Good Control Web Services

Last updated: August 10, 2015

Version: GC 2.0.*xx.yy*

Good Dynamics™

Legal Notice

This document, as well as all accompanying documents for this product, is published by Good Technology Corporation ("Good"). Good may have patents or pending patent applications, trademarks, copyrights, and other intellectual property rights covering the subject matter in these documents. The furnishing of this, or any other document, does not in any way imply any license to these or other intellectual properties, except as expressly provided in written license agreements with Good. This document is for the use of licensed or authorized users only. No part of this document may be used, sold, reproduced, stored in a database or retrieval system or transmitted in any form or by any means, electronic or physical, for any purpose, other than the purchaser's authorized use without the express written permission of Good. Any unauthorized copying, distribution or disclosure of information is a violation of copyright laws.

While every effort has been made to ensure technical accuracy, information in this document is subject to change without notice and does not represent a commitment on the part of Good. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of those written agreements.

The documentation provided is subject to change at Good's sole discretion without notice. It is your responsibility to utilize the most current documentation available. Good assumes no duty to update you, and therefore Good recommends that you check frequently for new versions. This documentation is provided "as is" and Good assumes no liability for the accuracy or completeness of the content. The content of this document may contain information regarding Good's future plans, including roadmaps and feature sets not yet available. It is stressed that this information is non-binding and Good creates no contractual obligation to deliver the features and functionality described herein, and expressly disclaims all theories of contract, detrimental reliance and/or promissory estoppel or similar theories.

Legal Information

Patent Information: https://www1.good.com/legal/other-legal.html#trademark

Table of Contents

Revision History

Good Control Web Services

Date	Description

2015-08-10	• Updated for latest release: New requests for working with Samsung KNOX domains and Android device passwords. See What's New?
	• Added complete lists of requests for the SOAP interface:
	  • Alphabetical List of Requests in gc.wsdl
	  • Alphabetical List of Requests in cap.wsdl
	• Added advice about About SOAP-Aware Client Software: Use Your Favorite

2015-07-16	Added helpful details about ways to base-64-encode your credentials in Background on HTTP API Usage: Endpoint, Authorization, HTTP Verbs, and More
2015-06-18	Added new summary Requests by Cataegory: Device Management HTTP API.
2015-06-08	• Removed API calling sequence diagram because it was not helpful. • Added details about required MIME types (HTTP header Content-type) for requests in Background on HTTP API Usage: Endpoint, Authorization, HTTP Verbs, and More
2015-06-04	Added new subsection "Constructing Your Requests" in Background on HTTP API Usage: Endpoint, Authorization, HTTP Verbs, and More with examples of fully formed requests for the device management HTTP API
2015-05-18	• Version numbers updated for latest release • Corrected URL for endpoint for device management: `mdm`, not `mam`.
2015-04-30	Updated for latest release: HTTP API for device management. See HTTP API for Device Management. Replaced erroneous ns2 and ns6 namespace tags with urn in Example: Adding a User to GC from an Active Directory Domain.
2015-03-26	Updated for latest release: • The cap.wsdl file includes new or updated requests for managing applications. These requests are listed in Application Management Requests
2015-03-17	Corrected erroneous <ns2:stringID> field name in GetDirectoryServicesRequest to <ns2:sessionId> in Example: Adding a User to GC from an Active Directory Domain
2015-03-12	Removed erroneous information about changing the hostnames specified in the WSDL files for the various namespaces. No need to change these hostnames; do not modify the WSDL

Date	Description
	as delivered.
2015-01-15	Version number updated for latest release; no content changes.
2015-01-05	Added explicit section on required endpoints for requests. See Endpoints for SOAP Requests.
2014-12-16	Styling and page layout updated
2014-10-13	Corrected example of `GetDirectoryUserResponse`, `AddUserRequest`, and `AddUserResponse` to show `<stringId>` element, instead of erroneous `<emailAddress>` to conform with `gc.wsdl`.
2014-10-07	Corrected example to use `GetDirectoryUsersRequest`, not `GetUsersRequest`.
2014-09-29	Added information about the additional WSDL file `c:\good\docs\cap.wsdl`.
2014-09-25	• Updated for latest version of GC • Version stamp on cover page
2014-08-21	Added standard list of all Good Dynamics documentation
2014-08-11	Start of revision history

Good Control Web Services

Good Control has a web services interface for programatically administering the system.

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.

This guide introduces the GC Web Services at a high level and gives you pointers to the necessary service definitions and an overview of the requests available for your administrative use. We assume that in general you are familiar with how SOAP-based web services work.

What's New?

Highlighted here are some of the recent changes and improvements.

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 Good 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 Good For KNOX “enterprise" domain. After all, Good 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 Good for KNOX.

In earlier releases, if an administrator turned off Good 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 Good 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.

Relationship to Cloud GC: Feature Supported

The feature, service, server type, or software described here is fully supported on and compatible with Good Control Cloud.

About Good 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 Proxy	2.0.3.7
Good Control	2.0.3.11
GD SDK for Microsoft Windows	1.0.749
GD SDK for Android	2.0.1226
GD PhoneGap	2.0.71
GD SDK for iOS	2.0.4407
Digital Authentication Framework (DAF) • Android • iOS	 • 2.0.147 • 2.0.174

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

GC and WSDL Documentation

In many ways, the GC Web Services interface mirrors the functions GC console web-based user interface, with which you should be familiar:

- The Good Control console comes with extensive online help. After logging in, in the upper right, click **Help** > **Help Contents**.
- The same information is also available as a PDF book on the Good Developer Network; see Good Control Console Online Help.
- For the GC Web Services, the main source of truth is the `gc.wsdl` and `cap.wsdl` files. 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
```

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.

Good Technology 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.

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. Other user-related requests are listed in Categories of Requests.

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>
```

GC and CAP WSDL: Location, Request Syntax, Responses, and Errors

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 Good Technology representative.

The top of both files also define other required schemas.

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

Endpoints for SOAP Requests

The GC web services has two endpoints, depending on which of the WSDL files you are working with, either gc.wsdl or cap.wsdl. In either case, in the endpoints below, *localhost* is the full qualified domain name of your GC server

- gc.wsdl: `https://localhost/gc/services/GCService`
- cap.wsdl: `https://localhost/gc/soapproxy/cap`
- CAP service endpoint:

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. Many of the requests are group by administrative function in Categories of Requests.

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.

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

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>
  .
  .
  .
```

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

Alphabetical List of Requests in gc.wsdl

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

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

```
AddAdministratorRequest
AddCertificateRequest
AddGCServiceAdminRequest
AddOrUpdateApplicationServerRequest
AddOrUpdateDomainServerRequest
AddRoleMembersRequest
AddServerRequest
AddSplitBillingPkgRequest
AddTrustedCertificateRequest
AddUserRequest
AssignSplitBillingPkgRequest
AuditTrailExportRequest
AuditTrailPurgeRequest
BulkAddUsersFromGroupRequest
BulkAddUsersRequest
BulkManageUsersRequest
CancelUploadLogScheduleRequest
ChangeGCServiceAdminPasswordRequest
ChangePolicyAppRequest
ChangePolicyUserRequest
ClearUploadLogMessagesRequest
CopyPolicyRequest
```

```
CopyRoleRequest
DeleteContainerRequest
DeleteGPClusterRequest
DeleteRoleRequest
DeleteSplitBillingPkgRequest
DisableContainerLoggingRequest
EnableContainerLoggingRequest
EndSessionRequest
ExportComplianceReportRequest
ExportContainerReportRequest
FetchDomainsRequest
GenerateAccessKeysRequest
GenerateRestrictedAccessKeyRequest
GenerateUnlockAccessKeyRequest
GetAccessKeysRequest
GetActivatedContainersRequest
GetAdGroupPreviewUsersRequest
GetAdministratorsRequest
GetAllPoliciesRequest
GetAllProvisionedContainersRequest
GetAllowV1PinsRequest
GetAppInfoRequest
GetAppPolicyNameRequest
GetAppPolicyRequest
GetAppVersionOverrideRequest
GetAppsRequest
GetAppsWithPolicyRequest
GetBulkAddUsersUpdateRequest
GetBulkManageUsersUpdateRequest
GetBulkUsersConfigRequest
GetBulkUsersUsersRequest
GetCertificatesRequest
GetContainerEventsRequest
GetDashboardDataRequest
```

GetDeploymentInfoRequest

GetDevicesRequest

GetDirectConnectInfoRequest

GetDirectoryUsersRequest

GetDomainsRequest

GetEffectiveRightsForUserRequest

GetGCPropertiesRequest

GetGCReportsLimitRequest

GetGPClusterListRequest

GetGPClusterServerListRequest

GetGroupsRequest

GetJobByIdRequest

GetJobsRequest

GetLicenseSerialRequest

GetPolicyDetailRequest

GetPolicyNameRequest

GetRegistrationEmailInfoRequest

GetRoleRequest

GetSelfServiceInfoRequest

GetSessionInfoRequest

GetSessionTimeoutRequest

GetSplitBillingPkgRequest

GetTempUnlockPasswordRequest

GetTempUnlockTypeRequest

GetThisServerRequest

GetTrustedCertificatesRequest

GetUnlockAccessKeyRequest

GetUploadLogMessagesRequest

GetUploadLogScheduleRequest

GetUseLowPortsRequest

GetUserRequest

GetUsersRequest

GetWrappingEngineVersionRequest

GetWrappingPropertyRequest

ListRolesRequest

LockContainerRequest

MakeDefaultPolicyRequest

NoopRequest

PingSigningServerRequest

RemoveAccessKeyRequest

RemoveAdministratorRequest

RemoveAppRequest

RemoveCertificateRequest

RemovePolicySetRequest

RemoveRoleMemberRequest

RemoveTrustedCertificateRequest

RemoveUnlockAccessKeyRequest

RemoveUserRequest

ResendWelcomeEmailRequest

ResetTempPasswordRequest

SearchAdministratorsRequest

SendEnrollmentKeyEmailRequest

SendPinEmailRequest

ServerStatusRequest

SetAdGroupPreviewUsersRequest

SetAllowV1PinsRequest

SetDeploymentInfoRequest

SetGCPropertiesRequest

SetNewPasswordRequest

SetRegistrationEmailInfoRequest

SetSelfServiceInfoRequest

SetSessionTimeoutRequest

SetUploadLogScheduleRequest

SetUseLowPortsRequest

SetWrappingPropertyRequest

SignIccCertificateRequest

TriggerAppPolicyDownloadRequest

UnAssignSplitBillingPkgRequest

```
UnregisterServerRequest
UpdateAppPoliciesRequest
UpdateAppRequest
UpdateAppVersionOverrideRequest
UpdateCertificateRequest
UpdateContainerManagementAppServerRequest
UpdateDirectConnectInfoRequest
UpdateDomainsRequest
UpdateGPClustersRequest
UpdatePolicyNameDescRequest
UpdatePolicySetRequest
UpdateRoleRequest
UpdateRoleRightsRequest
UpdateTrustedCertificateRequest
UpdateWrappingEngineVersionRequest
UploadClientLogMessageRequest
WrapAppRequest
certSignRequest
getServerListRequest
getUnassignedServerListRequest
```

Alphabetical List of Requests in cap.wsdl

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

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

```
addAdminRequest
addAppCategoryRequest
addAppRequest
addAppServiceRequest
addAppTagRequest
addAppVersionRequest
addAssociationsRequest
addBundleRequest
```

addBundleVersionLocalesVersionRequest

addCategoryLocaleRequest

addCategoryRequest

addEnterpriseRequest

addGroupRequest

addGroupUserRequest

addGroupsUsersRequest

addOrganizationRequest

addResourceLinksRequest

addResourceRequest

addResourceSetsRequest

addScreenshotsRequest

addServiceRequest

addServiceVersionRequest

addVersionLocaleRequest

createAppAndVersionRequest

createAppVersionRequest

editAppBinaryVersionMetaDataRequest

editAppBinaryVersionReleaseNotesRequest

editAppIconRequest

editAppPlatformDescriptionRequest

fetchAppMetaDataRequest

getAdminsRequest

getAppCategoriesRequest

getAppDetailsRequest

getAppLocalAddressRequest

getAppPermissionsRequest

getAppPolicyInfoRequest

getAppPolicyRequest

getAppPolicyVersionListRequest

getAppServicesRequest

getAppTagsRequest

getAppVersionAudienceRequest

getAppVersionsRequest

```
getAppsPublishedToOrganizationRequest
getAppsRequest
getAssociationsRequest
getBundlesRequest
getCategoriesRequest
getCategoryLocalesRequest
getDeviceGroupsRequest
getEnterpriseRequest
getEnterpriseServersRequest
getEnterprisesRequest
getGroupPermissionsRequest
getGroupsForUserRequest
getGroupsRequest
getOrganizationsRequest
getPermissionDetailsRequest
getPublicAppDetailsRequest
getPublicAppsRequest
getPublicServiceVersionsRequest
getResellersRequest
getResolvedPermissionsRequest
getResourceRequest
getResourceSetsRequest
getResourcesRequest
getServiceDetailsRequest
getServiceVersionInterfaceRequest
getServiceVersionsRequest
getServicesRequest
getUnassignedEnterprisesRequest
getUserPermissionsRequest
getUsersInGroupRequest
getUsersNotInGroupRequest
getUsersRequest
getVersionLocalesRequest
importOrganizationRequest
```

noopRequest

parseBinaryRequest

publishAppRequest

publishAppVersionRequest

removeAdminRequest

removeAppBinaryVersionRequest

removeAppCategoryRequest

removeAppRequest

removeAppServiceRequest

removeAppTagRequest

removeAppVersionRequest

removeAssociationsRequest

removeBundleRequest

removeCategoryLocaleRequest

removeCategoryRequest

removeEnterpriseRequest

removeGroupRequest

removeGroupUserRequest

removeOrganizationRequest

removeResourceLinksRequest

removeResourceRequest

removeResourceSetRequest

removeServiceRequest

removeServiceVersionRequest

removeVersionLocaleRequest

setAppLocalAddressRequest

setAppPolicyRequest

setGroupPermissionRequest

setUserPermissionRequest

unpublishAppRequest

unpublishAppVersionRequest

updateAdminRequest

updateAppMetaDataRequest

updateAppRequest

updateAppVersionRequest

updateBundleRequest

updateCategoryLocaleRequest

updateCategoryRequest

updateEnterpriseRequest

updateEnterpriseTypeRequest

updateGroupRequest

updateOrganizationRequest

updateResourceRequest

updateResourceSetRequest

updateServiceRequest

updateServiceVersionRequest

updateVersionLocaleRequest

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.

DM 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

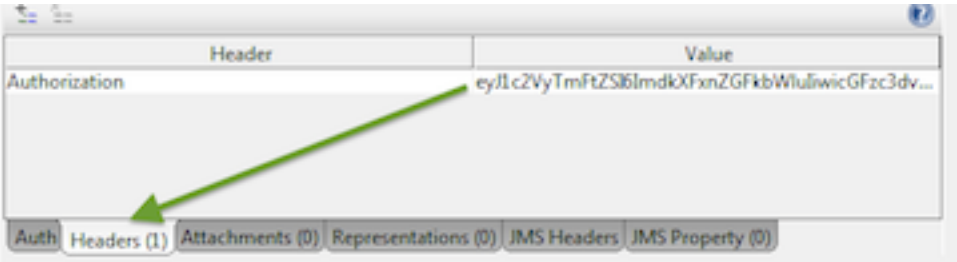
There are several ways you can base64-encode your credentials.

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","yourPassword":"Pa55"}' | base64
eyJ1c2VyTmFtZSI6InlvdXJEb21haW5cXHlvdXJHQ2xvZ2luIiwieW91clBhc3N3b3JkIjoiUGE1NSJ9Cg==
```

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 Good 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 DM 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 DM 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 GD 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

	Field	Required	Type	Values	Can be parameter	Description
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 objects		Object to configure the WiFI security settings
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

	Field	Required	Type	Values	Can be parameter	Description
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

	Field	Required	Type	Values	Can be parameter	Description
17	*securityConfig: enterprise: eapConfig:* **TTLSInnerIdentity**	Optional	string	PAP, CHAP, MSCHAP, MSCHAPv2		Mandatory if TTLS is one of the EAP types
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

	Field	Required	Type	Values	Can be parameter	Description
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	NAIs used to authenticate users.

	Field	Required	Type	Values	Description
			Each element max length is 512		
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 followin g child objects	Proxy Configuration applies to VPN and WiFi resource configuration s. Must be one of the following child objects.
02	proxyConfig: **automatic**	Optional	JSON Object	{ . . . }	Configuration of automatic proxy.
03	proxyConfig: automatic: **configUrl**	Mandator y	string	Max length 512	URL to use for configuring automatic proxy settings

N	Field	Required	Type	Values	Description
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
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,	true â€“ VPN will be used for all traffic on the device. Default

	Field	Required	Type	Values	Description
				false	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	JSONObject	{ . . . }		`Example: "ArubaVIA": { "authType" : { "certificate" : { "identity" : { "name" : "certName", "fileName" : "certFileName.p12", "content": "base64 encoded PKCS#12 format keystore", "password": "pAssWorD" }}}}`

	Field	Required	Type	Values	Can be parameter	Description
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	EmptyJSONobject	{ }		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
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.

	Field	Required	Type	Values	Can be parameter	Description
20	*onDemandRules*: **serverNamePattern**	Mandatory	string	Max length 512		DNS domain or DNS server settings (with wildcard matching), WiFi SSID, network interface type or reachable server detection are supported.
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**	Option	stri	Max length 512		Applicable only for L2TP

	Field	Required	Type	Values	Can be parameter	Description
		al	ng			
25	**group**	Optional	string	Max length 512		Applicable only for CiscoAnyConnect
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.

	Field	Required	Type	Values	Can be parameter	Description
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 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		WEEK is default. This corresponds to number 3.

	Field	Required	Type	Values	Can be parameter	Description
				• MONTH		
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).
06	*securityConfig*: *password*: **type**	Mandatory	string	WEP, WPA		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.

	Field	Required	Type	Values	Can be parameter	Description
				Minimum length of 8 characters		
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	
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:	L2TP, IPSec only

	Field	Required	Type	Values	Description
				HYBRID_RSA IKEV2_PSK IKEV2_RSA XAUTH_PSK XAUTH_RSA	
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
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
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	

N	Field	Required	Type	Values
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 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

	Field	Required	Type	Values	Can be parameter	Description
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

Good Dynamics Documentation

All documents are in PDF and available on the Good Developer Network.

Category	Title	Description
Cross-platform	• *Getting Started Guide for Marketplace Partners* • *Good Dynamics Platform Overview for Administrators and Developers* • *Good Cloud Deployment*	Overviews of the Good Dynamics system
	• *Good Device and Application Management* • *DM Enrollment: Good Agent for iOS* • *DM Enrollment: Good Agent for Android*	Device and application management on Good Control, including app distribution, with client-side device enrollment details
Security	*GD Security White Paper*	Description of the security aspects of Good Dynamics
	GD Security White Paper: Mobile Application Management	Focus specifically on application management
	Good Dynamics with Apple Touch ID	Discussion of the implementation of Good security with Apple's fingerprint recognition system
Servers	*GD Sizing Guide*	Recommendations and details about capacity planning for your GD deployment
	GD Server Preinstallation Checklist	Same checklist extracted from the installation guide below
	Good Dynamics Server Installation	Details on installing Good Control, Good Proxy, and the GC database
	GD Server Clustering and Affinities	Configuration details on increasing the capacity of your deployment
	Kerberos Constrained Delegation for Good Dynamics	Configuration details for integrating the Kerberos authentication system with GD
	Direct Connect	Configuring Direct Connect to securely access internal resources from the external Internet
	Easy Activation Overview	A look at the Easy Activation feature
	GD Server Backup and Restore	Minimal steps for backing up and restoring the GD system
	Good Control Online Help	Printable copy of the GC console online help
	Good Control Cloud Online Help	Printable copy of the Cloud GC console online help
	Good Control Web Services: Programmatic interfaces on Good Control	

Category	Title	Description
	• 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.	
	Good Wrapping Server Installation	Details for installing Good Wrapping server
	GD Application Wrapping Guide	Details about wrapping applications
Software Development	*GD Shared Services Framework*	Description of the GD shared services framework for software developers
	GD Connecting to A Clustered Application Server	Details necessary if you have clustered your application servers
iOS	• *GD SDK for iOS* • *API Reference for iOS*	Working with the GD SDK for iOS and the essential reference for developers
Android	• *GD SDK for Android* • *API Reference for Android*	Working with the GD SDK for Android and the essential reference for developers
Windows	• *GD SDK for Windows* • *API Reference for Windows*	Working with the GD SDK for Windows and the essential reference for developers
iOS, Android	*Good Launcher Library*	Source code and header files for implementing the popular Good Launcher interface
Cross-platform	*Getting Started Guide for PhoneGap Developers - iOS and Android*	Working with the GD SDK and the Cordova PhoneGap plugin
	GD Secure HTML5 Bundle Getting Started Guide for Developers	Working with the GD SDK and the secure HTML5 bundle
	GD Bindings for Xamarin for Android and for iOS and the *API Reference for Xamarin.iOS*	Working with the GD SDK and the Xamarin cross-platform integrated development environment For Xamarin.Android, no separate API reference is needed; see the standard GD SDK *API Reference for Android*