentication challenges.

Supported Username Variations

The following username variations are supported.

- **user –** A regular username is used.
- **user@example.com –** An email address is used as the username. For example, user@foo.bar.
- **user@sc –** A regular username is used and the security configuration is overridden by a username suffix.
- **user@example.com@sc –** An email address is used as the username and the security configuration is overridden by a username suffix. For example, user@foo.bar@sc.

For those usernames that override the security configuration, both the default security configuration associated with the domain as well as the security configuration specified in the X-SUP-SC request header are overridden. You must associate the specified security configuration with the domain to allow a successful connection, otherwise a 403 error is returned to the HTTP client. If a request refers to a non- existent domain or security configuration, a 404 error is returned. If you use mutual X.509 authentication with certificates, you can only override the domain default security configuration using request headers. If the specified security configuration does not exist, a 404 error is returned.

HTTP Cookies

Ensure that you enable HTTP cookies in your application. See RFC 6265 for information on standard HTTP cookie processing.

Relay Server Pass-Through Mode for HTTP Clients

HTTP clients can connect through a Relay Server to Unwired Server to use the REST services.

Reverse Proxies

Requirements for third party reverse proxies for use with Sybase Unwired Platform.

The reverse proxy must be a straight pass-though proxy server. Ensure the reverse proxy meets these requirements:

- Does not change the content encoding of the requests or responses. Chunked transfer encoding is the required data transfer mechanism. Content-length encoding is not supported.
- Does not remove any HTTP headers.
- The timeout period (if any) must be greater than the timeout used by the clients.
- The resulting URL that is passed to the Sybase Unwired Platform must be `http://HostName/[public/]<AppID>`, where [public] is optional and if included anonymous connection is allowed. Example, `http://suphost/public/empDir`.

Using Reverse Proxy for REST application

The REST application needs to communicate to the web service port. The default HTTP port which can be used for communication is 8000. The configuration policies are:

1. Map the root context of `http://reverseProxy:8000` to `http://supServer:8000`. The REST application sends a request directly to Unwired Server server. For example, `http://reverseProxy:8000/odata/applications/v1/empDir` and `http://reverproxy:8000/public/empDir`.
2. Map the "/smp/rest" context of `http://reverseProxy:8080` to `http://supServer:8000`. The REST application needs to add the "/smp/rest" context in request URL. For example, `http://reverseProxy:8080/smp/rest/odata/applications/v1/empDir` and `http://reverproxy:8080/smp/rest/public/empDir`.

Client Libraries

Optionally, you can include certain Sybase Unwired Platform or OData client libraries from the Sybase Mobile SDK to enable complex application capabilities, such as OData parsing and building, or data vault management.

You can include the following libraries if needed by your application:

- Datavault (for secure storage of sensitive data, or password policy capabilities)
- iOS, Android, or BlackBerry clients can use certain OData libraries:
  - OData parser (SDM parsing libraries)
  - OData Persistence and Cache
- iOS and Android clients can use REST SDK library.

Do not include the following libraries:

- Sybase Unwired Platform proxy libraries from the OData SDK
- Other native Sybase Unwired Platform libraries

Configuring Unwired Server

Configure access for authenticated or anonymous HTTP clients to connect to Unwired Server through a proxy connection.

Note: The administrator must configure Unwired Server before you develop HTTP client application.

Creating a Security Configuration for a Domain

Define a set of security providers in Sybase Control Center to protect domain resources, according to the specific security requirements of the domain. Create a security configuration, then map it to the desired domain.

A security configuration determines the scope of data access and security. A user must be part of the security repository used by the configured security providers to access any resources (that is, either a Sybase Control Center administration feature or a data set from a back-end data source) on a domain. See *Security Configurations* in the *Security* guide.

When you create a new security configuration, Sybase Unwired Platform sets the NoSecurity provider by default. Sybase recommends that after you add, configure, and validate your providers, you remove the NoSecurity provider. For more information on the NoSecurity provider, see *NoSecurity Configuration Properties* in*Sybase Control Center for Sybase Unwired Platform*.

1. In Sybase Control Center, add a new security configuration using the **Security** node.
2. In the left navigation pane, expand the **Security** folder and select the new configuration.
3. Use the **Authentication**, **Authorization**, **Attribution**, and **Audit** tabs to configure the appropriate security providers for each aspect of domain security.
4. Edit the security provider properties, as required.
5. Validate the configuration to ensure that Unwired Server accepts the changes.
6. Go to left navigation pane, select **Server** -> **Domains** -> **default** -> **Security** folder.
7. Assign the security configuration to a particular domain.

   Note: If HTTP request with X-SUP-SC header is not sent, then the security configuration is set as default.
8. Apply the changes to Unwired Server.

Creating an Application

Create a new application in Unwired Server.

1. In the left navigation pane of Sybase Control Center, click the **Applications** node and select the Applications tab in the right administration pane.

2. Click **New** to register an application.
3. In the Application Creation Wizard, enter a unique **Application ID**.
4. Enter a **Display name** and **Description** for the application.
5. Select the appropriate security configuration from the **Security configuration** drop-down list.
6. (Optional) For applications that do not require authentication, select the anonymous security configuration or the **Anonymous access** checkbox.
7. Select the appropriate domain from the **Domain** drop-down list.

   If you do not select the specific security configuration and domain from the list, the default values will be considered.
8. Click **Finish** to register the application with the configured settings.

Creating a Proxy Connection (Whitelisting)

Create a new connection in Sybase Control Center to allow a proxy connection (authenticated or anonymous) through Sybase Unwired Platform.

Note: When you set the proxy property with the endpoint address in the application template (either as part of the application creation or editing the application template created for that application), a proxy connection is generated automatically.

1. In the left navigation pane, expand the Domains folder, and select the default domain.
2. Select **Connections**.
3. In the right administration pane, select the **Connections** tab, and click **New**.
4. Enter a unique Connection pool name.

   The Connection pool must have the same name as the application ID.
5. Select the Proxy **Connection pool** type.
6. Select the appropriate template for the data source target from the **Use template** menu.
7. Set the **Address** property by clicking the corresponding cell and entering the address of the proxy endpoint. For example, `http://odata.example.com/v2/Catalog/`
8. Configure the following optional properties:

   Note: To access an external service, you must configure the `http.proxyHost` and `http.proxyPort` properties during server configuration in **Sybase Control Center > Servers > Server Configuration > General > User Options**. If you set or change the setting for `http.proxyHost`and `http.proxyPort`, you must restart the services using the stop/start service scripts. For more information, see *Administer > Unwired Server > Configuring Unwired Server to Securely Communicate With an HTTP Proxy* in *Sybase Control Center for Sybase Unwired Platform* .

   - **Allow Anonymous Access –** Enables anonymous authentication to Sybase Unwired Platform.

- **Certificate Alias –** Client SSL certificate stored in the Sybase Unwired Platform keystore to be forwarded to the EIS.
- **Username –** Username passed to the EIS.
- **Password –** Password passed to the EIS.
- **EnableHttpProxy –** Enables Internet proxy support in the proxy connector. EnableHttpProxy defaults to false. Set explicitly to `true` to enable.
- **EnableUrlRewrite –** Enables the proxy component to rewrite URLs (or URIs) embedded in a response, with SUP URLs that causes the client requests to be directed back to the Unwired Server (proxy component), rather than to the back end server. EnableUrlRewrite defaults to true. Set explicitly to `false` to disable.
- **ProxyHost –** Host name of the proxy server.
- **ProxyPort –** Port number.

Note:

- On a proxy connection, if the header for X-SUP-BACKEND-URL is not NULL, or `EnableURLRewrite` is false then no URL rewrite occurs for either the request or response content.
- To access the external services, you must configure `EnableHttpProxy` = True, `ProxyHost` = proxy, `ProxyPort` = 8080 in the connection pool.
- In REST services, the proxy URL is fetched from the application ID which is sent from the client device. The same application name is also present in the connection pool. This proxy URL is used for request/response.

9. Click **OK** to register the connection pool.

Managing Applications Connections

Manage application connections for authenticated users by creating a application template based on the template that was automatically created when you added the application.

Creating Application Connection Templates

Create application connection templates by setting appropriate properties and values.

1. In the left navigation pane, click the **Applications** node.
2. In the right administration pane, click the **Application Connection Templates** tab.
3. Click **New**.
4. Enter the **Template name** and **Description** for the application connection template.
5. Select the **Base template** from the drop-down list.
6. You can configure any of the following profiles. See *Application Connection Properties*:
    - Application Settings
    - Android Push Notifications
    - Apple Push Notifications

- BlackBerry Push Notifications
- Capabilities
- Connection
- Custom Settings
- Device Advanced
- Device Info
- Password Policy
- Proxy
- Security Settings
- User Registration

Note: To change the password policy, the **CapabilitiesPasswordPolicy** property should be set to true.

7. Click **OK**.

Assigning the Security Configurations

Assign the anonymous and authenticated security configuration to the domain.

1. In the left navigation pane, navigate to ***ClusterName*** **> Domains >** ***DomainName*** **> Security**.
2. In the right administration pane, select the **Security Configurations** tab and click **Assign**.
3. From the list of available security configurations, select the appropriate configuration for domain security, and click **OK**.

   To allow HTTP clients anonymous access, select the anonymous security configuration, and to allow authenticated access, select one or more security configurations that you created.

   If successful, an Assigned successfully message appears, and the newly added security configuration is listed in the domain-level Security node.

Developing the Application

Develop your HTTP client application to use the REST Services API to access Sybase Unwired Platform REST services.

Request URL Format

The HTTP client can access Sybase Unwired Platform services using a specific URL format.

For accessing services directly through Unwired Server without a Relay Server:

```
http://host:port/[public/]/{resource}/{resource-specific}
```

Host is defined by host name and should match with the domain registered with Sybase Unwired Platform. If the requested domain name does not match, default domain will be used.

The `resource` represents one of the built-in Sybase Unwired Platform services for managing application registration, settings, or customization resource bundles, or one of the configured Proxy Connection endpoints.

The `resource-specific` part of the URL specifies the actual resource being referenced.

For accessing services where a Relay Server is configured with the Unwired Server, the URL you use depends on whether Relay Server is installed on an IIS or Apache host.

For an IIS host:

```
http://{RelayServerName}:80/ias_relay_server/client/rs_client.dll/
{Unwired Server Farm ID}/[public/]/{ProxyName}
```

For an Apache host:

```
http://{RelayServerName}:80/cli/iasrelayserver/{Unwired Server Farm
ID}/[public/]/{ProxyName}
```

Requests to Sybase Unwired Platform require valid transport credentials unless made over a Sybase Unwired Platform protocol (Messaging-based synchronization and its Java implementation) or they explicitly request a public resource (indicated when a resource in the "public" context is requested). Each Sybase Unwired Platform domain can authenticate requests by using a default security configuration. If a domain has no default security configuration defined, the default security configuration, "admin," is used to authenticate the request.

By explicitly setting the HTTP headers, you can override the domain used to resolve the targeted endpoint, as well as the security configuration used to authenticate a request. For more information, see *HTTP Headers and Cookies* on page 25.

The resource part of the URL corresponds to a proxy endpoint name (connection type PROXY). The Sybase Unwired Platform Proxy component handles any request made towards a configured Proxy Connection Endpoint and later forwards the request to the URL specified in the connection template.

Example URL

Example of an HTTP URL request.

```
http://mobile.sap.com/com.sap.mobile.endpointABC/mybackendurl
```

The "mobile.sap.com" part of the URL refers to the Sybase Unwired Platform domain name, and "com.sap.mobile.endpointABC" is the connection name as defined within the scope of the Sybase Unwired Platform domain "mobile.sap.com".

In Sybase Unwired Platform, there must be a proxy endpoint named "com.sap.mobile.endpointABC" within the "mobile.sap.com" or "default" domains. If the connection "com.sap.mobile.endpointABC" is undefined, the device receives a 404 error.

Creating an Application Connection

You must explicitly register the application connection using the Sybase Unwired Platform. You can specify customized application properties for the client with the request. You should provide the application connection ID, X-SUP-APPCID, using an explicit request header or a cookie. If the value is missing, the Unwired Server generates a Universally Unique ID (UUID), which is communicated to the device through response cookie X-SUP-APPCID.

Note: OData proxy client must support cookies for registered users. During registration, the server sends the X-SUP-APPCID in a cookie. The client sends the cookie back in subsequent requests or set the X-SUP-APPCID in a header.

Create an anonymous or authenticated application connection by issuing a POST request to the following URL, and include the application connection properties in the request.

```
http://host:port/[public/]/odata/applications/{v1|latest}/{appid}/
Connections
```

The URL contains these components:

- **host –** host is defined by host name and should match with the domain registered with Sybase Unwired Platform. If the requested domain name does not match, default domain will be used.
- **port –** The port for listening for OData requests; by default, 8000.
- **“public” prefix –** If included, an anonymous security configuration is used.
- **odata/applications/ –** Refers to the OData services associated to the application resources.
- **{V1|latest} –** The version of the service document.

- **appid –** The name of the application.
- **Connections –** The name of the OData collection.

Application connection properties are optional. You can create an application connection without including any application properties. For the usage and description of application connection properties, search for *Application Connection Properties* in *System Administration* .

`DeviceType` is an application connection property that you may typically set. Valid values for `DeviceType` are:

- **Android –** Android devices.
- **iPhone –** iOS devices.
- **Blackberry –** Includes devices with BlackBerry OS 5 and below.
- **Rim6 –** Includes devices with BlackBerry OS6 and above.
- **WinMobile –** Includes devices with Windows Mobile 5 for Pocket PC and Windows Mobile 6 Professional.
- **WinSmartPhone –** Includes devices with Windows Mobile 5 for Smart Phone and Windows Mobile 6 Standard.
- **Windows –** Includes desktop or servers with Windows OS, such as Windows XP, Windows Vista, Windows 7, Windows Server series OS.

Specifying any other value for `DeviceType` causes "Unknown" to be returned in the `DeviceType` column.

For following example shows the HTTP request body to register an application connection using default values for all properties.

```
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://
schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
    <content type="application/xml">
        <m:properties/>
    </content>
</entry>
```

Anonymous Application Connections

For an anonymous application connection, an X-SUP-SECTOKEN cookie is generated when you create the application connection, and you must include the X-SUP-SECTOKEN in request header and cookie during subsequent read or write requests for the application settings. If you do not provide an X-SUP-SECTOKEN in the request for the anonymous application settings, you will receive a 403 error.

Getting Application Settings

You can retrieve the application connection settings for the device application instance by issuing the GET method.

You can retrieve application settings by either explicitly specifying the application connection ID, or by having the application connection ID determined from the call context (that is, from either the X-SUP-APPCID cookie or X-SUP-APPCID HTTP header, if supplied). On the first call you can simplify your client application code by having the application connection ID determined from the call context, since you have not yet received an application connection ID.

If you supply an application connection ID, perform an HTTP GET method at the following URL:

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Connections('{appcid}')
```

If you do not supply an application connection ID, perform an HTTP GET method at the following URL:

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Connections('*current ')
```

You can also retrieve a property value by appending the property name in the URL. For example, to retrieve the `ClientLogLevel` property value:

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Connections('{appcid}')/ClientLogLevel
```

Getting Changed Settings

You can conditionally retrieve only the changed settings.

```
http://host:port/[public/]odata/applications/{v1|latest}/
Connections('{appcid}')?If-None-Match="${ETag}"
```

The `${ETag}`part of the URL is a version identifier included in the response of the GET method. If the ETag value of the current application settings is the same as the value included in the request, a status code 304 without a response body is returned to the client to indicate there are no application settings changes.

Note: The query function is not supported.

Registering the Client for Native Push Notification

Enable native push notifications and register your application to receive push notifications.

Registering the Android Client

Register and enable your Android device clients to receive push notifications.

1. Obtain a GCM Sender ID. See instructions in Google's documentation for "Registering for GCM."
2. In Sybase Control Center configure push notification:
   a. In the Android Push Notifications section of the corresponding Application Template use the GCM Sender ID value to update the Sender ID, and set Enabled to true.
   b. Select Application > Properties > Push Configurations > Add to configure push settings in the application properties (setting the GCM server URL, and other parameters).
      - Use the API key from GCM to configure the API key.
3. Perform a one-time registration to enable push notifications for the application:
   a. Create an application connection. For example, for Android devices:

```
POST http://{domain-host-port}/odata/applications/v1/{appid}/
Connections
HTTP Header "Content-Type" = "application/atom+xml"
Body:
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://
schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/
dataservices">
  <content type="application/xml">
    <m:properties>
      <d:DeviceModel>Samsung Galaxy S2</d:DeviceModel>
      <d:DeviceType>Android</d:DeviceType>
      <d:DeviceSubType>Smartphone</d:DeviceSubType>
      <d:DevicePhoneNumber>555-555-1212</d:DevicePhoneNumber>
      <d:DeviceIMSI>123456</d:DeviceIMSI>
    </m:properties>
  </content>
</entry>
```

   Note: You can check the "AndroidGcmPushEnabled" value returned from the registration and only continue with the rest of the GCM/notification registration processing if it is "true".

   b. The application registers with the GCM server, using the value from the "AndroidGcmSenderId" property, as well as the Android Application ID (which you build into the application).
      A successful GCM registration results in a Registration ID back from Google.
   c. Using the ApplicationConnectionId ({appcid}) returned from the Sybase Unwired Platform registration call (in either the "X-SUP-APPCID" HTTP header or the "ApplicationConnectionId" property), the application should update the AndroidGcmRegistrationId property for the application connection for your application by using the GCM Registration ID returned by Google:

```
PUT http://{domain-host-port}/odata/applications/v1/{appid}/
Connections/('{appcid}')
```

```
HTTP Headers "Content-Type" = "application/atom+xml" and "X-
HTTP-METHOD" = "MERGE"
Body:
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://
schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/
dataservices">
  <content type="application/xml">
    <m:properties>
      <d:AndroidGcmRegistrationId>{GCM registration ID}</
d:AndroidGcmRegistrationId>
    </m:properties>
  </content>
</entry>
```

4. Native mobile listener should be implemented by the application to get the notification.

Registering the BlackBerry Client

Register and enable your BlackBerry device clients to receive push notifications.

1. In Sybase Control Center configure push notification:
   a. Select **Application** > **Properties** > **Push Configurations** > **Add**.
   b. Select BES as the Type, and enter the Name, BES Server URL and Suffix, Username, and Password.
2. Enable push notifications in the application itself:
   a. Create an application connection, and include the DeviceType properties.

```
POST http://{domain-host-port}/odata/applications/v1/{appid}/
Connections
HTTP Header "Content-Type" = "application/atom+xml"
Body:
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://
schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/
dataservices">
  <content type="application/xml">
    <m:properties>
      <d:DeviceModel>Storm</d:DeviceModel>
      <d:DeviceType>BlackBerry</d:DeviceType>
      <d:DeviceSubType>Smartphone</d:DeviceSubType>
      <d:DevicePhoneNumber>555-555-1212</d:DevicePhoneNumber>
      <d:DeviceIMSI>123456</d:DeviceIMSI>
    </m:properties>
  </content>
</entry>
```

   b. Implement the push notification listener interface in your application. For more information on implementing the push notification listener interface, see *Client Object API Usage > Push Synchronization Applications* in *Developer Guide: BlackBerry Object API Applications*.

Registering the iOS Client

Register and enable your iOS device clients to receive push notifications.

1. In Sybase Control Center configure push notification:
   a. Select **Application** > **Applications** > select the application > **Properties**.
   b. In the Properties window, select **Push Configuration** > **Add**.
   c. In the Push Configurations window, enter the Name, Domain, Type (APNS), Server (APNS Server), Port, Feedback server, and Feedback port.
   d. Browse and upload an Apple Push Notification Service p12 certificate, and enter the certificate password.
   e. Click **OK** to save.
2. Enable push notifications in the application itself:
   a. Call the `registerForRemoteNotificationTypes:` method.
   b. In your delegate, implement the `application:(UIApplication *)app didRegisterForRemoteNotificationsWithDeviceToken:` and `application:(UIApplication *)app didRegisterReceiveRemoteNotification:` methods to receive the device token and notifications.
   c. Create an application connection, and include the ApnsDeviceToken, and DeviceType properties.

      ApnsDeviceToken can be sent to the server using `setConfigurationProperty` method.

```
PUT http://{domain-host-port}/odata/applications/v1/{appid}/
Connections/('{appcid}')
HTTP Headers "Content-Type" = "application/atom+xml"
Body:
<?xml version='1.0' encoding='utf-8'?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://
schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/
dataservices" xml:base="http://host:port/odata/applications/
v1/com.example.IOS/">
  <id>http://host:port/odata/applications/v1/com.example.IOS/
Connections('32552613-470f-45e0-8acc-b7d73d501682')</id>
  <link rel="edit" title="Connection"
href="Connections('32552613-470f-45e0-8acc-b7d73d501682')" />
  <category term="applications.Connection" scheme="http://
schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
  <content type="application/xml">
    <m:properties>
      <d:ApnsDeviceToken>{APNS device token received by the
application from APNS}</d:ApnsDeviceToken>
      <d:DeviceType>iPhone</d:DeviceType>
    </m:properties>
  </content>
</entry>
```

3. Native mobile listener should be implemented by the application to get the notification.

Setting or Updating Application Settings

Set or update application settings by issuing a PUT or MERGE request.

Use the PUT method to update the application settings with all the properties in the request.

```
PUT http://host:port/[public/]odata/applications/{v1|latest}/
{appid}/Connections('{appcid}')
```

Use the MERGE method to update specified properties. Those properties that you do not specify retain their current values.

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Connections('{appcid}')
```

Note: For a list of application connection properties for the HTTP client, see *Reference > Application Connection Properties* in *Developer Guide: REST API Applications*.

The URL contains these components:

- **host –** The name of the Unwired Server.
- **port –** The port for listening for OData requests; by default, 8000.
- **"public" prefix –** If included, an anonymous security configuration is used.
- **odata/applications/ –** Refers to the OData services associated to the application resources.
- **{V1|latest} –** The version of the service document.
- **appid –** The name of the application.
- **Connections –** The name of the OData collection.
- **appcid –** The application connection ID of the application instance interacting with the service.

Deleting an Application Connection

When you no longer require an application connection (for example, when a user logs out), you can issue a DELETE method to delete the application connection using the same address where you retrieve the application connection settings.

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Connections('{appcid}')
```

Retrieving a Customization Resource Bundle

You can retrieve a customization resource bundle by issuing a GET method.

Issue the GET method to the following URL:

```
http://host:port/[public/] bundles/{appid}/[{resourceBundleName}]
```

For general authentication, and resource naming and resolution, apply the rules described in *Application Requirements > Authentication* in *Developer Guide: REST API Applications*.

If the `{resourceBundleName}` is specified in the URL, the specified resource bundle is returned in the response body as a stream, otherwise, the resource bundle bound to its application connection settings is returned.

If the resource bundle is not found in Sybase Unwired Platform, error code 404 is returned. You cannot issue other HTTP methods (PUT/POST/DELETE) at the above URL.

Uploading Business Transactions for Tracing

Access the Business Transaction XML upload service to upload business transactions for tracing.

Select one of the two URL modes, anonymous or authenticated upload.

For anonymous upload, perform a POST request at the following URL:

```
http(s)://host:port/[public/]btx/{appid}
```

The `{appid}` is the application identifier.

Including `/public` in the URL uploads the BTX anonymously.

Note: The administrator must enable anonymous access for the application.

For authenticated upload, perform a POST request at the following URL:

```
http(s)://host:port/btx/{appid}
```

The client must provide user credentials at the transport level (HTTP authentication).

Note: The above URL applies where the server name and domain name match. If they do not match, use the X-SUP-DOMAIN header to specify the domain name.

Include two main elements in the request:

- The BTX content, which you send as a multipart form request (the standard encoding used by browsers for HTTP file upload).
- The application identifier (provided in the URL)

For general authentication, and resource naming and resolution, apply the rules described in *Application Requirements > Authentication* in *Developer Guide: REST API Applications*.

Upon receiving the client request, the Unwired Server parses the multipart form request to gather the BTX content sent by the client and send this content to the SAP® Solution Manager in another multipart request.

For more information on supportability and the use of SAP Solution Manager with Sybase Unwired Platform, see *System Administration*.

Accessing a Service through a Proxy URL

Access your external service through a proxy URL. The URL supports read (HTTP GET), create (HTTP POST), update (HTTP PUT or HTTP MERGE), and delete (HTTP DELETE)

Perform an HTTP request at the following URL:

```
http://host:port/[public/]/{proxyName}
```

Testing Applications

Test native applications on a device or simulator.

For additional information about testing applications, see these topics in the Mobile Application Life Cycle collection:

- *Recommended Test Methodologies*
- *Best Practices for Testing Applications on a Physical Device*

Reference

Describes REST API resources.

Note: The example in this section use basic authentication, which you can see in the HTTP header, for example `Authorization: Basic` *`Base64-encoded string`*.

HTTP Headers and Cookies

Use HTTP headers and cookies to communicate application connection information or override the default security configuration of the domain.

Headers

- **X-SUP-DOMAIN –** Sybase Unwired Platform domain name to override the domain derived from the URL.

  For example, if the “mobile.sap.com” is the domain name requested in URL, but client wants to use the “mobile.application” domain. In this case, client can send HTTP request where X-SUP-DOMAIN header value will be “mobile.application”.
- **X-SUP-SC –** Security configuration to override domain default security configuration.

  For example, “mobile.application” domain has two assigned security configurations: “sc1” and “sc2”, and “sc1” is the default security configuration. By default, the “sc1” will be used to authenticate user. If client wants to authenticate using “sc2”, it can send HTTP request with the X-SUP-SC header as “sc2”.

  Note: X-SUP-DOMAIN and X-SUP-SC complement each other but do not necessarily need to be set together. If X-SUP-SC is used, you must associate the security configuration to the domain in which the request is authenticated to allow a successful connection, otherwise a 403 error is returned to the device. If a request refers to a non-existent domain or security configuration, 404 error is returned to the device. If the domain is disabled, a 503 error is returned.
- **X-SUP-BACKEND-URL –** Application can provide the backend request URL via X-SUP-BACKEND-URL header while sending the request to Unwired Server. The server forwards the request to the URL as provided in the X-SUP-BACKEND-URL.

  If the X-SUP-BACKEND-URL header is present in the request, then URL rewrite will be disabled, that is URL rewrite is turned off.

  Note: The URL should be whitelisted in server, else you receive an error.

Cookies

Cookies are returned by servers in the HTTP response header (`Set-Cookie` header) and included by the HTTP client (for example, a browser) in the subsequent HTTP request header (cookie header).

- **X-SUP-APPCID –** Communicates the application connection ID as generated by the application the request originates from, or as issued by the server during using the X-SUP-APPCID response header and the permanent X-SUP-APPCID cookie. Sybase Unwired Platform response headers carry the X-SUP-APPCID regardless of whether or not the incoming request carried a cookie or X-SUP-APPCID header. The X-SUP-APPCID value has a maximum length of 128 characters. If the request does not carry an X-SUP-APPCID, the server generates a value automatically.

  X-SUP-APPCID is received as a cookie, but sent back to the server either as a cookie or as a header.
- **X-SUP-SECTOKEN –** Cookie generated for anonymous registration.

  Security token generated to protect anonymous registered application connection settings not to be changed by other device users.

  X-SUP-SECTOKEN is received as a cookie, but sent back to the server either as a cookie or as a header.
- **X-SUP-SESSID –** X-SUP-SESSID is the actual JSESSIONID issued by the web container. If X-SUP-SESSID is provided in the request header, server bypasses the security configuration check during the session time.

  If the X-SUP-SESSID is not provided, the Sybase Unwired Platform performs authentication through security configuration against backend security server.

  The default session timeout is 30 minutes. If the session times out, or if the client does not send the X-SUP-SESSID cookie in the request, a new session is created on the server.

  You can change the default timeout by manually adding the session-timeout configuration into the `SUP_HOME\Servers\UnwiredServer\deploy\webapps\httpchannel\WEB-INF\web.xml` file, and restarting the server. You must manually modify all the nodes:

  ```
  <web-app>
  ...
  <session-config>
  <session-timeout>30</session-timeout>
  </session-config>
  ...
  </web-app>
  ```

  Note: The X-SUP-SESSID cookie should be used by load balancers to support server affinity. Based on the X-SUP-SESSID HTTP header, load balancers directly dispatches other client requests to Unwired Platform. For more information, see *Requirements for Client Load Balancing* in *Landscape Design and Integration*.

Application Connection Properties

Describes application connection properties, and indicates whether the properties are read-only or nullable from the HTTP client.

Note: If you attempt to modify a read-only property, the client application throws the following exception: `HTTP 403 - The property "XXX" cannot be updated by a client application`.

For usage and description of application connection properties, search for *Application Connection Properties* in *System Administration* .

Table 1. Application Connection Properties: Uncategorized

Property Name	Type	Read-only?	Is nullable?
ETag	String	Yes	No
ApplicationConnectionId	String	Yes	No

Table 2. Application Connection Properties: Android Push

Property Name	Type	Read-only?	Is nullable?
AndroidGcmPushEnabled	Boolean	No	No
AndroidGcmRegistrationId	String	No	Yes
AndroidGcmSenderId	String	Yes	Yes

Table 3. Application Connection Properties: Apple Push

Property Name	Type	Read-only?	Is nullable?
ApnsPushEnable	Boolean	No	No
ApnsAlerts	Boolean	No	No
ApnsBadges	Boolean	No	No
ApnsSounds	Boolean	No	No
ApnsAlertMessage	String	Yes	Yes

Property Name	Type	Read-only?	Is nullable?
ApnsDeviceToken	String	No	Yes

Table 4. Application Connection Properties: Application Settings

Property Name	Type	Read-only?	Is nullable?
CustomizationBundleId	String	Yes	Yes
ApplicationVersion	String	No	Yes
ClientSdkVersion	String	No	Yes

Table 5. Application Connection Properties: BlackBerry Push

Property Name	Type	Read-only?	Is nullable?
BlackberryPushEnabled	Boolean	No	No
BlackberryDevicePin	String	No	Yes
BlackberryBESListenerPort	Int32	No	No

Table 6. Application Connection Properties: Capabilities

Property Name	Type	Read-only?	Is nullable?
CapabilitiesPasswordPolicy	Boolean	No	No

Table 7. Application Connection Properties: Custom Settings

Property Name	Type	Read-only?	Is nullable?
CustomCustom1	String	No	Yes
CustomCustom2	String	No	Yes
CustomCustom3	String	No	Yes
CustomCustom4	String	No	Yes

Table 8. Application Connection Properties: Device Information

Property Name	Type	Read-only?	Is nullable?
DeviceModel	String	No	Yes
DeviceType	String	No	Yes
DeviceSubType	String	No	Yes
DevicePhoneNumber	String	No	Yes
DeviceIMSI	String	No	Yes

Table 9. Application Connection Properties: Password Policy

Property Name	Type	Read-only?	Is nullable?
PasswordPolicyEnabled	Boolean	Yes	No
PasswordPolicyDefaultPasswordAllowed	Boolean	Yes	No
PasswordPolicyMinLength	Int32	Yes	No
PasswordPolicyDigitRequired	Boolean	Yes	No
PasswordPolicyUpperRequired	Boolean	Yes	No
PasswordPolicyLowerRequired	Boolean	Yes	No
PasswordPolicySpecialRequired	Boolean	Yes	No
PasswordPolicyExpiresInNDays	Int32	Yes	No
PasswordPolicyMinUniqueChars	Int32	Yes	No
PasswordPolicyLockTimeout	Int32	Yes	No
PasswordPolicyRetryLimit	Int32	Yes	No

Table 10. Application Connection Properties: Proxy

Property Name	Type	Read-only?	Is nullable?
ProxyApplicationEnd-point	String	Yes	Yes
ProxyPushEndpoint	String	Yes	Yes

Proxy Responses

Proxy responses include all the cookies and headers from the proxied backend. Refer to the proxy documentation for more details.

Error Code and Message Format

Server returns different format for error code and messages according to different "Accept" value in request header.

Table 11. Accept header and data format

Type and Format	Accept header samples	Response body
XML	application/xml, application/xhtml+xml, application/atom+xml **Note:** If error occurs in an OData request, include (xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata") namespace, otherwise not.	<?xml version="1.0" encoding="utf-8" stand-alone="yes"?> <error xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"> <code>403</code> <message xml:lang="en-US">some specific error text string</message> </error>
JSON	application/json, text/json	{ "error": {"code": "403", "message": {"lang": "en-US", "value": "some specific error text string" } } }
TEXT	text/html, text/plain	"some specific error text string"

Note: If the Accept header does not include any of these data types, the response body is null.

Application Connections

Methods for creating, updating, or reading application connections.

Service Document

Get the service document for the application connection. Retrieving the service document allows the client to discover the capabilities and locations of the available collections.

Syntax

Perform an HTTP GET request at the following URL:

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}
```

Parameters

- **appid –** The application ID that uniquely identifies the application.

Returns

Returns a 200 OK status code if successful, and a service document in the response body.

Examples

- **Service document –** HTTP request header:

  ```
  GET /odata/applications/v1/com.sap.myapp HTTP/1.1
  Host: supserver:8000
  Connection: Keep-Alive
  User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
  Authorization: Basic REVWMDAwMTppbml0aWFs
  ```

  HTTP response header:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/xml;charset=utf-8
  Expires: Thu, 01 Jan 1970 00:00:00 GMT
  Set-Cookie: X-SUP-SESSID=97ts80gwhxkc;Path=/
  Set-Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771
  DataServiceVersion: 1.0
  Date: Tue, 14 Aug 2012 21:28:55 GMT
  Transfer-Encoding: chunked
  ```

  The HTTP response body is a service document that includes two collections named "Connections" and "Endpoints".

  ```
  <?xml version='1.0' encoding='utf-8'?>
  <service xmlns="http://www.w3.org/2007/app"
           xml:base="http://host:port/appSettings/odata/v1/appid/"
           xmlns:atom="http://www.w3.org/2005/Atom"
           xmlns:app="http://www.w3.org/2007/app">
  ```

```
    <workspace>
        <atom:title>Default</atom:title>
        <collection href="Connections">
            <atom:title>Connections</atom:title>
        <collection href="Endpoints">
            <atom:title>Endpoints</atom:title>
        </collection>
    </workspace>
</service>
```

Metadata

Get the metadata document that includes the metadata for the application connection settings and proxy endpoints.

Syntax

Perform an HTTP GET request at the following URL:

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
$metadata
```

Parameters

- **appid –** The application ID that uniquely identifies the application.

Returns

If successful, returns a 200 OK status code and a metadata document in the response body.

Examples

- **Get Metadata –** HTTP request header:

  ```
  GET /odata/applications/v1/com.sap.myapp/$metadata HTTP/1.1
  Host: supserver:8000
  Connection: Keep-Alive
  User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
  Authorization: Basic REVWMDAwMTppbml0aWFs
  ```

  HTTP response header:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/xml;charset=utf-8
  Expires: Thu, 01 Jan 1970 00:00:00 GMT
  Set-Cookie: X-SUP-SESSID=97ts80gwhxkc;Path=/
  Set-Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771
  DataServiceVersion: 1.0
  Date: Tue, 14 Aug 2012 21:32:34 GMT
  Transfer-Encoding: chunked
  ```

  HTTP response body (metadata):

  ```
  <?xml version='1.0' encoding='utf-8'?>
  <edmx:Edmx Version="1.0" xmlns:edmx="http://
  schemas.microsoft.com/ado/2007/06/edmx"
  ```

```
  xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/
metadata"
  xmlns:sup="http://www.sap.com/sup/odata">
  <edmx:DataServices m:DataServiceVersion="1.0">
    <Schema xmlns="http://schemas.microsoft.com/ado/2006/04/edm"
Namespace="applications">
      <EntityType Name="Connection">
        <Key>
          <PropertyRef Name="ApplicationConnectionId" />
        </Key>
        <Property Name="ETag" Type="Edm.String" Nullable="false"
sup:ReadOnly="true"/>
       <Property Name="ApplicationConnectionId" Type="Edm.String"
Nullable="false" sup:ReadOnly="true"/>
       <Property Name="AndroidGcmPushEnabled" Type="Edm.Boolean"
Nullable="false" sup:ReadOnly="false" />
        <Property Name="AndroidGcmRegistrationId"
Type="Edm.String" Nullable="true" sup:ReadOnly="false" />
        <Property Name="AndroidGcmSenderId" Type="Edm.String"
Nullable="true" sup:ReadOnly="true" />
        <Property Name="ApnsPushEnable" Type="Edm.Boolean"
Nullable="false" sup:ReadOnly="false" />
        <Property Name="ApnsAlerts" Type="Edm.Boolean"
Nullable="false" sup:ReadOnly="false" />
        <Property Name="ApnsBadges" Type="Edm.Boolean"
Nullable="false" sup:ReadOnly="false" />
        <Property Name="ApnsSounds" Type="Edm.Boolean"
Nullable="false" sup:ReadOnly="false" />
        <Property Name="ApnsAlertMessage" Type="Edm.String"
Nullable="true" sup:ReadOnly="true" />
        <Property Name="ApnsDeviceToken" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="CustomizationBundleId" Type="Edm.String"
Nullable="true" sup:ReadOnly="true" />
        <Property Name="ApplicationVersion" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="ClientSdkVersion" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
       <Property Name="BlackberryPushEnabled" Type="Edm.Boolean"
Nullable="false" sup:ReadOnly="false" />
        <Property Name="BlackberryDevicePin" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="BlackberryBESListenerPort"
Type="Edm.Int32" Nullable="false" sup:ReadOnly="false" />
        <Property Name="CapabilitiesPasswordPolicy"
Type="Edm.Boolean" Nullable="false" sup:ReadOnly="false" />
        <Property Name="CustomCustom1" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="CustomCustom2" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="CustomCustom3" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="CustomCustom4" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="DeviceModel" Type="Edm.String"
```

```
Nullable="true" sup:ReadOnly="false" />
        <Property Name="DeviceType" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="DeviceSubType" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="DevicePhoneNumber" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="DeviceIMSI" Type="Edm.String"
Nullable="true" sup:ReadOnly="false" />
        <Property Name="PasswordPolicyEnabled" Type="Edm.Boolean"
Nullable="false" sup:ReadOnly="true" />
        <Property Name="PasswordPolicyDefaultPasswordAllowed"
Type="Edm.Boolean" Nullable="false" sup:ReadOnly="true" />
        <Property Name="PasswordPolicyMinLength" Type="Edm.Int32"
Nullable="false" sup:ReadOnly="true" />
        <Property Name="PasswordPolicyDigitRequired"
Type="Edm.Boolean" Nullable="false" sup:ReadOnly="true" />
        <Property Name="PasswordPolicyUpperRequired"
Type="Edm.Boolean" Nullable="false" sup:ReadOnly="true" />
        <Property Name="PasswordPolicyLowerRequired"
Type="Edm.Boolean" Nullable="false" sup:ReadOnly="true" />
        <Property Name="PasswordPolicySpecialRequired"
Type="Edm.Boolean" Nullable="false" sup:ReadOnly="true" />
        <Property Name="PasswordPolicyExpiresInNDays"
Type="Edm.Int32" Nullable="false" sup:ReadOnly="true" />
        <Property Name="PasswordPolicyMinUniqueChars"
Type="Edm.Int32" Nullable="false" sup:ReadOnly="true" />
        <Property Name="PasswordPolicyLockTimeout"
Type="Edm.Int32" Nullable="false" sup:ReadOnly="true" />
       <Property Name="PasswordPolicyRetryLimit" Type="Edm.Int32"
Nullable="false" sup:ReadOnly="true" />
        <Property Name="ProxyApplicationEndpoint"
Type="Edm.String" Nullable="true" sup:ReadOnly="true" />
        <Property Name="ProxyPushEndpoint" Type="Edm.String"
Nullable="true" sup:ReadOnly="true" />
      </EntityType>
      <EntityType Name="Endpoint">
        <Key>
         <PropertyRef Name="EndpointName" />
       </Key>
       <Property Name="RemoteURL" Type="Edm.String"
Nullable="false" />
       <Property Name="EndpointName" Type="Edm.String"
Nullable="false" />
       <Property Name="AnonymousAccess" Type="Edm.Boolean"
Nullable="false" />
      </EntityType>

      <EntityContainer Name="Container"
m:IsDefaultEntityContainer="true">
        <EntitySet Name="Connections"
EntityType="applications.Connection" />
        <EntitySet Name="Endpoints"
EntityType="applications.Endpoint" /></EntityContainer>
    </Schema>
```

```
  </edmx:DataServices>
</edmx:Edmx>
```

Create Application Connection

Create an application connection and initially set the application connection settings. Because all application connection settings are optional, the minimal body contains no properties at all. Sybase Unwired Platform populates default values as needed.

Syntax

Perform an HTTP POST request at the following URL:

```
http://host:port/[public/]/odata/applications/{v1|latest}/{appid}/
Connections
```

Parameters

- **appid –** The application ID that uniquely identifies the application.

Returns

If successful, a 201 Created status code is returned, and the new application connection settings are included in the response body.

Examples

- **Create application connection –** Example HTTP request header:

  ```
  POST /odata/applications/v1/com.sap.myapp/Connections HTTP/1.1
  Content-Length: 4704
  Content-Type: application/atom+xml; charset=UTF-8
  Host: supserver:8000
  Connection: Keep-Alive
  User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
  Authorization: Basic REVWMDAwMTppbml0aWFs
  ```

  Example HTTP response header:

  ```
  HTTP/1.1 201 Created
  Content-Type: application/atom+xml;charset=utf-8
  Expires: Thu, 01 Jan 1970 00:00:00 GMT
  Set-Cookie: X-SUP-SESSID=97ts80gwhxkc;Path=/
  Set-Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771
  Location: http:/supserver:8000/applications/
  Connections('b6d50e93-bcaa-439d-9741-660a3cb56771')
  DataServiceVersion: 1.0
  Date: Mon, 13 Aug 2012 22:40:50 GMT
  Transfer-Encoding: chunked
  ```

  Minimal body:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <entry xml:base="http://localhost:8000/public/odata/applications/
  ```

```
v1/com.sap.myapp/"
xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://
schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
  <content type="application/xml">
    <m:properties>
    </m:properties>
  </content>
</entry>
```

Get Application Settings

Get the application settings.

Syntax

Perform an HTTP GET request at the following URL:

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Connections('{appcid}')
```

Parameters

- **appid –** The application ID that uniquely identifies the application.
- **appcid –** The application connection ID of the application instance interacting with the service.

Returns

If successful, returns a 200 OK status code and an HTTP response body with the application settings.

Examples

- **Get application settings –** HTTP request header:

  ```
  GET /odata/applications/v1/com.sap.myapp/Connections('b6d50e93-
  bcaa-439d-9741-660a3cb56771') HTTP/1.1
  Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771; X-SUP-
  SESSID=97ts80gwhxkc
  Host: supserver:8000
  Connection: Keep-Alive
  User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
  Authorization: Basic REVWMDAwMTppbml0aWFs
  ```

  HTTP response header:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/atom+xml;charset=utf-8
  Expires: Thu, 01 Jan 1970 00:00:00 GMT
  Set-Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771
  DataServiceVersion: 1.0
  ```

```
Date: Mon, 13 Aug 2012 22:56:50 GMT
Transfer-Encoding: chunked
```

HTTP response body:

```
<?xml version='1.0' encoding='utf-8'?>
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:m="http://schemas.microsoft.com/ado/2007/08/
dataservices/metadata"
       xmlns:d="http://schemas.microsoft.com/ado/2007/08/
dataservices"
       xml:base="http://localhost:8000/odata/applications/v1/
e2eTest/">
  <id>http://localhost:8000/odata/applications/v1/e2eTest/
Connections('c9d8a9da-9f36-4ae5-9da5-37d6d90483b5')</id>
  <title type="text" />
  <updated>2012-06-28T09:55:48Z</updated>
  <author><name /></author>
  <link rel="edit" title="Connections"
href="Connections('c9d8a9da-9f36-4ae5-9da5-37d6d90483b5')" />
  <category term="applications.Connection" scheme="http://
schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
  <content type="application/xml">
    <m:properties>
      <d:ETag>2012-06-28 17:55:47.685</d:ETag>

<d:ApplicationConnectionId>c9d8a9da-9f36-4ae5-9da5-37d6d90483b5</
d:ApplicationConnectionId>
      <d:AndroidGcmPushEnabled m:type="Edm.Boolean">false</
d:AndroidGcmPushEnabled>
      <d:AndroidGcmRegistrationId m:null="true" />
      <d:AndroidGcmSenderId m:null="true" />
      <d:ApnsPushEnable m:type="Edm.Boolean">true</
d:ApnsPushEnable>
      <d:ApnsAlerts m:type="Edm.Boolean">true</d:ApnsAlerts>
      <d:ApnsBadgesm:type="Edm.Boolean">true</d:ApnsBadges>
      <d:ApnsSounds m:type="Edm.Boolean">true</d:ApnsSounds>
     <d:ApnsAlertMessage>New Items Available</d:ApnsAlertMessage>
      <d:ApnsDeviceToken m:null="true" />
      <d:CustomizationBundleId>TestMDC:1.0</d:
CustomizationBundleId>
      <d:ApplicationVersion m:null="true" />
      <d:ClientSdkVersion m:null="true" />
      <d:BlackberryPushEnabled m:type="Edm.Boolean">true</
d:BlackberryPushEnabled>
      <d:BlackberryDevicePin>00000000</d:BlackberryDevicePin>
      <d:BlackberryBESListenerPort m:type="Edm.Int32">5011</
d:BlackberryBESListenerPort>
      <d:CapabilitiesPasswordPolicy m:type="Edm.Boolean">false</
d:CapabilitiesPasswordPolicy>
      <d:CustomCustom1>custom1</d:CustomCustom1>
      <d:CustomCustom2 m:null="true" />
      <d:CustomCustom3 m:null="true" />
      <d:CustomCustom4 m:null="true" />
      <d:DeviceModel m:null="true" />
      <d:DeviceType>Unknown</d:DeviceType>
      <d:DeviceSubType m:null="true" />
      <d:DevicePhoneNumber>12345678901</d:DevicePhoneNumber>
      <d:DeviceIMSI m:null="true" />
      <d:PasswordPolicyEnabled m:type="Edm.Boolean">true</
d:PasswordPolicyEnabled>
      <d:PasswordPolicyDefaultPasswordAllowed
m:type="Edm.Boolean">false</
d:PasswordPolicyDefaultPasswordAllowed>
      <d:PasswordPolicyMinLength m:type="Edm.Int32">8</
d:PasswordPolicyMinLength>
      <d:PasswordPolicyDigitRequired m:type="Edm.Boolean">false</
d:PasswordPolicyDigitRequired>
      <d:PasswordPolicyUpperRequired m:type="Edm.Boolean">false</
d:PasswordPolicyUpperRequired>
      <d:PasswordPolicyLowerRequired m:type="Edm.Boolean">false</
d:PasswordPolicyLowerRequired>
      <d:PasswordPolicySpecialRequired
m:type="Edm.Boolean">false</d:PasswordPolicySpecialRequired>
      <d:PasswordPolicyExpiresInNDays m:type="Edm.Int32">0</
d:PasswordPolicyExpiresInNDays>
      <d:PasswordPolicyMinUniqueChars m:type="Edm.Int32">0</
d:PasswordPolicyMinUniqueChars>
      <d:PasswordPolicyLockTimeout m:type="Edm.Int32">0</
d:PasswordPolicyLockTimeout>
      <d:PasswordPolicyRetryLimit m:type="Edm.Int32">20</
d:PasswordPolicyRetryLimit>
      <d:ProxyApplicationEndpointm:null="true" />
      <d:ProxyPushEndpoint>http://port:8000/GWC/SUPNotification</
d:ProxyPushEndpoint>
  </content>
</entry>
```

Get Proxy Endpoints

Get all the proxy endpoints.

HTTP client can get the proxy endpoints according to different domain through OData service. Client can set the domain value in request header `X-SUP-DOMAIN`.

Note:

- If `X-SUP-DOMAIN` is not set in the request header, the “default” domain will be used.
- Only GET operation is supported for proxy endpoints. If client tries to send POST/PUT/DELETE request, the server returns 501 error.

Perform an HTTP GET request at the following URL:

Syntax

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Endpoints
```

Parameters

- **appid –** The application ID that uniquely identifies the application.

Examples

- **Get proxy endpoints –** HTTP request header:

  ```
  X-SUP-DOMAIN: default
  ```

  HTTP response header:

  ```
  Status Code: 200 OK
  Content-Type: application/xml;charset=utf-8
  DataServiceVersion: 2.0
  Date: Thu, 02 May 2013 06:52:21 GMT
  Expires: Thu, 01 Jan 1970 00:00:00 GMT
  Set-Cookie: X-SUP-APPCID=9dffe5e9-5768-47a6-8220-144a2e0c751d
  Transfer-Encoding: chunked
  ```

  HTTP response body:

  ```
  <?xml version='1.0' encoding='utf-8'?>
     <feed xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://
  schemas.microsoft.com/ado/2007/08/dataservices/metadata"
     xmlns:d="http://schemas.microsoft.com/ado/2007/08/
  dataservices"
     xml:base="http://localhost:8000/odata/applications/v1/
  e2eTest/">
    <title type="text">Endpoints</title>
    <id>http://localhost:8000/odata/applications/v1/e2eTest/
  Endpoints</id>
    <updated>2013-04-10T02:36:23Z</updated>
    <link rel="self" title="Endpoints" href="Endpoints" />
    <m:count>2</m:count>
    <entry>
       <id>http://localhost:8000/odata/applications/v1/e2eTest/
  Endpoints('endpoint1')</id>
      <title type="text" />
      <updated>2013-04-10T02:36:23Z</updated>
      <author>
        <name/>
      </author>
      <link rel="edit" title="Endpoint"
  href="Endpoints('endpoint1')" />
      <category term="applications.Endpoint"    scheme="http://
  schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
      <content type="application/xml">
      <m:properties>
        <d:RemoteURL>http://</d:RemoteURL>
        <d:EndpointName>endpoint1</d:EndpointName>
        <d:AnonymousAccess m:type="Edm.Boolean">true</
  d:AnonymousAccess>
      </m:properties>
    </content>
   </entry>
  <entry>
  ```

```
     <id>http://localhost:8000/odata/applications/v1/e2eTest/
Endpoints('endpoint2')</id>
    <title type="text" />
    <updated>2013-04-10T02:36:23Z</updated>
    <author>
      <name/>
    </author>
    <link rel="edit" title="Endpoint"
href="Endpoints('endpoint2')" />
    <category term="applications.Endpoint"    scheme="http://
schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
    <content type="application/xml">
    <m:properties>
      <d:RemoteURL>http://</d:RemoteURL>
      <d:EndpointName>endpoint2</d:EndpointName>
      <d:AnonymousAccess m:type="Edm.Boolean">false</
d:AnonymousAccess>
    </m:properties>
  </content>
 </entry>
</feed>
```

Get Proxy Endpoint by Endpoint Name

Get a specific endpoint by specify the endpoint name.

Perform an HTTP GET request at the following URL:

Syntax

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Endpoints('{endpoint}')
```

Parameters

- **appid –** The application ID that uniquely identifies the application.
- **endpoint –** The proxy endpoint name.

Returns

Returns 404 error if client tries to retrieve a non-existing endpoint.

```
Response header:
Status Code: 404 Not Found
Content-Type: application/atom+xml;charset=utf-8
Date: Wed, 10 Apr 2013 02:51:56 GMT
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: X-SUP-APPCID=d6d5859c-07a2-4997-a4b0-dc3e3d1f987f
Transfer-Encoding: chunked
```

Examples

- **Get proxy endpoints by name –** HTTP request header:

  ```
  X-SUP-DOMAIN: default
  ```

HTTP response header:

```
Status Code: 200 OK
Content-Type: application/xml;charset=utf-8
DataServiceVersion: 1.0
Date: Thu, 02 May 2013 07:00:43 GMT
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: X-SUP-APPCID=9dffe5e9-5768-47a6-8220-144a2e0c751d
Transfer-Encoding: chunked
```

HTTP response body:

```
<?xml version='1.0' encoding='utf-8'?>
<entry xmlns="http://www.w3.org/2005/Atom"   xmlns:m="http://
schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xml:base="http://localhost:8000/odata/applications/v1/e2eTest/">
  <id>http://localhost:8000/odata/applications/v1/e2eTest/
Endpoints('endpoint1')</id>
  <title type="text" />
  <updated>2013-04-10T02:46:59Z</updated>
  <author>
    <name />
  </author>
  <link rel="edit" title="Endpoint"
href="Endpoints('endpoint1')" />
  <category term="applications.Endpoint" scheme="http://
schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
  <content type="application/xml">
    <m:properties>
      <d:RemoteURL>http://</d:RemoteURL>
      <d:EndpointName>endpoint1</d:EndpointName>
      <d:AnonymousAccess m:type="Edm.Boolean">true</
d:AnonymousAccess>
    </m:properties>
  </content>
</entry>
```

Get Proxy Endpoint by Property Name

Get a proxy endpoint property value by appending the property name in URL.

Perform an HTTP GET request at the following URL:

Syntax

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Endpoints('{endpoint}')/EndpointName
```

Parameters

- **appid –** The application ID that uniquely identifies the application.
- **endpoint –** The proxy endpoint name.

Examples

- **Get property endpoint by property name –** HTTP request header:

  ```
  X-SUP-DOMAIN: default
  ```

  HTTP response header:

  ```
  Status Code: 200 OK
  Content-Type: application/xml;charset=utf-8
  DataServiceVersion: 1.0
  Date: Thu, 02 May 2013 07:03:05 GMT
  Expires: Thu, 01 Jan 1970 00:00:00 GMT
  Set-Cookie: X-SUP-APPCID=9dffe5e9-5768-47a6-8220-144a2e0c751d
  Transfer-Encoding: chunked
  ```

  HTTP response body

  ```
  <?xml version='1.0' encoding='utf-8'?>
  <EndpointName xmlns="http://schemas.microsoft.com/ado/2007/08/
  dataservices">
    endpoint1
  </EndpointName>
  ```

Get Application Settings (Property)

Get the specific property value for a property from the application settings.

Syntax

Perform an HTTP GET request at the following URL:

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Connections(‘{appcid}’)/<property-name>
```

Parameters

- **appid –** The application ID that uniquely identifies the application.
- **appcid –** The application connection ID of the application instance interacting with the service.
- **property-name –** The property-name can be appended to the URL to retrieve the value of a specific property.

Returns

If successful, returns a 200 OK status code and an HTTP response body with the application settings.

Examples

- **Get the DeviceType property –** HTTP request:

  ```
  GET /odata/applications/v1/com.sap.myapp/Connections('b6d50e93-
  bcaa-439d-9741-660a3cb56771')/DeviceType HTTP/1.1
  ```

```
Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771; X-SUP-
SESSID=97ts80gwhxkc
Host: supserver:8000
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
Authorization: Basic REVWMDAwMTppbml0aWFs
```

HTTP response header:

```
HTTP/1.1 200 OK
Content-Type: application/xml;charset=utf-8
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771
DataServiceVersion: 1.0
Date: Mon, 13 Aug 2012 23:00:27 GMT
Transfer-Encoding: chunked
```

HTTP response body:

```
<?xml version='1.0' encoding='utf-8'?>
<DeviceType xmlns="http://schemas.microsoft.com/ado/2007/08/
dataservices">
Windows
</DeviceType>
```

Update Application Settings (PUT)

Update the application settings with all the properties in the request.

Any properties that you do not specify (and which can be changed) are set to the default value.

Syntax

```
PUT http://host:port/[public/]odata/applications/{v1|latest}/
{appid}/Connections(‘{appcid}’)
```

Parameters

- **appid –** The application ID that uniquely identifies the application.
- **appcid –** The application connection ID of the application instance interacting with the service.

Returns

When processing a PUT request, Sybase Unwired Platform returns a 200 status code to indicate success, and there is no response body.

If you never explicitly registered the client, returns a 404 status code.

Examples

- **Update application settings (PUT) –** HTTP request header:

  ```
  PUT /odata/applications/v1/com.sap.myapp/Connections('b6d50e93-
  bcaa-439d-9741-660a3cb56771') HTTP/1.1
  ```

```
Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771; X-SUP-
SESSID=97ts80gwhxkc
Content-Length: 4744
Content-Type: application/atom+xml; charset=UTF-8
Host: supserver:8000
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
Authorization: Basic REVWMDAwMTppbml0aWFs
```

[PUT HTTP request message body]

HTTP response header:

```
HTTP/1.1 200 OK
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771
DataServiceVersion: 1.0
Date: Mon, 13 Aug 2012 23:07:51 GMT
Content-Length: 0
```

Update Application Settings (MERGE)

Update the specified application settings.

Use the MERGE method to update the specified properties. Those properties that you do not specify retain their current values.

Syntax

Perform a MERGE request at the following URL:

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Connections(‘{appcid}’)
```

Parameters

- **appid –** The application ID that uniquely identifies the application.
- **appcid –** The application connection ID of the application instance interacting with the service.

Returns

If successful, returns a 200 OK status code.

If you never explicitly registered the client, returns a 404 status code.

Examples

- **Update application settings (MERGE) –** HTTP request header:

```
POST /odata/applications/v1/com.sap.myapp/Connections('b6d50e93-
bcaa-439d-9741-660a3cb56771') HTTP/1.1
X-HTTP-METHOD:MERGE
Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771; X-SUP-
```

```
SESSID=97ts80gwhxkc
Content-Length: 4768
Content-Type: application/atom+xml; charset=UTF-8
Host: supserver:8000
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
Authorization: Basic REVWMDAwMTppbml0aWFs
```

[MERGE HTTP request message body]

HTTP request body:

```
<?xml version='1.0' encoding='utf-8'?>
  <entry xmlns="http://www.w3.org/2005/Atom"
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/
dataservices"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/
dataservices/metadata">
    <title type="text"/>
    <updated>2012-06-15T02:23:29Z</updated>
    <author>
      <name/>
     </author>
     <category term="applications.Connection" scheme="http://
schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
       <content type="application/xml">
       <m:properties>
         <d:DeviceType>Unknown</d:DeviceType>
         …
       </m:properties>
       </content>
  </entry>
```

HTTP response header:

```
HTTP/1.1 200 OK
Content-Length: 0
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771
DataServiceVersion: 1.0
Date: Mon, 13 Aug 2012 23:13:04 GMT
```

Delete Application Connection

Delete an application connection.

Syntax

Perform an HTTP DELETE request at the following URL:

```
http://host:port/[public/]odata/applications/{v1|latest}/{appid}/
Connections(‘{appcid}’)
```

Parameters

- **appid –** The application ID that uniquely identifies the application.
- **appcid –** The application connection ID of the application instance interacting with the service.

Returns

If successful, returns a 200 OK status code.

If you never explicity registered the client, returns a 404 status code.

Examples

- **Delete application connection –** HTTP request:

  ```
  DELETE /odata/applications/v1/com.sap.myapp/
  Connections('b6d50e93-bcaa-439d-9741-660a3cb56771') HTTP/1.1
  Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771; X-SUP-
  SESSID=97ts80gwhxkc
  Host: supserver:8000
  Connection: Keep-Alive
  User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
  Authorization: Basic REVWMDAwMTppbml0aWFs
  ```

  HTTP response:

  ```
  HTTP/1.1 200 OK
  Expires: Thu, 01 Jan 1970 00:00:00 GMT
  Set-Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771
  DataServiceVersion: 1.0
  Date: Mon, 13 Aug 2012 23:19:42 GMT
  Content-Length: 0
  ```

Application Customization Resource Bundles

Methods for customizing applications.

Download Customization Resource Bundle

Download the customization resource bundle.

Note: Downloading customization resource bundle using REST API is supported only in Object API SDK or OData SDK applications.

Syntax

Perform an HTTP GET request at the following URL:

```
http://host:port/[public/] bundles/{appid}/[{resourceBundleName}]
```

Parameters

- **resourceBundleName –** Optionally specifies a resource bundle to be returned.

Returns

If you specify the `{resourceBundleName}` in the URL, the specified resource bundle is returned in the response body as a stream. Otherwise, the resource bundle bound to its application connection settings is returned.

Examples

- **Download –** HTTP request:

  ```
  GET /bundles/com.sap.myapp/MyApp:1.0 HTTP/1.1
  Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771; X-SUP-
  SESSID=97ts80gwhxkc
  Host: supserver:8000
  Connection: Keep-Alive
  User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
  Authorization: Basic REVWMDAwMTppbml0aWFs
  ```

  HTTP response:

  ```
  HTTP/1.1 200 OK
  Expires: Thu, 01 Jan 1970 00:00:00 GMT
  Set-Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771
  Transfer-Encoding: chunked

  [resource bundle content]
  ```

Usage

For information on the format of customization resource bundles, see *Administer > Applications > Maintaining Activated Applications > Modifying Application Properties > Application Customization Resource Bundles* in *Sybase Control Center for Sybase Unwired Platform*.

Business Transactions

Methods for uploading business transactions.

Upload BTX

Access the Business Transaction XML upload service to upload business transactions for tracing.

Upon receiving the client request, the Unwired Server parses the multipart form request to gather the Business Transaction content sent by the client and sends this content to the SAP Solution Manager in another multipart request.

Syntax

Perform a POST request to the following URL:

```
http://host:port/[public/]btx/{appid}
```

Parameters

None.

Returns

On successful upload, the client receives a 201 Created status code. Otherwise, an HTTP failure code and failure message are returned.

Examples

- **Upload Business Transaction –** HTTP request header:

  ```
  POST /btx/com.sap.myapp/ HTTP/1.1
  Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771; X-SUP-
  SESSID=97ts80gwhxkc
  Content-Length: 40
  Content-Type: application/atom+xml; charset=UTF-8
  Host: supserver:8000
  Connection: Keep-Alive
  User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
  Authorization: Basic REVWMDAwMTppbml0aWFs
  ```

  [POST HTTP request message body]

  HTTP response header:

  ```
  HTTP/1.1 201 Created
  Content-Type: application/atom+xml;charset=utf-8
  Expires: Thu, 01 Jan 1970 00:00:00 GMT
  Date: Mon, 13 Aug 2012 22:40:50 GMT
  Transfer-Encoding: chunked
  ```

Usage

Use this method to support your application using SAP Solution Manager. See SAP Solution Manager documentation for more information.

Proxy Connections

Methods for accessing proxy connections.

Access External Service

Unwired Server can also be used for accessing OData services provided by any external OData provider. The proxy URL supports read (HTTP GET), create (HTTP POST), update (HTTP PUT or HTTP MERGE), and delete (HTTP DELETE).

Syntax

Perform an HTTP request at the following URL:

```
http://host:port/[public/]/{proxyName}
```

Parameters

- **proxyName –** URL represents one of the configured proxy connection endpoints (data services exposed through Sybase Unwired Platform) whitelisted.

Returns

If successful, returns a 200 OK status code and a response body.

Examples

- **Access external service –** HTTP request header:

  ```
  GET /com.sap.myapp HTTP/1.1
  Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771; X-SUP-
  SESSID=97ts80gwhxkc
  Host: supserver:8000
  Connection: Keep-Alive
  User-Agent: Apache-HttpClient/4.1.3 (java 1.5)
  Authorization: Basic REVWMDAwMTppbml0aWFs
  ```

  HTTP response header:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/atomsvc+xml
  Expires: Thu, 01 Jan 1970 00:00:00 GMT
  Set-Cookie: X-SUP-APPCID=b6d50e93-bcaa-439d-9741-660a3cb56771
  server: SAP NetWeaver Application Server / ABAP 731
  dataserviceversion: 2.0
  set-cookie: SAP_SESSIONID_DG1_001=<Some-Token>; path=/
  set-cookie: MYSAPSSO2=<SSO-Token>; path=/; domain=.sap.com
  Content-Length: 2651
  ```

Index