Development Guide: Good Control Web Services

Last updated: Friday, June 23, 2017
Version: GC 3.0.xx.yy

Contents

Revision history 6

Good Control/UEM web services 7

- Relation of UEM web services and Good Control web services 7
- UEM Getting Started Guide for Making Web Services calls 7
- What's new in Good Control web services 7
  - Global changes: new product names and abbreviations 7
  - Changed SOAP API types, fields, and operations in gc.wsdl and cap.wsdl 8
  - Changes in web services in BlackBerry Unified Endpoint Manager 9
  - Deprecations in BES12/Good Control integration 14
  - Deprecations in standalone Good Control 15
- What was new in release Good Control 2.1.xx.yy 15
  - http APIs for managing KNOX domains 15
  - HTTP API to deactivate Good-for-KNOX 15
  - HTTP API to reset Android password 15
  - manage lists of iOS applications 16
  - New or changed SOAP API types, fields, and requests/responses in gc.wsdl 16
  - What was new in release GC 2.2.xx.yy 17
- About BlackBerry Dynamics software version numbers 19
- Contacting support for application development 19

Default permissions and web services requests for predefined roles 19

- About permissions for web services requests 20
- Specific permissions for Global Administrators role 20
- Specific permissions for Help Desk Administrators role 20
- Specific permissions for Service Accounts role 21

Basics of WSDL and SOAP 21
About SOAP-aware client software: use your favorite 22
Good Control SOAP: location, request syntax, responses, and errors 22
Location and other required schemas 22
endpoints for standalone Good Control SOAP requests 22
Request syntax 23
MIME type of request 23
Transaction security 23
Response syntax 24
Error types 24
Important notes about DeviceType 25
Example: adding a user to GC from an Active Directory domain 26
getdirectoryusersrequest 26
adduserrequest 27
Alphabetical list of operations in gc.wsdl 30
Alphabetical list of operations in cap.wsdl 34
HTTP API for Device Management 37
Intended Audience and Skills 37
How to Use The Documentation 37
Background on HTTP API Usage: Endpoint, Authorization, HTTP Verbs, and More 37
Resource Identification 37
Constructing a Request: Putting It Together 37
Authorization Header 38
Example Response: Retrieving Device Details 41
Validation and HTTP Error Responses 42
Requests by Cataegory: Device Management HTTP API 42
Configuration : MDM Configuration API 42
Device : Managed Devices API 43
EnterpriseResource : Enterprise Resource Management API 43

Reports : MDM server reports API 43
Policy : Policy Set's device rules and device policies API 44
Activation : MDM Activation API 44
Device Policy 44
Creating Policy 44
Updating Policy 44
Password Restrictions 45
Device Policy Details 46
Platform Specific Policies 46
Device Details 49
Platform Status 51
Enterprise Resource Configurations 52
How to read the tables 52
iOS Resource Configurations 53
WiFi 53
VPN 57
ActiveSync 62
Plist 63
WebClip 64
KNOX 64
WiFi 64
VPN 65
PSK 67
Certificate 67
ActiveSync 68
Credentials 71
Windows 71
WebLink 71
BlackBerry Dynamics documentation 72

Revision history

Good Control Web Services

Date	Description
2017-03-30	Correction to detail on base64 encoding in "Base64-Encoding Your Credentials" in Background on HTTP API Usage: Endpoint, Authorization, HTTP Verbs, and More . Proper syntax for the command requires the quoted **"password"** keyword.
2017-02-02	Corrections to Changed SOAP API types, fields, and operations in gc.wsdl and cap.wsdl
2017-01-31	Version numbers updated for latest release; no content changes.
2016-12-19	Version numbers updated for latest release; no content changes.
2016-07-08	Back by popular demand: • Alphabetical list of operations in gc.wsdl • Alphabetical list of operations in cap.wsdl
2016-07-01	Updated for latest release. See New or changed SOAP API types, fields, and requests/responses in gc.wsdl .
2016-03-28	Added Default permissions and web services requests for predefined roles .
2016-03-10	Truncated revision history to reduce bulk.
2016-02-25	Added Important notes about DeviceType .
2016-01-28	Clarification that namespace prefixes have changed in this release. See What's new in Good Control web services
2016-01-21	Updated for latest release: new requests for many new features of Good Control. See What's new in Good Control web services
2015-12-23	Version numbers updated for latest release; no content changes.
2015-12-16	Added new section MIME type of request and a clarification about the necessary schema in Transaction security .

Good Control/UEM web services

Good Control has a web services interface for programatically administering the GC system itself and for device management. There are two main groups of services.

- SOAP/WSDL-over-HTTPS for working with the Good Control System itself, users, policies, and so forth. The web services are based on SOAP (Simple Object Access Protocol) and WSDL (Web Services Definition Language) over HTTPS. This is a long-standing, popular programming paradigm that is familiar to many programmers. The GC WSDL related information is in Good Control SOAP: location, request syntax, responses, and errors .
- HTTP (or REST) API for working with device management. This is a more recent programming paradigm than SOAP. The HTTP API has functions for device policies, device configurations, and much more related exclusively to device management. These details are in HTTP API for Device Management .

Relation of UEM web services and Good Control web services

The web services operations on UEM and on Good Control are the same, but the setup of your client and other aspects are different. The focus of this guide is on the web services offered with Good Control.

UEM Getting Started Guide for Making Web Services calls

A tutorial specifically for using web services with UEM is highly recommended: UEM Getting Started Guide for Making Web Services calls.

What's new in Good Control web services

Highlighted here are some of the recent changes and improvements.

Global changes: new product names and abbreviations

This release includes major changes in naming to align former "Good Technology" products with BlackBerry.

Old name	New Name	Old abbreviation	New abbreviation
Good Control	None. This name refers to the standalone server that is not integrated with the BlackBerry Unified Endpoint Manager (BlackBerry UEM). When the server is integrated with the BlackBerry Unified Endpoint Manager, it is called BlackBerry Control.	GC	None
Good Proxy	None. This name refers to the standalone server that is not integrated with the BlackBerry Unified Endpoint Manager (BlackBerry UEM). When the server is integrated with the BlackBerry	GP	None

Old name	New Name	Old abbreviation	New abbreviation
	Unified Endpoint Manager, it is called BlackBerry Proxy.		
Good Dynamics	BlackBerry Dynamics	GD	None. We no longer abbreviate the name of this product line.
Good Dynamics Software Development Kit	BlackBerry Dynamics Software Development Kit	GD SDK	BlackBerry Dynamics SDK
Good Enterprise Mobility Server	BlackBerry Enterprise Mobility Server	GEMS	None. We no longer abbreviate the name of this product line.

Changed SOAP API types, fields, and operations in gc.wsdl and cap.wsdl

ContainerType status now includes failed activation

The complex type **`DeviceType`**, used by the **`GetDevices`** operation and others, now includes the **`ActivationFailed`** status for containers.

Updates to GetClientCertificateForUser and appDetailsForKeyStore types

The complex type **`GetClientCertificateForUser`** now includes **`certificateDefinitionID`**, which is the identifier for the certificate definition associated with the certificate.

The complex type **`appDetailsForKeyStore`** includes **`containerID`**and **`deviceSerialNumber`**.

New field from getDevices API

The response to the **`getDevices`** API now includes the following data:

- Manufacturer name

Field timeOutExpiry added to various APIs

The **`timeOutExpiry`** field has been added to the following operations. This field sets the time period after which the request is no longer valid.

- **`GenerateAccessKeys`**
- **`GenerateUnlockAccessKey`**

The **`pinExpiryTimeOut`** field has been added to the following operations. This field is the length of time that an access key is usable.

- **`BulkAddUsers`**
- **`BulkManageUsers`**

ContainerType new fields

The **`ContainerType`** object now includes the following new fields:

- Lock status
- Lock reason
- Authentication delegate

New CAP API to get organization id

The **`getOrganizationId`** API, which takes as input either the Enterprise ID or the Good Control serial number, returns the corresponding organization ID.

CAP API getAppDetails returns app categories

The response from the CAP API **`getAppDetails`** now includes information about the categories associated with an application.

Changes in web services in BlackBerry Unified Endpoint Manager

> **Note:** The details here apply only to the use of web services in BlackBerry Unified Endpoint Manager, that is the integrated BES12 and Good Control. The use of web services in standalone Good Control is not affected by these details.

In BlackBerry Unified Endpoint Manager (BES12 version 12.6 integrated mode), the following changes have been made to the GD SOAP APIs:

1. All GD SOAP APIs (GC SOAP and CAP SOAP) that are not listed in the following tables have been removed.
2. All GD MDM REST APIs have been removed.
3. GD APIs will be available on BES12 API port 18084 by default.
4. Port 18084 uses a different SSL certificate than the one that the GC server uses. API clients must trust the SSL certificate from BES12 API port 18084.
5. Integrated mode does not support GC 'policyset'; the BES12 policies replace that functionality. All GD APIs that managed the 'policyset' are impacted: some are no longer supported and others cannot honor 'policyset' related input/output information.
6. Integrated mode maps the GC 'application group' to the BES 'user group'. All GD APIs that managed the 'application group' are impacted: most are supported but are implemented using the BES12 'user group' and there are a few that are no longer supported.

7. APIs supported in integrated mode will not work with previously persisted entity Ids from a standalone GC server, for example userId, groupId, containerId.

Supported GC SOAP APIs

API Name	Behavior change
deleteContainer	None
generateAccessKeys	Integrated mode uses a new default email template for sending emails for access keys. Unless the administrator has updated the email template after upgrading to integrated mode, email messages might look different than the ones that GC sends in standalone mode. Any user Ids that the API client system previously persisted, do not work as the “userId” parameter value.
generateUnlockAccessKey	Any user Ids that the API client system previously persisted, do not work as the “userId” parameter value. Retrieve the “userId” for users using the “getUser’ API
getAccessKeys	Any user Ids that the API client system previously persisted, do not work as the “userId” parameter value. Retrieve the “userId” for users using the “getUser’ API
getActivatedContainers	Any user Ids that the API client system previously persisted, do not work as the “userId” parameter value. Retrieve the “userId” for users using the “getUser’ API
getAppInfo	The response does not include the “serverList” and “policySetId” elements. This information is not available or applicable in integrated mode.
getApps	The response does not include the “serverList” and “policySetId” elements. This information is not available or applicable in integrated mode.
getDevices	Any user Ids that the API client system previously persisted, do not work as the “userId” parameter value. Retrieve the “userId” for users using the “getUser’ API
getGPClusterList	None
getGPClusterServerList	None
getServerList	None
getTempUnlockPassword	None
getUnlockAccessKeys	None
getUser	The response does not include “policySetId” and “policyName”. The “appsGroupCount” element represents the number of BES user groups that the user belongs to.
lockContainer	None

API Name	Behavior change
removeAccessKey	None
sendPinEmail	Integrated mode uses a new default email template for sending email messages for access keys. Unless the administrator has updated the email template after upgrading to integrated mode, email messages might look different than the ones that GC sends in standalone mode

Supported CAP SOAP APIs

API Name	Behavior change
AddApp	The API adds a GD application entitlement with app_type as 'O' (organization), app_realm as 'E' (enterprise), app_visibility as 'PRV' (private), and client_type as 'NativeContainer'. The API displays an error if the request parameters do not match as listed previously. The API ignores the “purchase_url” request parameter
AddGroup	The API creates the BES user group. The request parameter “group_type” only supports the value of ‘e’ (Enterprise) and displays an error if you use any other values. The owner identifiers (“enterpriseId”/”organizationId”/”resellerId”) do not support accepting any value from the API client and display an error if you provide a value.
AddGroupsUsers	The API adds users to the BES user groups. Any group Ids and user Ids that the API client system previously persisted do not work as request parameter values. If you set the “replace” parameter to ‘true’, user(s) are removed from any other BES user groups and are added to newly specified BES user group
AddGroupUser	The API adds users to the BES user group. Any group Ids and user Ids that the API client system previously persisted do not work as request parameter values.
getApps	None
getGroupPermissions	The API returns a list of GD applications assigned to a BES user group with appropriate permission dispositions (ALLOW and DENY). Any group Ids that the API client system previously persisted do not work as request parameter values. Here are few caveats: • app_version is not returned in the response • only applications with client_type = NATIVE_CONTAINER are supported • organization info is excluded from the response • app_realm value can only be 'E' for enterprise • app_type is 'G' for any apps that start with "com.good" or "com.blackberry" and 'O' for all others

API Name	Behavior change
getGroups	The API returns the BES user groups filtered by group name. The “member_count” in the response indicates the number of BES users in the user group. The values supported for the “group_Type” parameter are: • E="Everyone" • e="All enterprise groups, except 'Everyone'" • null= "All Groups" The API displays an error if you provide an unsupported value for “group_Type” The owner identifiers “enterpriseId”/”organizationId”/”resellerId” will not be supported
getGroupsForUser	The API returns a list of BES user groups that a given BES user is part of. Any user Ids that the API client system previously persisted do not work as request parameter values.
getUsersInGroup	The API returns a list of BES users that are part of a given BES user group. Any group Ids that the API client system previously persisted do not work as request parameter values.
RemoveApp	The API removes the GD application entitlement from the system. The API displays errors if the removal fails because of BES12 rules; for example, the application is already assigned to a user/group
RemoveGroup	The API deletes the BES user group. Any group Ids that the API client system previously persisted do not work as “group_id” parameter value. The API performs checks and displays to be consistent with integrated mode. Some of these errors are new.
removeGroupUser	The API removes users from the BES user group. Any group Ids and user Ids that the API client system previously persisted do not work as request parameter values.
setGroupPermission	The API assigns GD applications with appropriate permission dispositions to a BES user group. Any group Ids that the API client system previously persisted do not work as request parameter values. Here are few caveats: • app_version_id is not supported in the request • app_id must be provided in the request because the default permissions are not supported in the integrated mode • only the following permission dispositions are supported: UNDEFINED, ALLOW, and DENY. UNDEFINED removes the application from the BES user group

New REST APIs

The exact syntax for these APIs is viewable at .

API Name	Behavior
Assign compliance/security policies to GD app	API allows assigning compliance and security policies to a GD application entitlement **Note:** The GC SOAP API 'changePolicyApp' is no longer available This API provides the required functionality
Assign policy to group	API allows assigning: • a policy to one or more user groups • one or more policies to a user group
Assign policy to user	API allows assigning: • a policy to one or more users • one or more policies to a user **Note:** The GC SOAP API 'changePolicyUser' is no longer available. This API provides the required functionality.
Create device activation password	API allows creating a device activation password for a user.
Create user	API creates a user with basic user attributes. The new user gets a self-service role in BES12 and is added to the "All Users" group. The new user also receives default policies. By default, the user is enabled for MDM service, but the API allows you to disable MDM service. **Note:** GC SOAP API 'addUser' is removed and this API provides the required functionality
Query activation email templates	API allows querying activation email templates.
Query policies	API allows querying policies by: • profile category • user id which returns all profiles assigned to user • group id which returns all profiles assigned to user group **Note:** The GC SOAP API 'getAllPolicies' is no longer available. This API provides the required functionality.
Query user groups	API allows querying user groups by: • group id

API Name	Behavior
	• group name • profile id which returns all user groups that the profile is assigned to • user id which is assigned to a user group
Replace policies for a group	API replaces all policies assigned to a user group with a new set of policies
Replace policies for a user	API replaces all policies assigned to a user with a new set of policies

Deprecations in BES12/Good Control integration

Change to BlackBerry service name in integrated BES12

When Good Control is upgraded to the BlackBerry Unified Endpoint Manager (also sometimes called "integrated BES12 mode"), the internal service name of "Good Control Server" is changed to "BlackBerry Control Server". Any customer using any tool that depends on this service must modify their tool in integrated mode to use the new internal name

Likewise, "Good Proxy" is referred to as "BlackBerry Proxy" in integrated mode.

App version level entitlement not supported

GD entitlement (allowed and disallowed GD apps) by GD application version is no longer supported. An app is either always allowed or always disallowed regardless of GD app version. This does not affect GD services, which can still be associated with app versions.

Note: Application version entitlement in the standalone Good Control is not affected by this change.

Connectivity profiles no longer hierarchical in BES12 integration

The hierarchical model that is supported in standalone Good Control is not supported in integrated mode with BES12. The ability to change the master connectivity profile and have it propagated to child profiles is no longer be available. The administrator is only be able to copy an existing connectivity profile.

Web services API deprecation and new replacement REST apis

The SOAP and HTTP APIs available in standalone Good Control are deprecated in integrated BES12 mode and have different behaviors Instead, REST API equivalents are available. SeeChanges in web services in BlackBerry Unified Endpoint Manager for more information.

Some of the latest standalone Good Control features not supported

- Automated GP upgrade
- New security policy for allowing no password
- New security plicy controlling client log upload

- Connectivity Profile export/import via CSV
- Other miscellaneous changes

Deprecations in standalone Good Control

Oracle supported only for upgrade, not new installation

The Oracle database is only supported in upgrades of current GC machines, includes installation on new node to an existing cluster.

Fresh installations for new customers and upgrading to BlackBerry Unified Endpoint Manager allow only SQL Server database.

What was new in release Good Control 2.1.xx.yy

http APIs for managing KNOX domains

- An enterprise wants to put custom keyboards and launchers on their Samsung devices. In order for them to work, admin needs to be able to specify these apps to be part of BlackBerry For KNOX “shared" domain. GC MDM API givesthe administrator the ability to specify them. All the admin needs is to get the certificate for the app and call the API with admin credentials.
- An enterprise also wants the choice of which apps should be the BlackBerry For KNOX “enterprise" domain. After all, BlackBerry For KNOX is for protecting their enterprise apps from all the other apps in the universe. The GC MDM API in this release allows the administrator to specify these apps. It has the same requirement. You need the package name and certificate. You need to call the GC MDM API to set it using admin credentials.

Syntax summary of new URIs

DELETE /mdm/config/knox-domain/{domain} => Delete all applications from KNOX domain

GET /mdm/config/knox-domain/{domain} => Get KNOX domain applications configuration

PUT /mdm/config/knox-domain/{domain} => Set KNOX domain applications configuration

HTTP API to deactivate Good-for-KNOX

This feature is the ability to turn off BlackBerry for KNOX.

In earlier releases, if an administrator turned off BlackBerry For KNOX in a Good Control device policy, it had no effect. With this release, all the application domains on the device are restored and BlackBerry For KNOX is disabled on the device.

See the action parameter on the Devices API POST /mdm/devices/{deviceID}/{action}.

HTTP API to reset Android password

You can now clear an Android device password from Good Control. In previous releases, the administrator had no ability to clear device password on Android devices. With this release, they now have this capability.

manage lists of iOS applications

Summary syntax of new URIs:

PUT /mdm/apps/ios/appstore => Sets managed applications list

GET /mdm/apps/ios/appstore => Gets managed applications list

New or changed SOAP API types, fields, and requests/responses in gc.wsdl

Most of the new or changed requests and associated types, enumerations and other definitions relate to new features in this release. These features are described in Good Control online help and other guides listed in BlackBerry Dynamics documentation .

This is only a high-level summary of changes. Consult the latest **`gc.wsdl`** file for exact details about these requests and associated types, enumerations and other definitions, only some of which are shown here.

New operations

- **`TestCAConnection`**: Make a test connection to the Certificate Authority (CA) server defined in **Certificate Definitions** in Good Control
- **VerifyCertificate**: Verify that the certificate for the CA server is still good (defined in **Certificate Definitions** in Good Control)
- **EnforceContainerAction**: Allows the caller to send block, unblock, or wipe actions to containers
- **UpdateUserAttributes**: Allows the caller to update the directory attributes of users (for instance, display name). This is normally done from AD via an AD Sync job and so is only needed for AD users if the AD Sync job is not running.
- **GetLoginConfig**: Provides some information about how the login process has been configured; for instance, if Kerberos Single Sign-On is configured.

Updates to existing operations

These are additional fields, changes to data types, or other miscellaneous changes.

- **`userId`** and **`policySetId`** were changed from a 32-bit integer to 64-bit integer. This has no backwards compatibility impact for existing clients because requests and responses still easily fit within the 32-bits.
- **`connectionProfileId`** added to **`User`**.

  Note: The **`connectionProfileId`** field is intended for possible use in the future. Do not use until that time.

- **PasswordPolicy**:
  - **`requirePasswordNotTouchIDPeriod`**: int
  - **`pwdFingerprint`**: boolean
  - **`allowFingerprintOnColdStart`**: boolean
  - **`requirePasswordNotFingerprintPeriod`**: int
- **`DeviceType`**: **`deviceHardware`**: string

- **JobType**: hostname: string
- **GenerateAccessKeysRequest**: **timeOutExpiry**: long
- **GenerateUnlockAccessKeyRequest**: **timeOutExpiry**: long
- **BulkAddUsersRequest**: **pinExpiryTimeout**: long
- **BulkManageUsersRequest**: **pinExpiryTimeout**: long
- **SetFeaturesRequest**:
  - **iosMdmAgent**: string
  - **androidMdmAgent**: string

What was new in release GC 2.2.xx.yy

Changes in SOAP namespace prefixes

Namespace prefixes in XML programming are a mechanism to provide a shorthand reference to schemas or other declarations and to prevent "name clashes": the same name inadvertently used for different objects from different namespaces.

Namespace prefixing for the Good Control SOAP APIs is handled by an underlying public SOAP library. The namespace prefixes are not defined in the GC's WSDL files; they are generated at runtime by this public library.

Note: The automatically generated namespace prefixes can change randomly from response to response.

Example

In the following XML document fragment:

- On the first line, **xmlns:ns6="urn:gc10.good.com"** defines the namespace prefix **ns6**.
- On the last line, **serialNumber** is the local name of the element, **n6** the namespace prefix, and **urn:gc10.good.com** the namespace of the element.

```
.
.
.
        <ns6:GetDevicesResponse xmlns:ns6="urn:gc10.good.com">
               <ns6:deviceList>
                 <ns6:serialNumber>ABCDEF1234</ns6:serialNumber>
.
.
.
```

The following suggestions can help guard against such changes in the future:

1. Use a namespace-aware XML parser and look for elements with namespace equal to **urn:gc10.good.com** and the local name of the element (in the example above, **serialNumber**).
2. Do not rely on namespace prefixes to remain the same from release to release or response to response.

New or changed SOAP API types, fields, and requests/responses in gc.wsdl

Most of the new or changed requests and associated types, enumerations and other definitions relate to new features in this release and are grouped by feature below. These features are described in *BlackBerry Device and Application Management*, Good Control online help, and other guides listed in BlackBerry Dynamics documentation .

This is only a high-level summary of changes. Consult the new gc.wsdl file for exact details about these requests and associated types, enumerations and other definitions, only some of which are shown here.

The names of the requests and types are in general self-explanatory.

Apps

- New DenyNativeAppVersionRequest and response
- New GetDeniedNativeAppVersionRequest and response
- New ReinstallAppRequest and response for managed apps
- New UpdateAllowedAppIdRequest and response are deprecated and will be removed in the next release.

Certificate management

- New AddCertificateDefinitionRequest and response
- New GetAllCertificateDefinitionRequest and response
- New GetCertificateDefinitionByIdRequest and response
- GetUsePkcs12CertificateManagementRequest and response renamed GetAllowUseClientCertificateRequest and response
- New RemoveCertificateDefinitionRequest and response
- New UpdateCertificateDefinitionRequest and response

Containers

- GetBulkContainerSummariesRequest and response

Device management

- Boolean isMDM added to Policy type
- New GetAllDeviceRulesRequest and response
- New GetDeviceRulesRequest and response
- New UpdateDeviceRulesRequest and response

Miscellaneous changes or additions

- Field lastActivityDate added to ContainerActivityType type
- DeleteMultipleOrOneJobRequest and response

About BlackBerry Dynamics software version numbers

The cover of this document shows the base or major version number of the product, but not the full, exact version number (which includes "point releases"), which can change over time while the major version number remains the same. The document, however, is always current with the latest release.

Product	Version
Good Control	3.0.56.79
Good Proxy	3.0.56.32

If in doubt about the exact version number of a product, check the BlackBerry Developer Network for the latest release.

Contacting support for application development

For assistance with difficulties developing a BlackBerry Dynamics-based application, there are several resources:

1. The developer support forums on the BlackBerry Developer Network at https://community.good.com/community/gdn/support
2. Use the MyAccount system at http://myaccount.blackberry.com

For the second option, include the following details:

- Platform: Android, iOS, macOS, Windows
- BlackBerry Dynamics SDK version number
- Name of IDE or other tools
- Any build logs that you think are useful to solve your difficulty

Default permissions and web services requests for predefined roles

Good Control creates the following predefined roles, which are granted specific rights. These roles have certain permissions to perform functions in the Good Control UI or with the GC web services.

- **Good Control Global Administrators** - Administrators with this role are granted the privilege to all functions, modify settings for all GC servers, and make changes to any user account. Additionally, these administrators can create, delete, and modify any other roles. The first Good Control Global Administrator is created from the Active Directory user specified during the installation of the first GC server in your server cluster.
- **Help Desk Administrators** - This role has limited access to GC data and functions. Administrators with this role are able only to view user account information, including application permissions, and to manage containers for all GC users. For example, they can delete, lock, or unlock any GD application for any GC user. Administrators with this role

can generate an access key for any user, but as a security measure, they are not allowed to view the entire access key. Instead, GC displays only the final five characters of the access key.

- **Service Accounts** - This role is for use by third-party server monitoring and reporting tools. These administrators can do all functions except role management.

About permissions for web services requests

Based on the specific permissions listed below for the various roles, you can correlate with the GC's SOAP request names or HTTP API names to determine if a role has the necessary permission to execute a particular request.

For example, the Help Desk Administrator role can execute the **`GenerateAccessKeysRequest`** but cannot execute the **`GetUsersRequest`** or **`GetPolicyDetailRequest`**.

Specific permissions for Global Administrators role

The following are the permissions for the Good Control Global Administrators role: all permissions.

The Global Administrator role can execute any of GC's web services requests.

- Users and Devices: All Access
  - Devices
- Entitlement Groups: All Access
- Container/Device Management: All Access
  - Create New Access Key
  - View Full Access Keys for All Users
- Policy Sets: All Access
  - Apple DEP Profiles
- Applications, Shared Services, and Application Wrapping: All Access
- Roles: All Access
- Server Configuration: All Access

Specific permissions for Help Desk Administrators role

The following are the permissions for the Help Desk Administrators role.

The Help Desk Administrator role can execute web services requests that relate to container and device management and user roles.

- Users and Devices: No Access
  - Devices
- Entitlement Groups: No Access

- Container/Device Management: All Access
  - Create New Access Key
  - View Full Access Keys for All Users
- Policy Sets: No Access
  - Apple DEP Profiles
- Applications, Shared Services, and Application Wrapping: No Access
- Roles: All Access
- Server Configuration: No Access

Specific permissions for Service Accounts role

The following are the permissions for the Service Accounts role: all permissions except roles.

The Service Account role can execute all web services requests except those related to roles.

- Users and Devices: All Access
  - Devices
- Entitlement Groups: All Access
- Container/Device Management: All Access
  - Create New Access Key
  - View Full Access Keys for All Users
- Policy Sets: All Access
  - Apple DEP Profiles
- Applications, Shared Services, and Application Wrapping: All Access
- Roles: No Access
- Server Configuration: All Access

Basics of WSDL and SOAP

If you are unfamiliar with web services, SOAP or WSDL, you should become familiar with the basics before reading farther. This guide includes only minimal tutorial information.

A wealth of information on the Internet is useful. Below are a few links:

- Wikipedia SOAP
- W3CSchools SOAP Tutorials
- W3C WSDL Specification
- O'Reilly Web Services, Chapter 6: WSDL Essentials

About SOAP-aware client software: use your favorite

To work with SOAP over HTTPS, you need a SOAP-aware client that can send requests, understand the SOAP semantics, and so on.

BlackBerry does not supply such client software. There are many, many different clients available (many free) on the Internet that you can use. To name only a few:

- SOAPUI
- Eclipse
- PHP add-on libraries
- curl
- Microsoft's PowerShell

In short, use the SOAP-aware client that you like best.

Good Control SOAP: location, request syntax, responses, and errors

Good Control includes a SOAP interface for administrative operations outside of the Good Control console. The GC and CAP WSDL files contain definitions of SOAP requests and their corresponding responses, including all fields, types, and error definitions.

Location and other required schemas

On every on-premise, installed Good Control server the **`gc.wsdl`** and **`cap.wsdl`** files are located as follows:

```
c:\good\docs\gc.wsdl
c:\good\docs\cap.wsdl
```

Otherwise, to get a copy of the files for your IDE, contact your BlackBerry representative.

The top of both files also define other required schemas.

Note: Do not alter the definitions in the WSDL files.

endpoints for standalone Good Control SOAP requests

The GC web services have two endpoints, depending on which of the WSDL files you are working with, either gc.wsdl or cap.wsdl.

Note: In the endpoints below, *localhost* is the fully qualified domain name of your GC server. Port 443 is implied by the use of the HTTPS protocol.

- gc.wsdl: **https://***localhost***/gc/services/GCService**
- cap.wsdl: **https://***localhost***/gc/soapproxy/cap**

Request syntax

In general, the request names follow the form:

***verbObject*Request**

where:

verb	is **Get**, **Add**, **Update** , **Delete**, **Remove**, and so on
Object	is one of GC's categories of administrative functions or focus, such as users, groups, roles, certificates, logs, and more.

Every request has its own unique fields (or elements) that are required or optional, as defined in the WSDL file. The field names are prefixed with the **<ns6:*fieldname*>** prefix.

MIME type of request

You should set the **Content-type** in the header of your HTTPS request to **text/html** or you can leave the **Content-type** header out altogether.

> **Note:** Do not set the MIME type to **application/xml**. This will result in an error.

Transaction security

The GC web services rely on the WS-Security (WSSE) schema for protection transactions with your GC administrator credentials. The WSSE security type is username/password protection.

The SOAP header of every request must include the inclusion of the WSSE schema and your username and password, as shown in the example below. Notice that your username must match the AD ***domain\username*** syntax:

```
  .
  .
  .
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-10" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>
          someDomain\someAdminUsername
        </wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordText">
          my.password
        </wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  .
```

.
.

Important: Make sure that you use the exact version of this schema:

docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd

Some IDEs might automatically select an older version or other schemas that appear similar but are not correct.

Response syntax

Responses for successful requests in general simply return a response body with the defined elements and values for the response. Every response has unique fields (or elements) that generally correspond to the fields on the request but are prefixed with the **<ns2:*fieldname*>** prefix.

Responses for requests that result in an error return a defined error message, as defined in the WSDL and listed in Error types

Error types

If a request results in an error, the system returns an error message in the body of the response. Here is an example of an error response:

```
Content-Type: application/xop+xml; charset=UTF-8; type="text/xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:B5B451F4DB81FB94A81407454300744@apache.org>

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<soapenv:Fault>
<faultcode>soapenv:Server</faultcode>
<faultstring>
com.good.gmc.roles.AuthenticateAndEnforce::
ACCESS_MODULE::INVALID_CREDENTIALS::
Invalid username and/or password.
</faultstring>
<detail>
 <gc:Fault xmlns:gc="urn:fault.gc10.good.com">
   <gc:faultCode>INVALID_CREDENTIALS</gc:faultCode>
   <gc:faultMessage>Invalid username and/or password.</gc:faultMessage>
  </gc:Fault>
</detail>
</soapenv:Fault></soapenv:Body></soapenv:Envelope>
--MIMEBoundaryurn_uuid_B5B451F4DB81FB94A81407454300743--
```

The error types are enumerated near the beginning of the gc.wsdl file and are in general self-explanatory:

- INVALID_CREDENTIALS
- INVALID_USER_OR_PASSWORD
- ERROR_INSUFFICIENT_RIGHTS
- OPERATION_NOT_ALLOWED_FOR_SELF_SERVICE
- ILLEGAL_PARAMETER_FOR_SELF_SERVICE
- INVALID_PARAMETERS
- DB_EXCEPTION
- DATA_TOO_LONG
- SERVICE_EXCEPTION
- DB_CONSTRAINT_VOILATION
- AUTH_DELEGATION_EXCEPTION
- APP_POLICY_OVERRIDE_EXCEPTION
- DIRECT_CONNECT_INFO_EXCEPTION
- OPERATION_NOT_ALLOWED
- INVALID_USERID
- APPLICATION_NOT_FOUND
- USER_NOT_FOUND
- USER_NOT_ENTITLED
- USER_ALREADY_EXISTS
- USER_ACCOUNT_LOCKED

Important notes about DeviceType

The SOAP API **`DeviceType`** complex type is used by **`GetDevicesRequest`** and other requests. Here are notes about how this type works.

> **Note:** The MDM HTTP API also includes a request that will return information about devices that are managed. See **`GET /mdm/devices`** in Device Details .

For **`DeviceType`**, the SOAP API attempts to gather details about the device via the BlackBerry Dynamics SDK and other sources on the device.

For phone number:

- On iOS, there is no mechanism to retrieve the data.
- On Android there is no reliable mechanism to retrieve the data.

For carrier info:

- On iOS if the device has a configured carrier network, **`DeviceType`** return its value; otherwise, it returns **`unknown`**.

- On Android, first an attempt to retrieve the SIM's operator name is made. If that is unsuccessful, an attempt is made to retrieve the network operator name (when the device is not roaming). If both attempts fail, the **DeviceType** returns null.

Example: adding a user to GC from an Active Directory domain

Here is an example of programming a common need for the GC administrator: adding a user from Active Directory without using the GC console.

Here we show the SOAP calls needed to add a user who already exists in the GC associated AD domains:

1. With **GetDirectoryUsersRequest**, we search the Active Directory for a user named "smith".
2. With **AddUserRequest**, we add that user to the GC.

getdirectoryusersrequest

We first need to search for a user. We invoke **GetDirectoryUsersRequest** to retrieve a list of users whose names match "smith", as specified in the **<searchString>** element:

```
POST https://localhost/gc/services/GCService HTTP/1.1
Content-Type: text/xml; charset=UTF-8
SOAPAction: "urn:gc10.good.com:gcServer:GetDirectoryUsersRequest"
User-Agent: Axis2
Host: localhost
Content-Length: 946

<?xml version="1.0" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-10" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>
          someDomain\someAdminUsername
        </wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordText">
          my.password
        </wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <urn:GetDirectoryUsersRequest xmlns:urn="urn:gc10.good.com">
      <urn:searchString>
        smith
      </urn:searchString>
    </urn:GetDirectoryUsersRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

The GC web service returns a response like this:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=UTF-8
Content-Length: 530
Date: Wed, 14 Mar 2012 16:44:07 GMT

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <urn:GetDirectoryUserResponse xmlns:urn="urn:gc10.good.com">
      <urn:users>
        <urn:displayName>
          John Smith
        </urn:displayName>
        <urn:sessionId>
          jsmith1@somecorp.com
        </urn:sessionId>
        <urn:domain>
          some.domain.com
        </urn:domain>
        <urn:firstName>
          John
        </urn:firstName>
        <urn:lastName>
          Smith
        </urn:lastName>
      </urn:users>
      <urn:isPartialResult>
        false
      </urn:isPartialResult>
    </urn:GetDirectoryUserResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

adduserrequest

We take the returned values and pass them to **`AddUserRequest`**. Essentially, we can take the fields and values returned by from **`GetUsersResponse`**, change the namespace from **`<urn:fieldname>`** to **`<urn:fieldname>`**, and pass the values verbatim to **`AddUserRequest`**:

```
POST https://localhost/gc/services/GCService HTTP/1.1
Content-Type: text/xml; charset=UTF-8
SOAPAction: "urn:gc10.good.com:gcServer:AddUserRequest"
User-Agent: Axis2
Host: localhost
Content-Length: 1283

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-10" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>
          someDomain\someAdminUsername
```

```
        </wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordText">
          my.password
        </wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <urn:AddUserRequest xmlns:urn="urn:gc10.good.com">
      <urn:user>
        <urn:displayName>
          John Smith
        </urn:displayName>
        <urn:sessionId>
          jsmith1@somecorp.com
        </urn:sessionId>
        <urn:domain>
          some.domain
        </urn:domain>
        <urn:firstName>
          John
        </urn:firstName>
        <urn:lastName>
          Smith
        </urn:lastName>
      </urn:user>
    </urn:AddUserRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

On success, the system responds like this:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=UTF-8
Content-Length: 755
Date: Wed, 14 Mar 2012 16:44:10 GMT

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <urn:AddUserResponse xmlns:urn="urn:gc10.good.com">
      <urn:user>
        <urn:userId>
          7733
        </urn:userId>
        <urn:displayName>
          John Smith
        </urn:displayName>
        <urn:sessionId>
          jsmith1@somecorp.com
        </urn:sessionId>
        <urn:domain>
          some.domain
        </urn:domain>
        <urn:firstName>
          John
        </urn:firstName>
```

```
        <urn:lastName>
          Smith
        </urn:lastName>
        <urn:status>
          1
        </urn:status>
      </urn:user>
    </urn:AddUserResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Alphabetical list of operations in gc.wsdl

This is an alphabetical list by name of the SOAP operations in **`C:\good\docs\gc.wsdl`**.

The name of a operation can give you an idea of its purpose, but be sure to consult the actual WSDL file for precise syntax and semantics.

AddAdGroupSync

AddAdministrator

AddAppPolicy

AddAppsToAllowedList

AddCertificate

AddCertificateDefinition

AddClientCertificate

AddConnectionProfile

AddGCServiceAdmin

AddOrUpdateApplicationServer

AddOrUpdateDomainServer

AddRoleMembers

AddServer

AddSplitBillingPkg

AddTrustedCertificate

AddUser

AssignSplitBillingPkg

AuditTrailExport

AuditTrailPurge

BulkAddUsers

BulkAddUsersFromGroup

BulkManageUsers

CancelUploadLogSchedule

ChangeGCServiceAdminPassword

ChangePolicyApp

ChangePolicyUser

ClearUploadLogMessages

CopyPolicy

CopyRole

DeleteAdGroupSync

DeleteAllowedAppId

DeleteAppPolicyById

DeleteConnectionProfile

DeleteContainer

DeleteGPCluster

DeleteMultipleOrOneJob

DeleteRole

DeleteSplitBillingPkg

DenyNativeAppVersion

DisableContainerLogging

EnableContainerLogging

EndSession

EnforceContainerAction

EnrollMDMDevice

ExportComplianceReport

ExportContainerReport

FetchDomains

FlattenConnectionProfiles

GenerateAccessKeys

GenerateRestrictedAccessKey

GenerateUnlockAccessKey

GetAccessKeys

GetActivatedContainers

GetAdGroupPreviewUsers

GetAdGroupSync

GetAdministrators
GetAdSyncBulkGroups
GetAllAdGroupSync
GetAllCertificateDefinition
GetAllClientCertificatesForUser
GetAllConnectionProfile
GetAllDeviceRules
GetAllowedApps
GetAllowV1Pins
GetAllPolicies
GetAllProvisionedContainers
GetAppInfo
GetAppPolicy
GetAppPolicyById
GetAppPolicyName
GetAppPolicyTemplate
GetApps
GetAppsWithPolicy
GetAppVersionOverride
GetBaseConnectionProfile
GetBulkAddUsersUpdate
GetBulkContainerSummaries
GetBulkManageUsersUpdate
GetBulkUsersConfig
GetBulkUsersUsers
GetCertificateDefinitionById
GetCertificates
GetClientCertificateDetailsById
GetConnectionProfile
GetConnectionProfileAndRules
GetConnectionProfileRules
GetContainerCommunicationProtocols
GetContainerEvents
GetContainerLockStatus
GetDashboardData
GetDeniedNativeAppVersion
GetDeploymentInfo
GetDetailedLogging
GetDeviceRules
GetDevices
GetDirectConnectInfo
GetDirectoryUsers
GetDomains
GetEffectiveRightsForUser
GetFeatures
GetGCProperties
GetGCRealmUser
GetGCReportsLimit
GetGPClusterList
GetGPClusterServerList
GetGroups
GetIntegrationState
GetJobById
GetJobs
GetLicenseSerial
GetLoginConfig
GetPolicyDetail
GetPolicyName
GetPublicCertificateInfo
GetRegistrationEmailInfo
GetRole
GetSelfServiceInfo

GetServerHudsonBuildInfo
getServerList
GetSessionInfo
GetSessionTimeout
GetSplitBillingPkg
GetTempUnlockPassword
GetTempUnlockType
GetThisServer
GetTrustedCertificates
getUnassignedServerList
GetUnlockAccessKey
GetUploadLogMessages
GetUploadLogSchedule
GetUseLowPorts
GetUsePkcs12CertificateManagement
GetUser
GetUserAppPolicies
GetUsers
GetWrappingEngineVersion
GetWrappingProperty
ListRoles
LockContainer
MakeDefaultPolicy
Noop
NotifyPolicyUpdates
PingSigningServer
ReinstallApp
RemoveAccessKey
RemoveAdministrator
RemoveApp
RemoveCertificate
RemoveCertificateDefinition
RemoveClientCertificate
RemovePolicySet
RemoveRoleMember
RemoveTrustedCertificate
RemoveUnlockAccessKey
RemoveUser
ResendWelcomeEmail
ResetTempPassword
SearchAdministrators
SendEnrollmentKeyEmail
SendPinEmail
ServerStatus
SetAdGroupPreviewUsers
SetAllowV1Pins
SetContainerCommunicationProtocols
SetDeploymentInfo
SetFeatures
SetGCProperties
SetMDMRequired
SetNewPassword
SetRegistrationEmailInfo
SetSelfServiceInfo
SetSessionTimeout
SetUploadLogSchedule
SetUseLowPorts
SetWrappingProperty
SignIccCertificate
TestCAConnection
TriggerAppPolicyDownload
UnAssignSplitBillingPkg

UnEnrollMDMDevice

UnregisterServer

UpdateAdGroupSync

UpdateAllowedAppId

UpdateApp

UpdateAppPolicies

UpdateAppPolicyById

UpdateAppVersionOverride

UpdateCertificate

UpdateCertificateDefinition

UpdateConnectionProfile

UpdateConnectionProfileRules

UpdateContainerManagementAppServer

UpdateDeviceRules

UpdateDirectConnectInfo

UpdateDomains

UpdateGPClusters

UpdatePolicyNameDesc

UpdatePolicySet

UpdatePolicySetConnectionProfile

UpdateRole

UpdateRoleRights

UpdateTrustedCertificate

UpdateUserAppPolicies

UpdateUserAttributes

UpdateWrappingEngineVersion

UploadClientLogMessage

VerifyCertificate

WrapApp

Alphabetical list of operations in cap.wsdl

This is an alphabetical list by name of the SOAP operations in **`C:\good\docs\cap.wsdl`**.

The name of an operation can give you an idea of its purpose, but be sure to consult the actual file for precise syntax and semantics.

addAdmin

addApp

addAppCategory

addAppService

addAppTag

addAppVersion

addAssociations

addBundle

addBundleVersionLocalesVersion

addCategory

addCategoryLocale

addEnterprise

addGroup

addGroupsUsers

addGroupUser

addOrganization

addResource

addResourceLinks

addResourceSets

addScreenshots

addService

addServiceVersion

addVersionLocale

createAppAndVersion

createAppVersion

editAppBinaryVersionMetaData

editAppBinaryVersionReleaseNotes

editAppIcon

editAppPlatformDescription

fetchAppMetaData

getAdmins

getAppCategories

getAppDetails

getAppDetailsByNativeVersion

getAppDownloadInfoForUser

getAppLocalAddress

getAppPermissions

getAppPolicy

getAppPolicyInfo

getAppPolicyVersionList

getAppPolicyVersionList"type="cap:EmptyType

getApps

getAppServices

getAppsProvidingService

getAppsPublishedToOrganization

getAppTags

getAppVersionAudience

getAppVersions

getAssociations

getBundles

getCategories

getCategoryLocales

getDeviceGroups

getDownloadAppsForUser

getEnterprise

getEnterprises
getEnterpriseServers
getEntitlementId
getGroupPermissions
getGroups
getGroupsForUser
getOrganizationId
getOrganizations
getPermissionDetails
getPublicAppDetails
getPublicApps
getPublicServiceVersions
getResellers
getResolvedPermissions
getResource
getResources
getResourceSets
getServiceDetails
getServices
getServiceVersionInterface
getServiceVersions
getUnassignedEnterprises
getUserPermissions
getUsers
getUsersForDownloadApp
getUsersInGroup
getUsersNotInGroup
getVersionLocales
importOrganization
isGDEnabledApplication
noop
parseBinary
publishApp
publishAppVersion
removeAdmin
removeApp
removeAppBinaryVersion
removeAppCategory
removeAppService
removeAppTag
removeAppVersion
removeAssociations
removeBundle
removeCategory
removeCategoryLocale
removeEnterprise
removeGroup
removeGroupUser
removeOrganization
removeResource
removeResourceLinks
removeResourceSet
removeService
removeServiceVersion
removeVersionLocale
setAppLocalAddress
setAppPolicy
setGroupPermission
setUserPermission
setUserPermissions
unpublishApp
unpublishAppVersion

updateAdmin

updateApp

updateAppMetaData

updateAppVersion

updateBundle

updateCategory

updateCategoryLocale

updateEnterprise

updateEnterpriseType

updateGroup

updateOrganization

updateResource

updateResourceSet

updateService

updateServiceVersion

updateVersionLocale

HTTP API for Device Management

Here are details on the HTTP API for device management via Good Control.

The device management HTTP API is not based on SOAP but on a different programming model that relies on the HTTP "verbs" (methods) GET, PUT, POST, and DELETE to pass requests that usually include a payload (content body) formatted in JSON (JavaScript Object Notation).

Included in this guide are essential details on set-up, such as endpoints, authentication, security, and basic usage. Exact syntax and request names are documented in the separate API reference for the HTTP API for device management.

Intended Audience and Skills

You should be familiar with HTTP methods and message bodies in JSON.

This is not a tutorial on general HTTP API programming; the document assumes that you are familiar with it.

How to Use The Documentation

1. Start with the syntax details in the downloadable zipfile of HTTP request/response documentation. This describes the request syntax and is the essential starting point for all developers.
2. The remainder of this guide details the various JSON-format request and response bodies and their fields. These fields and bodies are used with the requests detailed in the above.

Background on HTTP API Usage: Endpoint, Authorization, HTTP Verbs, and More

Resource Identification

Each API resource has an identifying URI.

That identifying URI will either use enclosing (owning/parent) object identifier or object own id.

Example:

- Device Rules belong to a Policy Set -> The HTTP API uses Policy Set ID to get device rules
- A device is identified by its own system wide unique ID.

Constructing a Request: Putting It Together

Here are details on how you to build your requests.

Endpoint for MDM API

Your requests must be sent to the following endpoint on your Good Control server:

https://*fully_qualified_domain_name_of_your_gc***/gc/rest-api/mdm/***desired_request*

where:

Part	Description
https://	You must use SSL.
fully_qualified_domain_name_of_your_gc	Is the fully qualified domain name of your Good Control server, like **gc.mycorporation.com** .
/gc/rest-api	Is the leading portion of the URI and is this exact literal string.
/*desired_request*	Is one of the defined MDM API requests described in the accompanying API reference and described below

Authorization Header

MDM HTTP API does not have its own authentication. It expects that GC server previously authenticated user successfully.

MDM HTTP API expects authentication result (token) in Authorization HTTP request header for all methods and for every request.

Value of Authorization header must be a Base64-encoded object of the following form:

```
{
       "userName" : "value",
       "password" : "value"
}
```

- **userName** must contain any form of what GC server considers a user login (Good Control currently uses "fullUsername" property of TokenInfo object).
- **password** must contain any form of what GC server considers a user password (Good Control uses text based token given by a GCserver when authentication succeeds)

After base64-encoding, the actual HTTP header looks like this:

Authorization: eyJ1c2VyTmFtZSIgOiAiZ2NVc2VyRG9tYWluXFxnY3N5c2FkbWluIiwgInBhc3N3b3JkIiA6ICJwYXNzd29yZCJ9

The standard GC authorization mechanism (call to AuthenticateAndEnforce) is used directly by MDM HTTP API permission-checking HTTP filter.

Almost each MDM API call has GC right associated with it.

Base64-Encoding Your Credentials

There are several ways you can base64-encode your credentials.

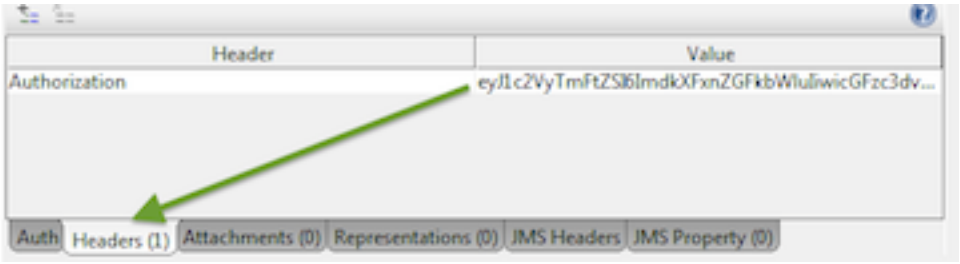
base64 command on Linux. On Linux systems or similar, such as macOS, you can encode right in the shell with the following pipe to the base64 command. In the following example, ***yourDomain***, ***yourGCloginName*** and ***yourPassword*** are all variables you supply, but **userName** and Pa55 are literals:

```
$ echo '
{"userName":"yourDomain\\yourGCloginName","password":"yourGCPassword"}' |
base64

eyJ1c2VyTmFtZSI6InlvdXJEb21haW5cXHlvdXJHQ2xvZ2luIiwieW91clBhc3N3b3JkIjoiUG
E1NSJ9Cg==
```

Online encoders. You can also use online encoders, such as https://www.base64decode.org/. However, because you are dealing with your own sensitive credentials, this is not recommended.

About Authentication in SOAPUI Client. In the SOAPUI client, do not use the Auth tab. Instead, use the **Headers** tab and the **Authorization** key to store the base64-encoded credentials:



Semantics of HTTP Verbs (Methods)

HTTP request methods define action semantics.

HTTP Verb (Method)	Meaning
POST	Create an object
GET	Query for data
PUT	Partial or full update of an object
DELETE	Delete an object

Note: PATCH is not currently used by the MDM HTTP API.

Content-type

Many requests, especially for POST and PUT methods, require a specific MIME type in the HTTP **Content-type** header to match the content in the body of the request.

Important: These required MIME types are listed in the API reference for every request that requires them.

The GET method does not include a request body, so no MIME type-is required.

For some operations with DELETE that require a a request body, the MIME can be set as text/plain:

```
Content-type: text/plain
```

Request Parameter Type

Depending on the **parameter type**, shown in the API reference for the request, you need to put your arguments in different locations:

Parameter Type	Meaning
path	Often used the POST, PUT, and DELETE methods, the parameter and its values must be put directly on the URI. Sometimes parameter type path takes a variable directly on the URI, this variable is indicated like **`{someVariableName}`**
body	Usually used with PUT and POST methods, the request's data must come in the body of the request, in JSON format
query	Usually a GET method, the request uses the QUERY STRING notation on the URI, like: `...?`*`parameter_name`*`=`*`value`*

Fully Formed Examples of Request to Retrieve Device IDs and Unenroll a Device

Assume we are running Good Control on a machine called **`goodcontrol.mycompany.com`**. Here are some fully formed examples of representative HTTP API requests showing this hostname.

The HTTP API reference follows a pattern that you need to interpret:

- A request consists of the HTTP method POST, GET, PUT, or DELETE.
- This followed by the URI (sometimes called a "route") for the request.
- The parameters and arguments must be added in the proper location, depending on the parameter type.

Retrieve Device IDs

Let's look at the request to obtain device IDs. In the API Reference, this request is listed as:

Note that the **`GET`** (or other HTTP method) is not included visibly as part the requests by is the HTTP METHOD needed to send the request to the server.

This request has a parameter type of **query** to retrieve the devices for a single user. This means the request can also look like this:

`GET /mdm/devices?user=`*`real_user_id`*

So, our fully formed request to retrieve device IDs for a single user looks like this.

Where?	What?
HTTP Header	Authorization: eyJ1c2VyTmFtZSIgOiAiZ2NVc2VyRG9tYWluXFxnY3N5c2FkbWluIiwgInBhc3N3b3JkIiA6ICJwYXNzd29yZCJ9
HTTP Header	No Content-type is needed with a GET.
HTTP Method	GET
Actual Request	https://goodcontrol.mycompany.com/gc/rest-api/mdm/devices?user=971249862098249724790

Unenroll a Device

Let's look at the API reference request to unenroll a device:

`DELETE /mdm/devices/{deviceId}` Unenroll device

This request has a parameter type of **path**, which means the argument comes on the URI itself. The argument is a variable device ID (an actual device ID), as indicated by the notation **`{deviceId}`**.

So after we retrieve the pertinent device ID, our fully formed request to unenroll it looks like this.

Where?	What?
HTTP Header	Authorization: eyJ1c2VyTmFtZSIgOiAiZ2NVc2VyRG9tYWluXFxnY3N5c2FkbWluIiwgInBhc3N3b3JkIiA6ICJwYXNzd29yZCJ9
HTTP Header	This request does not have a request body, so no Content-typeis needed. Other DELETE requests that have request bodies, and require Content-type: text/plain
HTTP Method	DELETE
Actual Request	https://goodcontrol.mycompany.com/gc/rest-api/mdm/devices/XYZZY12465DHWJWHJ

Example Response: Retrieving Device Details

The syntax of the request to retrieve details about devices belonging to a single user is described in Fully Formed Examples of Request to Retrieve Device IDs and Unenroll a Device .

This is the following request:

`GET https://`*`fully_qualified_domain_name_of_your_gc`***`/gc/rest-api/mdm/devices?user=`***`userID`**

The fields in the JSON response look like this. These fields are detailed in Device Details .

```
{
  "@class" : "com.good.gmc.api.model.device.DeviceDetails",
  "name" : "name",
  "uid" : "uid",
  "managementStatus" : {
    "lastSyncTime" : 1424814086524,
    "lastPushTime" : 1424814086524,
    "activationTime" : 1424814086524,
```

```
    "policyName" : "policy",
    "ownership" : "COMPANY"
  },
  "model" : "model",
  "platformStatus" : {
  },
  "hardware" : {
    "wifiMac" : "wifi-mac",
    "bluetoothMac" : "bluetooth-mac"
  },
  "integrity" : {
    "jailbroken" : false,
    "hasAppViolation" : false
  }
}
```

Validation and HTTP Error Responses

All inputs are validated with a sensible standard HTTP error response given back in case of failures:

- Not Found (404) is returned when object can't be accessed by a user.
- Bad Request (400) is returned when request can't be properly parsed.
- Unprocessable Entity (422) is returned when request is well formed by has validation problems with problem details stated in response body

Requests by Cataegory: Device Management HTTP API

This is a summary of available requests in the BlackBerry device management HTTP API. The notation is in the form:

HTTP_method URI => Description

> **Important:** Be sure to consult the full HTTP API Reference for precise syntax and semantics, which can vary by HTTP method and so on. For instance, query string arguments that might be required (especially for GET method) are not shown.

For interpretation of the API Reference syntax, such as variable notation like {reportType}, see Background on HTTP API Usage: Endpoint, Authorization, HTTP Verbs, and More .

Configuration : MDM Configuration API

POST /mdm/config/endorse-apns-csr => Generate endorsed APNS CSR

GET /mdm/config/apns-certificate => Get APNS certificate

PUT /mdm/config/apns-certificate => Set APNS certificate

GET /mdm/config/gcm-config => Get Google Cloud Messaging Configuration

PUT /mdm/config/gcm-config => Set Google Cloud Messaging Configuration

GET /mdm/config/elm-key => Get ELM license key

PUT /mdm/config/elm-key => Set ELM license key

GET /mdm/config/klm-key => Get KLM license key

PUT /mdm/config/klm-key => Set KLM license key

GET /mdm/config/app-compliance-list/{listType} => Get applications compliance list

POST /mdm/config/app-compliance-list/{listType} => Add applications to compliance list

DELETE /mdm/config/app-compliance-list/{listType} => Deletes applications from compliance list

DELETE /mdm/config/knox-domain/{domain} => Delete applications from KNOX domain

DELETE /mdm/config/knox-domain/{domain} => Delete all applications from KNOX domain

GET /mdm/config/knox-domain/{domain} => Get KNOX domain applications configuration

PUT /mdm/config/knox-domain/{domain} => Set KNOX domain applications configuration

Device : Managed Devices API

PUT /mdm/devices/{deviceId}/action/reinstall-app/{bundleId} => Reinstalls managed application on a device

GET /mdm/devices/ => Get all user devices

DELETE /mdm/devices/{deviceId} =>Unenroll device

GET /mdm/devices/{deviceId} => Get device by id

POST /mdm/devices/{deviceId}/{action} => Performs action on a device: reset password, lock, wipe, deactivate device

EnterpriseResource : Enterprise Resource Management API

GET /mdm/er/{platform}/{resourceType}/ => Get all enterprise resource by platform and type

POST /mdm/er/{platform}/{resourceType}/ =>Create enterprise resource

GET /mdm/er/{platform}/{resourceType}/{id} =>Get enterprise resource by id

PUT /mdm/er/{platform}/{resourceType}/{id} =>Update enterprise resource

DELETE /mdm/er/{platform}/{resourceType}/{id} =>Delete enterprise resource

Reports : MDM server reports API

DELETE /mdm/reports/{reportType}/schedule => Delete report schedule

POST /mdm/reports/{reportType}/schedule => Set report schedule

GET /mdm/reports/{reportType}/schedule => Get report schedule

GET /mdm/reports/{reportType} => Get report

GET /mdm/reports/{reportType}/lastrun => Get last report execution time

Policy : Policy Set's device rules and device policies API

PUT /mdm/rules/{policySetId} => Update device rules

GET /mdm/rules/{policySetId} => Get device rules

POST /mdm/policy => Create device policy

GET /mdm/policy => Get all device policies

PUT /mdm/policy/{policyId} => Update device policy

DELETE /mdm/policy/{policyId} => Delete device policy

GET /mdm/policy/{policyId} => Get device policy

GET /mdm/policy/{policyId}/{platform} => Get platform policy

PUT /mdm/policy/{policyId}/{platform} => Update platform policy

Activation : MDM Activation API

POST /mdm/ios-co-enrollment-url/{userId} => Generate enrollment URL for company owned devices

DELETE /mdm/activation/{userId} => Delete some activation codes

DELETE /mdm/activation/{userId} => Delete all activation codes

GET /mdm/activation/{userId} => Get activation codes

POST /mdm/activation/{userId}/{type} => Generate activation cod

Device Policy

Device Policy structure is used in policy create (**`POST`**) and update (**`PUT`**) operations.

Creating Policy

When creating policy only name property needs to be provided, all other values will be populated from system defaults.

Example:

```
{
        "name" : "my-policy",
        "description" : "To be used by my department"
}
```

Updating Policy

When updating policy all supported properties can be changed.

Application compliance mode sets how tenant's applications compliance list should be evaluated for devices getting the policy.

Supported values are:

- **BLACKLIST**
- **WHITELIST**
- **DISABLED**

```
{
        "name" : "name",
        "description" : "description",
        "passwordRestrictions" : {
        },
        "applicationComplianceMode" : "DISABLED"
}
```

Password Restrictions

Password restrictions are requirements Device Management policy imposes for all platforms applicable to device PIN/passcode.

```
{
        "quality" : {
        },
        "minLength" : 8,
        "age" : 5,
        "historyDepth" : 5,
        "maxFailedAttempts" : 3,
        "inactivityTimeout" : 5,
        "maxGracePeriod" : 100,
        "minMutations" : 2,
        "maxSequentialChars" : 2,
        "passwordRequired" : true
}
```

In addition to various passcode properties it's possible to set required quality of the password.

Password Quality

Three types of password quality are currently supported:

- Alphanumeric

```
"quality" : {
        "alphanumeric" : { }
}
```

- Simple

```
"quality" : {
       "simple" : {
                "type" : "ANY"
       }
}
```

Type property sets subtype of simple password quality, supported values are: ANY, NUMERIC, ALPHABETIC

- Complex

Complex passwords have their own set of properties allowing fine tuning of required password complexity

```
"quality" : {
      "complex" : {
        "minSymbolsRequired" : 1,
        "minDigitsRequired" : 1,
        "minLowercaseLettersRequired" : 1,
        "minUppercaseLettersRequired" : 1,
        "minLettersRequired" : 1,
        "minNonLettersRequired" : 1,
        "minPasswordComplexCharacters" : 3
      }
}
```

Device Policy Details

Fetching Specific Policy

When fetching specific policy, Policy Details will be returned.

It adds the property **`deviceCount`** to the Policy item that contains a number of managed devices associated with the policy.

Fetching All Policies

In case when specific policy id is not provided to get policy method all tenant's device policies will be returned.

Returned Policy Details will have no password restrictions information in them but will contain all other PolicyDetails properties:

name, description, applicationComplianceMode and deviceCount.

Platform Specific Policies

Platform specific policy consists of platform specific device restrictions, enterprise resource names and device permissions Device Management will require to be granted by user.

```
{
       "deviceRestrictions" : {
       },
       "enterpriseResources" : ["my-wifi", "my-vpn"],
       "devicePermissions" : ["AllowEraseDevice", "AllowDeviceLockAndPasscodeRemoval"]
}
```

Currently there are no enterprise resource types supported on vanilla Android platform

Currently supported device permissions:

iOS	Android	Permission
Supports		AllowInspectInstalledConfigurationProfile
Supports		AllowInstallAndRemoveConfigurationProfile
Supports	Supports	AllowDeviceLockAndPasscodeRemoval
Supports	Supports	AllowEraseDevice
Supports		AllowQueryDeviceInformation
Supports	Supports	AllowQueryNetworkInformation
Supports		AllowInspectInstalledProvisioningProfile
Supports		AllowInstallAndRemoveProvisioningProfile
Supports	Supports	AllowInspectInstalledApplication
Supports	Supports	AllowRestrictionRelatedQuery
Supports	Supports	AllowSecurityRelatedQuery
Supports		AllowManipulateSettings
Supports	Supports	AllowAppManagement

There are no permissions currently defined for KNOX and Windows platforms.

iOS Platform Policy

Supported iOS device restrictions object example:

```
"deviceRestrictions" : {
    "@class":"com.good.gmc.api.model.policy.IosRestrictions",
    "allowAddingGameCenterFriends" : false,
    "allowAppInstallation" : false,
    "allowAssistant" : false,
    "allowAssistantWhileLocked" : false,
    "allowBookstoreErotica" : false,
    "allowCamera" : false,
    "allowCloudBackup" : false,
    "allowCloudDocumentSync" : false,
    "allowCloudKeychainSync" : false,
    "allowDiagnosticSubmission" : false,
    "allowExplicitContent" : false,
    "allowFingerprintForUnlock" : false,
    "allowGlobalBackgroundFetchWhenRoaming" : false,
    "allowInAppPurchases" : false,
    "allowLockScreenControlCenter" : false,
    "allowLockScreenNotificationsView" : false,
    "allowLockScreenTodayView" : false,
```

```
    "allowMultiplayerGaming" : false,
    "allowOpenFromManagedToUnmanaged" : false,
    "allowOpenFromUnmanagedToManaged" : false,
    "allowOtaPkiUpdates" : false,
    "allowPassbookWhileLocked" : false,
    "allowPhotoStream" : false,
    "allowSafari" : false,
    "allowScreenShot" : false,
    "allowSharedStream" : false,
    "allowUntrustedTlsPrompt" : false,
    "allowVideoConferencing" : false,
    "allowVoiceDialing" : false,
    "allowYoutube" : false,
    "allowItunes" : false,
    "forceAssistantProfanityFilter" : false,
    "forceEncryptedBackup" : false,
    "forceItunesStorePasswordEntry" : false,
    "forceLimitAdTracking" : false,
    "ratingApps" : null,
    "ratingMovies" : null,
    "ratingRegion" : null,
    "ratingTvShows" : null,
    "safariAcceptCookies" : null,
    "safariAllowAutoFillEnable" : false,
    "safariAllowJavascriptEnable" : false,
    "safariAllowPopupsEnable" : false,
    "safariForceFraudWarningEnable" : false,
    "forceAirplayOutgoingRequestsPairingPasswordEnable" : false,
    "forceAirplayIncomingRequestsPairingPasswordEnable" : false,
    "allowManagedAppsCloudSync" : false,
    "allowActivityContinuation" : false,
    "allowEnterpriseBookBackup" : false,
    "allowEnterpriseBookMetadataSync" : false,
    "allowSpotlightInternetResults" : false
}
```

Android Device Restrictions

Supported Android device restrictions example:

```
"deviceRestrictions" : {
 "@class":"com.good.gmc.api.model.policy.AndroidRestrictions",
 "disableCamera" : false,
 "encryptInternalStorage" : false
}
```

KNOX Device Restrictions

Supported KNOX device restrictions example:

```
"deviceRestrictions" : {
    "@class":"com.good.gmc.api.model.policy.KnoxRestrictions",
```

```
    "encryptSDCard" : false,
    "disableSMS" : false,
    "disableMMS" : false,
    "disableSVoice" : false,
    "disableSDCard" : false,
    "disableNFC" : false,
    "disableAndroidBeam" : false,
    "disableCellularData" : false,
    "disableFactoryReset" : false,
    "disableNativeBrowser" : false,
    "disableNoticeAndConsentBanner" : false,
    "disableRoamingData" : false,
    "disableRoamingSync" : false,
    "disableRoamingVoiceCall" : false,
    "disableScreenCapture" : false,
    "disableLockScreenShortcuts" : false,
    "disableLockScreenWidgets" : false,
    "disableWiFi" : false,
    "disableWiFiAutoConnect" : false,
    "disableBluetooth" : false,
    "disableGooglePlay" : false,
    "disableNonMarketApp" : false,
    "disableOTAOSUpdate" : false,
    "disableUsbDebugging" : false,
    "disableUsbMediaPlayer" : false,
    "disableUsbHostStorage" : false,
    "disableBluetoothTethering" : false,
    "disableUsbTethering" : false,
    "disableWiFiTethering" : false,
    "enableCommonCriteriaMode" : false,
    "disableYouTube" : false,
    "attestationEnabled" : false,
    "attestationFrequency" : 0,
    "knoxPremiumEnabled" : false
```

Windows Device Restrictions

Supported Windows device restrictions example:

```
"deviceRestrictions" : {
    "@class":"com.good.gmc.api.model.policy.WindowsRestrictions",
       "userAccountControlStatus" : "ALWAYS_NOTIFY",
       "allowDataWhileRoaming" : false,
       "allowDiagnosticSubmission" : false,
       "allowMSAccountOptionalForModernApp" : false,
       "requireSmartScreenInIE" : false
}
```

Device Details

Device details are returned whenever device is queried using GC's Managed Device API.

Depending on underlying device type the response could be a generic set of details a or a more specialized descendant of it.

Example of Device Details JSON:

```
{
  "@class" : "com.good.gmc.api.model.device.DeviceDetails",
  "name" : "name",
  "uid" : "uid",
  "managementStatus" : {
    "lastSyncTime" : 1424814086524,
    "lastPushTime" : 1424814086524,
    "activationTime" : 1424814086524,
    "policyName" : "policy",
    "ownership" : "COMPANY"
  },
  "model" : "model",
  "platformStatus" : {
  },
  "hardware" : {
    "wifiMac" : "wifi-mac",
    "bluetoothMac" : "bluetooth-mac"
  },
  "integrity" : {
    "jailbroken" : false,
    "hasAppViolation" : false
  }
}
```

PhoneDetails currently adds only 2 properties: phoneNumber and imei.

```
{
  "@class" : "com.good.gmc.api.model.device.PhoneDetails",
  "name" : "name",
  "uid" : "uid",
  "managementStatus" : {
    "lastSyncTime" : 1424814238107,
    "lastPushTime" : 1424814238129,
    "activationTime" : 1424814238129,
    "policyName" : "policy",
    "ownership" : "COMPANY"
  },
  "model" : "model",
  "platformStatus" : {
  },
  "hardware" : {
    "wifiMac" : "wifi-mac",
    "bluetoothMac" : "bluetooth-mac"
  },
  "integrity" : {
    "jailbroken" : false,
    "hasAppViolation" : false
  },
  "phoneNumber" : "123",
  "imei" : "456"
```

```
}
```

Platform Status

Platform status is a set of device properties specific to a platform.

In addition to primitive device properties information about installed applications will be included.

Example of platform status:

```
{
  "@class" : "com.good.gmc.api.model.device.PlatformStatus",
  "osVersion" : "8",
  "platformId" : "platformid",
  "apps" : [ {
  } ],
}
```

Application Details

"apps" property of platform status will contain JSON array with a semantic of set of AppDetails items:

```
[ {
    "@class" : "com.good.gmc.api.model.device.AppDetails",
    "name" : "name",
    "version" : "ver",
    "managed" : true,
    "violation" : false
  } ]
```

Specialized versions of platform status are available for KNOX and Windows.

KNOX Platform Status

KNOX platform status contains many KNOX (including SAFE) device specific properties:

```
{
    "@class" : "com.good.gmc.api.model.device.KnoxPlatformStatus",
    "osVersion" : "8",
    "platformId" : "platformid",
    "apps" : [ {
    } ],
    "safeEnabled" : true,
    "knoxVersion" : "1",
    "goodForKnoxEnabled" : false,
    "attestationFailed" : false,
    "lastAttestationTime" : 1424818030513
}
```

Additionally KNOX platform status has specialized version of application details object - KnoxAppDetails.

It adds "domain" property to AppDetails that refers to KNOX domain name of the application.

```
[ {
      "@class" : "com.good.gmc.api.model.device.KnoxAppDetails",
```

```
      "name" : "name",
      "version" : "ver",
      "managed" : true,
      "violation" : false,
      "domain" : "dom"
    } ]
```

Windows Platform Status

Windows platform status currently has no support for apps, so apps property will always be reported empty.

Example:

```
{
    "@class" : "com.good.gmc.api.model.device.WindowsPlatformStatus",
    "osVersion" : "8",
    "platformId" : "platformid",
    "apps" : [ {
    } ],
    "windowsUpdateStatus" : "Auto",
    "antiVirusStatus" : "Good",
    "antiVirusSignatureStatus" : "Expired",
    "firewallStatus" : "Good",
    "manufacturer" : "manuf",
    "wifiEnabled" : true,
    "bluetoothEnabled" : true,
    "encryptionRequired" : true,
    "dataWhileRoamingEnabled" : true,
    "antivirusEnabled" : true,
    "firewallEnabled" : true
  }
```

Enterprise Resource Configurations

Currently BlackBerry Dynamics MDM support the following enterprise resource configurations:

Platform	Resource Types
iOS	WiFi, VPN, ActiveSync(Exchange), Plist, Credentials, WebClip
KNOX	WiFi, VPN, ActiveSync(Exchange), Credentials
Windows	WebLink

How to read the tables

- Each field is a JSON attribute.
- The field names use ":" to indicate that the right side of the ":" is a sub-attribute of the left side. For example: a field of "enterprise: domain" is represented in JSON as "enterprise" : { "domain": {...} }.

- In addition to using ":" , sub-attributes are also indented, to visually aid with understanding the structure.
- Some fields can take parameter, and later on the value will be replaced with **user property** value. A parameter is a string begin and end with '%'. For example '%user_name%' is a parameter, and there need to be a property 'user_name' associated with the user. User properties will not affect profiles that have been sent to devices, and updating user properties will not automatically update profiles that are installed on devices. Fields with "Can be parameter" set to yes can take parameter as value.

iOS Resource Configurations

WiFi

	Field	Required	Type	Values	Can be parameter	Description
01	**ssid**	Mandatory	string	Max length 512		WiFi network name (SSID) that device should connect Ignored for Hotspot 2.0
02	**hidden**	Optional	boolean	true, false		Indicates if the configured SSID is not broadcasting. Having this information allows iOS to search for this SSID in a different way. false by default
03	**autoJoin**	Optional	boolean	true, false		Indicates whether the device should automatically connect to this SSID, when it is found. true by default
04	**securityConfig**	Mandatory	JSON Object	{ . . . } Must be only one of the following child		Object to configure the WiFI security settings

	Field	Required	Type	Values	Can be parameter	Description
				objects		
05	*securityConfig*: **password**	Optional	JSON Object	{ . . . }		This presents PSK-based networks (Pre-shared key).
06	*securityConfig*: *password*: **type**	Mandatory	string	WEP, WPA, ANY		WiFi standard to use
07	securityConfig: *password*: **password**	Mandatory	string	Max length of 512.	Yes	Pre-Shared key (password) used by all devices to connect to the network.
08	*securityConfig*:**unsecured**	Optional	Empty JSON object	{ }		Indicates an open SSID network with no security
09	*securityConfig:* **enterprise**	Optional	JSON Object	{ . . . }		This object represents 802.1X networks (WPA2 Enterprise)
10	*securityConfig: enterprise*:**eapConfig**		JSON Object	{ . . . }		EAP configuration
11	*securityConfig: enterprise: eapConfig*:**eap**	Min one element	JSON Array	*List of:* TLS, TTLS, PEAP, LEAP, EAP_ FAST, EAP_ SIM, EAP_ AKA		802.1X EAP methods
12	*securityConfig: enterprise: eapConfig*:**userName**	Optional	string	Max length of 512.	Yes	
13	*securityConfig: enterprise: eapConfig*:**password**	Optional	string	Max length of 512	Yes	Set on device if omitted.
14	*securityConfig: enterprise: eapConfig:* **useOneTimePassword**	Optional	boolean	true, false		false by default. If set to true, the "password" attribute will be ignored.
15	*securityConfig: enterprise: eapConfig*:**identity**	Optional	JSON Object	See Credentials section		Mandatory for TLS.
16	*securityConfig: enterprise: eapConfig*:**outerIdentity**	Optional	string	Max length of 512		External identity used to protect the real identity of the user
17	*securityConfig: enterprise: eapConfig:*	Optional	string	PAP, CHAP, MSCHAP, MSCHAPv2		Mandatory if TTLS is one of the EAP types

	Field	Required	Type	Values	Can be parameter	Description
	TTLSInnerIdentity					
18	*securityConfig: enterprise:***trustConfig**	Optional	JSON Object	{ . . . }		Trust configuration that describes what certificate authorities / certificates can be trusted to make the WiFi connection
19	*securityConfig: enterprise: trustConfig:* **trustedServerNames**	Optional	JSON Array	Array of strings		Each string representing DNS or CN (Common Name)
20	*securityConfig: enterprise: trustConfig:* **trustedCertificates**	Optional	JSON Array	Array of Credentials objects		Each Credentials object represents Root/Intermediate certificates that the device should trust.
21	*securityConfig: enterprise: trustConfig:* **allowTrustExceptions**	Optional	boolean	true, false		true by default
22	*securityConfig: enterprise:***eapFastConfig**	Optional	JSON Object	{ . . . }		Configuration unique to EAP_FAST EAP method. If EAP_FAST EAP method is not used, this attribute is not needed.
23	*securityConfig: enterprise: eapFastConfig:***usePAC**	Optional	boolean	true, false		Indicates whether the device should use Protected Access Credentials (PAC) false by default
24	*securityConfig: enterprise: eapFastConfig:* **provisionPAC**	Optional	boolean	true, false		false by default
25	*securityConfig: enterprise: eapFastConfig:* **provisionPACAnonymously**	Optional	boolean	true, false		false by default
26	proxyConfig	Optional	JSON Array	See Proxy Config section		Assumed to be no proxy, if this property is missing.
27	hotspotConfig	Optional	JSON Array	See Hotspot Config section		Standard network is assumed if this property is missing.

Hotspot Config (for WiFi)

	Field	Required	Type	Values	Description
01	**hotspotConfig**	Optional	JSON Object	Must be only one of the following child objects	Object to configure Hotspots.
02	*hotspotConfig*: **legacy**	Optional	Empty JSON Object	{ }	Specifies that WiFi network is an legacy hotspot SSID. There is no further configuration needed.
03	*hotspotConfig*: **passpoint**	Optional	JSON Object	{ . . . }	New Hotspot 2.0 (Also known as Passpoint) to allow easy connection to service
04	*hotspotConfig*: *passpoint*: **domainName**	Mandatory	string	Max length of 512	Domain name used by the Hotspot 2.0 network
05	*hotspotConfig*: *passpoint*: **enableRoaming**	Optional	boolean	true, false	false by default
06	*hotspotConfig*: *passpoint*: **roamingProviders**	Optional	JSON Array of strings	[. . .] Each element max length is 512	Roaming partners associated with the service.
07	*hotspotConfig*: *passpoint*: **networkAccessRealms**	Optional	JSON Array of strings Each element max length is 512	[. . .] Each element max length is 512	NAIs used to authenticate users.
08	*hotspotConfig*: *passpoint*: **mccPlusMncs**	Optional	JSON Array of strings Each element max length is 512	[. . .] Each element max length is 512	MCC and MNC of the operators providing the WiFi server
09	*hotspotConfig*: *passpoint*: **operatorName**	Mandatory	string	Max length of 512	Operator Name that is displayed by the WiFi network

Proxy Config (for WiFi and VPN Config)

N	Field	Required	Type	Values	Description
01	**proxyConfig**	Optional	JSON object	Must be only one of the following	Proxy Configuration applies to VPN

N	Field	Required	Type	Values	Description
				child objects	and WiFi resource configurations. Must be one of the following child objects.
02	proxyConfig: **automatic**	Optional	JSON Object	{ . . . }	Configuration of automatic proxy.
03	proxyConfig: automatic: **configUrl**	Mandatory	string	Max length 512	URL to use for configuring automatic proxy settings
04	proxyConfig: automatic: **enableFallbackToDirectConnection**	Optional	boolean	true, false	If true, the client will use direct connect if proxy is not available
05	proxyConfig: **manual**	Optional	JSON object	{ . . . }	Configuration of manual proxy.
06	proxyConfig: manual: **host**	Mandatory	string	Max size 512	DNS or IP address of the host
07	proxyConfig: manual: **port**	Mandatory	number	Positive number	Port number to access proxy
08	proxyConfig: manual: **username**	Optional	string	Max length 512	user name used to connect to the proxy
09	proxyConfig: manual: **password**	Optional	string	Max length 512	password used to authenticate with the proxy

VPN

	Field	Required	Type	Values	Description
01	**name**	Mandatory	string	Max length 512	Name unique within typeConfig within a specific enterprise.
02	**typeConfig**	Mandatory	JSON Object	{ . . . }	Configuration of different types of VPN clients that can be configured

	Field	Required	Type	Values	Description
03	*typeConfig*: **<vT>**	Mandatory	JSON Object	See Type Config section.	<vT> represents one of the following keys: L2TP, PPTP, IPSec, CiscoAnyConnect, Juniper, F5, SonicWALL, ArubaVIA, or CustomSSL.
04	**proxyConfig**	Optional	JSON Object	{ . . . }	See Proxy Config section.
05	**useForAllTraffic**	Optional	boolean	true, false	true â€“ VPN will be used for all traffic on the device. Default value is false.
06	**vendorConfig**	Optional	JSON Object	{ . . . }	This JSON object is a key:value pairs needed to configure Custom SSL vendor client.
07	*vendorConfig* : **<key>**	Optional	string	Max length 512	<key> is arbitrary string, representing a Custom SSL vendor property. The key represents a custom property that takes a string value.

Type Config (for VPN Config)

	Field	Required	Type	Values	Can be parameter	Description
01	**hostName**	Mandatory	string	Max length 512		IP address or DNS for the VPN server host
02	**userName**	Optional	string	Max length 512	Yes	User name used for authentication
03	**authType**	Mandatory	JSONObject	{ . . . }		One of the following auth type is mandatory
04	*authType:* **password**	Optional	JSONObject	{ . . . }		Example: "PPTP": { "password": { "password": "my-password" }}
05	*authType: password:* **password**	Optional	string	Max length 512	Yes	
06	*authType* **:certificate**	Optional	JSONo	{ . . . }		Example: "ArubaVIA": { "authType" : { "certificate" : { "identity" : { "name" : "certName",

	Field	Required	Type	Values	Can be parameter	Description
			bject			"fileName" : "certFileName.p12", "content": "base64 encoded PKCS#12 format keystore", "password": "pAssWorD" }}}}
07	*authType: certificate:* **identity**	Mandatory	JSONobject	See Credentials Section		
08	*authType: certificate:* **requirePinDuringConnection**	Optional	boolean	true, false		Applies only to IPSec. Defaults to false for IPSec VPN.
09	*authType:* **password+certificate**	Optional	JSONobject	{...}		Applies only to F5 or CustomVPN. Example: "F5" : { "authType" : { "password+certificate": { "password": "my-password", "identity" : { "name" : "certName", "fileName" : "certFileName.p12", "content": "base64 encoded PKCS#12 format keystore", "password": "pAssWorD" }}}}
10	*authType: password+certificate:* **password**	Optional	string	Max length 512	Yes	Password used to authenticate the user
11	*authType: password+certificate:* **identity**	Mandatory	JSONObject	See Credentials Section		Identity certificate used to authenticate the user
12	*authType:* **rsa-token**	Optional	EmptyJSONo	{}		Applies only to PPTP or L2TP. There is no further configuration required. The user has to enter RSA identifier when they connect.

	Field	Required	Type	Values	Can be parameter	Description
			bject			
13	*authType:* **shared-secret**	Optional	JSONobject	{ . . . }		Applies only to IPSec
14	*authType: shared-secret:* **group**	Optional	string	Max length 512		IPSec group identifier for the connection
15	*authType: shared-secret:* **sharedSecret**	Optional	string	Max length 512		IPSec shared secret
16	*authType: shared-secret:* **useHybridAuthentication**	Optional	boolean	true, false		Whether IPSec should used hybrid authentication. Defaults to false.
17	*authType: shared-secret:* **promptUserForPassword**	Optional	boolean	true, false		Whether IPSec should prompt user for password. Defaults to false.
18	**enableOnDemand**	Optional	boolean	true, false		Defaults to false. On demand rules are only applicable to certificate based connections. Does not apply to L2TP and PPTP VPNs.
D19	**onDemandRules**	Optional	JSONarray	[. . .]		Configuration of On demand rules used by certificate based VPN. Does not apply to L2TP and PPTP VPNs.
20	*onDemandRule*	Manda	stri	Max length 512		DNS domain or DNS server settings (with wildcard matching), WiFi SSID, network interface type or reachable server detection are supported.

	Field	Required	Type	Values	Can be parameter	Description
	s: **serverNamePattern**	tory	ng			
21	*onDemandRules*: **action**	Mandatory	string	• CONNECT_ALWAYS • CONNECT_NEVER • CONNECT_IF_NEEDED		But mandatory if serverNamePattern is specified CONNECT_ALWAYS, CONNECT_NEVER, CONNECT_IF_NEEDED
22	**maxIdleBeforeDisconnect**	Optional	number	Positive number		Time value in seconds. Value of 0 means never disconnect. Applies only when enableOnDemand is turned on. Does not apply to L2TP and PPTP VPNs.
23	**encryptionLevel**	Optional	string	AUTO, MAXIMUM, NONE		Applies only to PPTP
24	**sharedSecret**	Optional	string	Max length 512		Applicable only for L2TP
25	**group**	Optional	string	Max length 512		Applicable only for CiscoAnyConnect

	Field	Required	Type	Values	Can be parameter	Description
26	**domain**	Optional	string	Max length 512		Applicable only for SonicWALL
27	**role**	Optional	string	Max length 512		Applicable only for Juniper
28	**realm**	Optional	string	Max length 512		Applicable only for Juniper
29	**identifier**	Mandatory	string	Max length 512		Applicable only for CustomSSL. This field represents the VPN subtype. It's required in reverse DNS format.

ActiveSync

	Field	Required	Type	Values	Can be parameter	Description
01	**name**	Mandatory	string	Max length 512		Name of the ActiveSync Resource Configuration
02	**email**	Optional	string	Max length 512	Yes	User's email address
03	**allowToMoveMessageFromAccount**	Optional	boolean	true, false		Whether message are allowed to be moved to another account. true is default.
04	**allowRecentAddressSync**	Optional	boolean	true, false		Whether this account should be included in recent address syncing. true is default.
05	**limitSendingToMailAppOnly**	Optional	boolean	true, false		Allow only the Mail app to send outgoing message. false is default.
06	**sMimeConfig**	Optional	JSON Object	{ . . . }		S/MIME configuration.
07	*sMimeConfig*: **smime-disabled**	Optional	Empty JSON Object	{ . . . }		S/MIME is not supported. So, smime-disabled is the only supported option.
08	**hostName**	Mandatory	string	Max length 512		Microsoft

	Field	Required	Type	Values	Can be parameter	Description
						Exchange ActiveSync server hostname or IP address
09	**domain**	Optional	string	Max length 512	Yes	User's domain
10	*domain:* **userName**	Mandatory	string	Max length 512	Yes	Username for mail access. Recommended to be used with User Properties.
11	*domain:* **password**	Mandatory	string	Max length 512	Yes	Account Password.
12	**useSsl**	Optional	boolean	true, false		Use SSL for all communications to the server. true is default.
13	**daysToSync**	Optional	string	Either number between 0 to 5 or a string fro below • NO_LIMIT • DAY • THREE_ DAYS • WEEK • TWO_ WEEKS • MONTH		WEEK is default. This corresponds to number 3.
14	**identity**	Optional	JSON Object	See Credentials Config		Client certificate used to access Exchange

Plist

Plist configuration allows user to upload custom configuration profile created by Apple Configurator or other software.

JSON format is going to be the following:

```
{
   "name":"name for this profile"
   "plistBase64": "...<base 64 encoded mobileconfig file>..."
}
```

- name is required.
- plistBase64 is required.

WebClip

	Field	Required	Type	Values	Description
01	**url**	Mandatory	string	Max length 512	Name of the Web Clip Resource Configuration
02	**label**	Mandatory	string	Max length 512	Unique name for this resource
03	**icon**	Optional	string	base64	base64 encoded icon image to be used to display web clip on device.
04	**removable**	Optional	boolean	true, false	false by default
05	**showInFullScreen**	Optional	boolean	true, false	false by default
06	**displayIconWithoutVisualEffects**	Optional	boolean	true, false	false by default

KNOX

WiFi

	Field	Required	Type	Values	Can be parameter	Description
01	**ssid**	Mandatory	string	Max length 32		WiFi network name (SSID) that device should connect.
02	**hidden**	Optional	boolean	true, false		Indicates if the configured SSID is not broadcasting. false by default **Vanilla Android only**
03	**autoJoin**	Optional	boolean	true, false		Indicates whether the device should automatically connect to this SSID, when it is found. true by default.
04	**securityConfig**	Mandatory	JSON Object	{ . . . } Must be only one of the following child objects		Object to configure the WiFI security settings
05	*securityConfig*: **password**	Optional	JSON Object	{ . . . }		This presents PSK-based networks (Pre-shared key).

	Field	Required	Type	Values	Can be parameter	Description
06	*securityConfig*: *password*: **type**	Mandatory	string	WEP, WPA		WiFi standard to use
07	securityConfig: *password*: **password**	Mandatory	string	Max length of 512. Minimum length of 8 characters	Yes	Pre-Shared key (password) used by all devices to connect to the network.
08	*securityConfig*:**unsecured**	Optional	Empty JSON object	{ }		Indicates an open SSID network with no security
09	*securityConfig:* **enterprise**	Optional	JSON Object	{ . . . }		This object represents 802.1X networks (WPA2 Enterprise)
10	*securityConfig: enterprise*:**eapConfig**		JSON Object	{ . . . }		EAP configuration
11	*securityConfig: enterprise: eapConfig*:**eap**	Mandatory	string	TLS, TTLS, PEAP		802.1X EAP methods
12	*securityConfig: enterprise: eapConfig:* **userName**	Optional	string	Max length of 200.	Yes	
13	*securityConfig: enterprise: eapConfig:* **password**	Optional	string	Max length of 200	Yes	
14	*securityConfig: enterprise: eapConfig*:**identity**	Optional	JSON Object	See Credentials section		Mandatory for TLS.
15	*securityConfig: enterprise: eapConfig:* **outerIdentity**	Optional	string	Max length of 512		External identity used to protect the real identity of the user
16	*securityConfig: enterprise: eapConfig:* **TTLSInnerIdentity**	Optional	string	PAP, MSCHAP, MSCHAPv2, GTC, NONE		Mandatory if TTLS is one of the EAP types

VPN

	Field	Required	Type	Values	Description
01	name	Mandatory	string	Max length 512	

	Field	Required	Type	Values	Description
02	typeConfig	Mandatory	JSON Object	{...}	
03	*typeConfig* : **<vT>**	Mandatory	JSON Object	{...}	<vT> represents one of the following keys: L2TP, PPTP, IPSec, CiscoAnyConnect

Type Config

L2TP, IPSec, PPTP

	Field	Required	Type	Values	Description
01	hostName	Mandatory	string	Max length 512	
02	userName	Mandatory	string	Max length 512	
03	userPassword	Optional	string		
04	onlySecureConnections	Mandatory	bool		
05	dnsServers	Mandatory	list		
06	forwardRouters	Mandatory	list		
07	searchDomains	Mandatory	list		
08	type	Mandatory	string	L2TP: IPSEC_CRT IPSEC_PSK IPSec: HYBRID_RSA IKEV2_PSK IKEV2_RSA XAUTH_PSK XAUTH_RSA	L2TP, IPSec only
09	authType	Mandatory	JSON Object	{...}	L2TP, IPSec only
10	*authType*: **<vT>**	Mandatory	JSON Object	{...}	<vT> represents one of the following keys: PSK, Certificate L2TP, IPSec only
11	alwaysOn	Mandatory	bool		L2TP, IPSec only

	Field	Required	Type	Values	Description
12	enableSecret	Mandatory	bool		L2TP only
13	secret	Optional	string		L2TP only
14	identifier	Mandatory	string		IPSec only
15	ocspServerUrl	Optional	string		IPSec only
16	encryptionEnable	Mandatory	bool		PPTP only

Auth Type

PSK

N	Field	Required	Type	Values	Description
01	preSharedKey	Mandatory	string		

Certificate

N	Field	Required	Type	Values	Description
01	caCertName	Mandatory	string		
02	caCert	Mandatory	string	Base64 encoded cert	
03	userCertName	Mandatory	string		
04	userCert	Mandatory	string	Base64 encoded cert	

Cisco AnyConnect

N	Field	Required	Type	Values	Description
01	hostName	Mandatory	string	Max length 512	
02	type	Mandatory	string	ANYCONNECT	
03	certAuthMode	Mandatory	string	AUTOMATIC DISABLED MANUAL NULL	
04	certificate	Optional	string		BASE64 encoded PKCS12 certificate Mandatory if certAuthMode is AUTOMATIC or MANUAL
05	password	Optional	string		Mandatory if certAuthMode is AUTOMATIC or MANUAL

Example

```
{
                "name" : "L2TP",
                "typeConfig" : {
                        "L2TP" : {
```

```
                                "hostName" : "hostname",
                                "userName" : "user",
                                "userPassword" : null,
                                "onlySecureConnections" : false,
                                "dnsServers" : ["dns"],
                                "forwardRouters" : ["router"],
                                "searchDomains" : ["search"],
                                "alwaysOn" : true,
                                "type" : "IPSEC_CRT",
                                "enableSecret" : true,
                                "secret" : "secret",
                                "authType" : {
                                        "Certificate" : {
                                                "caCertName" : "ca-name",
                                                "caCert" : "ca-string",
                                                "userCertName" : "user-cert-name",
                                                "userCert" : "user-cert-string"
                                        }
                                }
                        }
                }
}
```

ActiveSync

N	Field	Required	Type	Values
01	name	Mandatory	string	Max length 512
02	acceptAllCertificates	Optional	bool	
03	certificateData	Optional	string	BASE64 encoded cert
04	certificatePassword	Optional	string	
05	displayName	Optional	string	
06	easDomain	Mandatory	string	Max length 512
07	easUser	Mandatory	string	Max length 512
08	email	Mandatory	string	Max length 512
09	emailNotificationVibrateAlways	Optional	bool	
10	emailNotificationVibrateWhenSilent	Optional	bool	
11	defaultAccount	Optional	bool	
12	notifyOnReceivingNewMail	Optional	bool	
13	peakSyncFrequency	Optional	string	NEVER AUTOMATIC FIVE_MINUTES TEN_MINUTES FIFTEEN_MINUTES THIRTY_MINUTES ONE_HOUR FOUR_HOURS TWELVE_HOURS

N	Field	Required	Type	Values
14	offPeakSyncFrequency	Optional	string	NEVER AUTOMATIC FIVE_MINUTES TEN_MINUTES FIFTEEN_MINUTES THIRTY_MINUTES ONE_HOUR FOUR_HOURS TWELVE_HOURS
15	peakDays	Optional	int	0 - 127
16	peakStartTime	Optional	int	0 - 1440
17	peakEndTime	Optional	int	0 - 1440
18	periodCalendar	Optional	string	ALL TWO_WEEKS ONE_MONTH THREE_MONTHS SIX_MONTHS
19	protocolVersion	Optional	string	
20	retrivalSize	Optional	string	HEADERS_ONLY HALF_KB ONE_KB TWO_KB FIVE_KB TEN_KB TWENTY_KB FIFTY_KB ONE_HUNDRED_KB ALL
21	roamingSyncSchedule	Optional	string	MANUAL USE_SYNC_SETTING
22	senderName	Optional	string	
23	serverAddress	Mandatory	string	Max length 512
24	serverPassword	Mandatory	string	
25	serverPathPrefix	Optional	string	
26	signature	Optional	string	
27	syncCalendar	Optional	bool	
28	syncContacts	Optional	bool	
29	syncTasks	Optional	bool	
30	syncNotes	Optional	bool	
31	syncInterval	Optional	string	NEVER AUTOMATIC FIVE_MINUTES TEN_MINUTES FIFTEEN_MINUTES THIRTY_MINUTES ONE_HOUR FOUR_HOURS TWELVE_HOURS
32	syncLookback	Optional	string	ONE_DAY THREE_DAYS

N	Field	Required	Type	Values
				ONE_WEEK TWO_WEEKS ONE_MONTH
33	useSSL	Optional	bool	
34	useTLS	Optional	bool	
35	allowIncomingAttachments	Optional	bool	

Example

```
{
        "name" : "",
        "config" : {
                "name" : "exchange1",
                "acceptAllCertificates" : false,
                "certificateData" : null,
                "certificatePassword" : null,
                "displayName" : null,
                "easDomain" : "xyz.com",
                "easUser" : "user1",
                "email" : "user1@xyz.com",
                "emailNotificationVibrateAlways" : false,
                "emailNotificationVibrateWhenSilent" : false,
                "defaultAccount" : false,
                "notifyOnReceivingNewMail" : false,
                "peakSyncFrequency" : null,
                "offPeakSyncFrequency" : null,
                "peakDays" : null,
                "peakStartTime" : null,
                "peakEndTime" : null,
                "periodCalendar" : null,
                "protocolVersion" : null,
                "retrivalSize" : null,
                "roamingSyncSchedule" : null,
                "senderName" : null,
                "serverAddress" : "mail.xyz.com",
                "serverPassword" : "password",
                "serverPathPrefix" : null,
                "signature" : null,
                "syncCalendar" : false,
                "syncContacts" : false,
                "syncTasks" : false,
                "syncNotes" : false,
                "syncInterval" : null,
                "syncLookback" : null,
                "useSSL" : false,
                "useTLS" : false,
                "allowIncomingAttachments" : true
        }
}
```

Credentials

Credentials resource used by some iOS and KNOX resource configurations to specify user's SSL identity used by resource configuration in PKCS12 format.

	Field	Required	Type	Values	Can be parameter	Description
01	**name**	Optional	string	Max length 512		Name of the Credentials Resource Configuration
02	**fileName**	Mandatory	string	Max length 512		Name as it should appear on the device
03	**content**	Mandatory	string	base64	Yes	base64 encoded PKCS#12 format keystore
04	**password**	Optional	string	Max length 512	Yes	Must be specified if keystore type is PKCS12. Must not be specified for others.
05	**type**	Optional*	string	PKCS1, PKCS12, ROOT		Type of the credential. PKCS12 is default.

Windows

WebLink

N	Field	Required	Type	Values	Can be parameter	Description
01	**url**	Mandatory	string	Max length 512		Link URL
02	**label**	Mandatory	string	Max length 512		Link label

BlackBerry Dynamics documentation

Category	Title	Description
Cross-platform	• *Getting Started Guide for Marketplace Partners* • *Good Control/Good Proxy Platform Overview for Administrators and Developers* • *Good Cloud Deployment*	Overviews of the BlackBerry Dynamics system
	• *Good Control Device and Application Management* • *Good Control Device Management Enrollment: Good Agent for iOS* • *Good Control Device Management Enrollment: Good Agent for Android*	Device and application management on Good Control, including app distribution, with client-side device enrollment details
Security	*BlackBerry Dynamics Security White Paper*	Description of the security aspects of BlackBerry Dynamics
	BlackBerry Dynamics and Fingerprint Authentication	Discussion of the implementation of BlackBerry security with fingerprint recognition systems: Apple Touch ID and Android Fingerprint
BlackBerry UEM	BlackBerry UEM Administration Guide	Approaches to administering the BlackBerry Unified Endpoint Manager
	Getting Started with BlackBerry UEM and BlackBerry Dynamics	Introductory material to administering the BlackBerry UEM with the BlackBerry Dynamics profile
Good Control	• *BlackBerry Secure Enterprise Planning Guide* • *BlackBerry Secure Servers Compatibility Matrix* • *BlackBerry Performance Calculator*	Guidelines and tools for planning your BlackBerry Secure Enterprise deployment
	Good Control/Good Proxy Server Preinstallation Checklist	Same checklist extracted from the installation guide below
	Good Control/Good Proxy Server	Details on installing Good Control, Good Proxy, and the GC

Category	Title	Description
	Installation	database
	Kerberos Constrained Delegation for Good Control	Configuration details for integrating the Kerberos authentication system with BlackBerry Dynamics
	Direct Connect for Good Control	Configuring BlackBerry Dynamics servers to securely access internal resources from the external Internet
	Good Control Easy Activation Overview	A look at the Easy Activation feature
	Good Control/Good Proxy Server Backup and Restore	Minimal steps for backing up and restoring the BlackBerry Dynamics system
	Good Control Online Help Good Control Server Property and Security Policy Reference	Printable copy of the GC console online help
	PKI Cert Creation via Good Control: Reference Implementation	A reference implementation in Java for creating end-user PKI certificates via Good Control and a Certificate Authority (CA)
	Good Control Cloud Online Help	Printable copy of the Cloud GC console online help
	Technical Brief: BlackBerry Dynamics Application Policies with XSD for app-policy XML	Description of formatting application policies for use in Good Control, with examples.
	Development Guide: Good Control Web Services	Programmatic interfaces on Good Control • Basic control and application management: SOAP over HTTPS. Documentation is in the WSDL files included with GC. • Device management: HTTP API (with JSON) for device management. Zipfile of API reference.
Software Development	*Developer Bootstrap: Good Control Essentials*	Bare minimum of information that a developer of BlackBerry Dynamics applications needs to get started with the Good Control server to test applications.
	BlackBerry Dynamics Shared Services Framework	Description of the BlackBerry Dynamics shared services framework for software developers
Android	• *Development Guide: BlackBerry Dynamics SDK for Android* • *API Reference for Android*	Working with the BlackBerry Dynamics SDK for Android and the essential reference for developers
iOS	• *Development Guide: BlackBerry Dynamics SDK for iOS*	Working with the BlackBerry Dynamics SDK for iOS and the essential reference for developers

Category	Title	Description
	• *API Reference for iOS*	
macOS	• *Development Guide: BlackBerry Dynamics SDK for macOS* • *API Reference for macOS*	Working with the BlackBerry Dynamics SDK for macOS and the essential reference for developers
Windows	• *Development Guide: BlackBerry Dynamics SDK for Universal Windows Platform (UWP)* • *API Reference for UWP*	Working with the BlackBerry Dynamics SDK for Universal Windows Platform (UWP) and the essential reference for developers
iOS, Android	*BlackBerry Launcher Library*	Source code and header files for implementing the popular BlackBerry Launcher interface
Cross-platform	*Development Guide: BlackBerry Dynamics SDK for Cordova for iOS and Android*	Working with the BlackBerry Dynamics SDK for Cordova plugins
	BlackBerry Dynamics Secure HTML5 Bundle Getting Started Guide for Developers	Working with the BlackBerry Dynamics SDK and the secure HTML5 bundle
	BlackBerry Dynamics Bindings for Xamarin.Android	Working with the BlackBerry Dynamics SDK and the Xamarin cross-platform integrated development environment For Xamarin.Android, no separate API reference is needed; see the standard BlackBerry Dynamics SDK *API Reference for Android*
	BlackBerry Dynamics Bindings for Xamarin.iOS and the *API Reference for Xamarin.iOS*	Working with the BlackBerry Dynamics SDK and the Xamarin cross-platform integrated development environment