GD Server Deployment Planning and Installation

Last updated: September 8, 2015

Versions GP 2.0.*xx.yy*, 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

© Copyright 2015. All rights reserved. All use is subject to license terms posted at www.good.com/legal. GOOD, GOOD TECHNOLOGY, the GOOD logo, GOOD FOR ENTERPRISE, GOOD FOR GOVERNMENT, GOOD FOR YOU, GOOD APPCENTRAL, GOOD DYNAMICS, SECURED BY GOOD, GOOD MOBILE MANAGER, GOOD CONNECT, GOOD SHARE, GOOD TRUST, GOOD VAULT, and GOOD DYNAMICS APPKINETICS are trademarks of Good Technology Corporation and its related entities. All third-party technology products are protected by issued and pending U.S. and foreign patents.

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

Table of Contents

Revision History

GD Deployment Planning and Installation

Date	Description
2015-09-08	Added Upgrade or Install GC and GP In Parallel.
2015-09-03	• Added guideline in Synchronized Time Among Load Balancers, GC, and GP. • Removed from Installing Good Proxy Server the erroneous recommendation to set the GP server clock behind that of the GC.
2015-08-27	• Release of GD SDK for iOS and for Android • Updated to include information about needing the 64-bit Windows operating system for Good Control is Now 64-bit. • Details from the body of the guide added directly to the checklist regarding exact outbound ports, minimum POC server hardware, and server operating system version, and networking tools.
2015-08-10	Updated for latest release: • Good Control now runs the 64-bit Java Runtime Engine (JRE). No impact on actual installation steps. • New sizing/capacity guidelines are in GC Sizing Recommendations: JRE Memory, DB Latency, CPUs and More. • See also Optional Notification Configuration for Performance Increase with Oracle.
2015-07-15	Clarified the hostname requirement for the GC in Pre-installation Checklist, Installing the First Good Control Server in Server Cluster, and Installing Additional Good Control Server in Server Cluster: must be a DNS A record (that is, canonical address record), not a DNS alias.
2015-07-09	Clarified necessities for SQL Database Account for Multi-Realm KCD Configuration.
2015-07-08	Clarification in Pre-installation Checklist of necessary GD database account: *not* the db_owner privilege, but the service or SQL account that actually owns the GD database.
2015-07-07	For qualification of "valid Internet domain names" in Installing the First Good Control Server in Server Cluster, added reference to RFC 1123 at https://tools.ietf.org/html/rfc1123. Specifically, no underscores are allowed.
2015-05-18	Version numbers updated for latest release; no content changes.
2015-04-30	Version numbers updated for latest release; no content changes.
2015-04-15	Added Migrating the Good Control Database.

Date	Description
2015-03-26	Updated for latest release: During GC and GP installation you can specify an alternate location for log files. Corrected network diagram in Opening and Checking Outbound Firewall Configurationsto include the bxenroll and bxcheckin server names required for Good device management. These hostnames were formerly only listed in the text.
2015-03-23	Included important advice that you must install or upgrade *all* servers in a Good Control cluster to the same version of GC before using them. Otherwise, the servers are left in an inconsistent state that can cause unpredictable problems.
2015-02-27	Updated Browser Recommendations
2015-02-17	Added topic Increasing the GC or GP Server's Java Heap Size.
2015-02-13	Added details about whitelisting the Good Mobile Device Management server by name: bxenroll.good.com. See Opening and Checking Outbound Firewall Configurations
2015-01-28	Updated for latest release
2015-01-22	• Procedure for upgrading the GC and GP in High-Level Steps for Upgrading GC or GP. • Guide regenerated to include missing network diagram reported from field.
2015-01-15	Version numbers updated for latest release. Includes updated content for Kerberos Constrained Delegation (mirrored in the seperate document for KCD): Support for Multiple Kerberos Realms.
2014-12-16	Styling and page layout updated
2014-12-15	Network latency between GC and its database must not be greater than 10 milliseconds.
2014-11-24	Add emphasis that the GC hostname is a fully qualified domain name (FQDN): no control characters, special characters, or wildcard characters allowed.
2014-11-10	Added note about uninstalling the GC or GP servers
2014-11-04	• Expanded details and task steps for installing GC and GP in different Active Directory domains. • Direct Connect: support for SSL-certificate-based client authentication. • Kerberos Constrained Delegation: reorganization for readability/logic
2014-10-21	Added note about ability to install the GC and GP servers in separate domains.
2014-10-14	• Minor addition to Good Proxy Scalability . • Clarification of *target* for How Kerberos Constrained Delegation Works.

Date	Description
2014-10-07	• Miscellaneous clarifications throughout.
2014-10-01	• Corrected steps for installing the GP server. • Added explicit version numbers to introduction.
2014-09-25	• Updated for latest versions of GC and GP. • Version stamp on cover page.
2014-09-10	Clarification of "SQL server hostname" for installing the GC: can include database name in the form of `mysql.mydomain.com\SQLEXPRESS`.
2014-09-09	• Various clarifications in "Pre-Installation Checklist" section. • Clarification of "SQL server hostname" for installing the GC: must be the fully qualified Internet domain name.
2014-09-01	• Corrected section "Internet Domain for GC and GP". The domain is the Active Directory domain, not the Internet domain. • Adjusted contradictory details in NOC IP address ranges in "Pre-installation Checklist". • Removed erroneous, outdated material from "After the Installation is Complete" section and added reference to list of documentation for information about features/tasks that might be of interest.
2014-08-21	Added standard list of all Good Dynamics documentation.
2014-08-20	Added note to GD Server/Network Specifications and Deployment Configurations on the necessity of non-null Display Names in the Active Directory service where user records are imported. If a Display Name from AD is null, the user cannot log in to the GC.
2014-08-14	• Clarified for the GP installer, you must know the fully qualified Internet domain name of your GD servers. Not just the bare `hostname`, but the full `hostname.somedomain.com`. • Out-of-date scalability information removed; link to GD Scalability. • Out-of-date details on Kerberos removed; link to Kerberos Constrained Delegation.
2014-08-13	Corrected recommendation on supported browsers.
2014-08-11	New subsection in introduction: "Simplified Views of Deployment Configurations".

Introduction

This guide includes basic hardware and operating system software specifications, deployment topologies, and the installation steps for the Good Control server, its associated database, and the Good Proxy server.

The guide is divided into several parts:

- The hardware and software requirements detailed in GD Server/Network Specifications and Deployment Configurations
- The sizing recommendations in Good Dynamics (GD) Scalability.
- The GC and GP installation steps detailed in Installing the GD Servers and Database
- **Advanced topics:** The deployment configuration variations detailed in Variations on Deployment Configurations, including the following sections, which are also available as separate guides (see Good Dynamics Documentation for details on the complete documentation):
  - GD Direct Connect
  - How Kerberos Constrained Delegation Works
  - Server Clustering and Affinities Introduction

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.

What's New?

Good Control is Now 64-bit

The Java Runtime Engine (JRE) installed with Good Control now runs in 64-bit mode (as opposed to 32-bit formerly).

The maximum amount of memory available to Good Control can thus be increased. For larger deployments, you should consider increasing the JRE's heap size beyond the default.

Important: To take advantage of the 64-bit JRE with Good Control, you must be running the 64-bit Windows operating system, if you are not already running it. Microsoft does not provide an easy path to upgrade the word size. So you need to install the Windows 64-bit OS first and then install the latest Good Control. Do a full backup of your current 32-bit GC installation as a safety measure.

GC Sizing Recommendations: JRE Memory, DB Latency, CPUs and More

The GC server runs the 64-bit Java Runtime Engine (JRE), version 1.7.

For a small deployment, the GC, GP, and database can be installed on the same machine. For any configuration larger than a small deployment, GC should be installed on a machine with no other servers running on it. If on a virtual machine, it should have dedicated CPUs and RAM. For more discussion of common deployment configurations, see *Good Dynamics Server Installation*.

Below are approximate sizing recommendations. These recommendations for optimal performance are based on our performance testing. Good Technology realizes that in actual deployment, constraints might sometime prevent complete adherence to these recommendations.

About Max GC-to-DB Latency. Good Technology recommends the network latency between GC and its database be as low as possible. Ideally, the servers should be located in the same data center. Testing with a consistent 20 ms latency shows that some operations are twice as slow and some operations are six times slower. While this larger latency might be acceptable for smaller deployments, latency becomes more important as a deployment size grows.

Number of Containers	GC CPU Cores	GC Max Heap Memory (JvmMS and JvmMX Settings)[1]	Minimum Physical Memory	Max GC-to-DB Latency	DB CPU Cores	DB Memory
1 to 999	2	640	1.5 GB	20 ms	1	2GB
1,000 to	4	1024, default	2 GB	10 ms	2	3GB

Number of Containers	GC CPU Cores	GC Max Heap Memory (JvmMS and JvmMX Settings)[1]	Minimum Physical Memory	Max GC-to-DB Latency	DB CPU Cores	DB Memory
19,999						
20,000 to 149,999	6	2048	3 GB	5 ms	3	4GB
150,000 to 300,000	8	4096	5 GB	2 ms	4	6GB

[1] If you need to adjust the heap size for the GC, see Increasing the GC or GP Server's Java Heap Size.

Optional Notification Configuration for Performance Increase with Oracle

In addition to the database grants list above, in you are using Oracle, you might want to issue the following. This allows Good Control to receive notifications about changes to the database, instead of having to poll for such changes. The variable *username* is the name of the GC database owner; you must substitute your real name:

```
grant change notification to username;
```

GD Server/Network Specifications and Deployment Configurations

At its most basic, the Good Dynamics (GD) installation includes a database, the Good Control (GC) server and the Good Proxy (GP) server, which can all be installed on a single machine. This is the most basic deployment configuration and is suitable for proof-of-concept or demonstration but is not suitable for production use. For an overview of the common deployment configurations, see Variations on Deployment Configurations.

The following are the minimum requirements to successfully install the Good Dynamics platform servers. These requirements apply to both physical and virtual machines:

- A host machine must be set up for the GC and GP servers. These servers can be installed on a single host machine or on separate machines. Other deployment variations are discussed in Variations on Deployment Configurations
- It is highly recommended that the database reside on a separate machine from the GD servers if you are setting up a production installation. However, you can install all components on the same machine if you are setting up a development environment. See other deployment variations in Variations on Deployment Configurations.
- Persons who handle the installation, maintenance, upgrade, and uninstallation of the GD servers must have Microsoft Windows administrative privileges on all necessary machines.
- A service account or administrator account must be set up for the GC and GP servers to run as. This account must have Microsoft Windows administrative privileges on the target machine and the "Log on as a service" privilege. For more information or step-by-step instructions on how to grant this privilege, see http://technet.microsoft.com and search for "Log on as a service".
- Ensure that the Display Name field in your Active Directory service is not null for all users who will log into the GC. Without a Display Name from AD, a user cannot login to GC and receives the message: Display Name Cannot be Null.

For sizing information, see the section Good Dynamics (GD) Scalability

Minimal Server Hardware Specifications

The following are the minimum hardware requirements for the GC and GP servers.

Note: These are *minimal* and are most often used when deploying all components on a single hardware system. In production, you will need more power. Exact hardware sizing depends on the performance you need. See the GD Sizing Guide for results of performance testing to help you make this determination.

Component	Minimal Processor	Minimum RAM	Minimum Disk
Good Control	Pentium dual-core, 2 GHz	4GB. GC allocates approximately 1.5GB of RAM at start-up	100GB. For installation, a minimum of 50GB is required for the installation files and log files.
Good Proxy	Pentium dual-core, 2 GHz	4GB. The default Java Runtime Engine (JRE) heap size is 2.5GB. The recommended heap size is 60% of physical memory. You can change the heap size after installation with the instructions in Increasing the GC or GP Server's Java Heap Size.	100GB. For installation, a minimum of 50GB is required for the installation files and log files.
Database	Pentium dual-core, 2 GHz		Initial size • Data files: 2GB, which grow approximately 2GB per year. • Redo logs: 100 MB

Server and OS Software Specifications

The GD servers require one of the following operating systems, real or virtualized:

- Windows Server 2012 or Windows 2012 R2
- Windows Server 2008 or 2008 R2, 64-bit versions
- Windows 7

  Note: Although Good Technology supports Windows 7 for development and testing, do not use Windows 7 as a production platform.

The GD servers need the following network connections:

- A connection to a Microsoft Active Directory server, unless you are installing the Good Proxy in a separate domain.
- A connection to a database (see Server and port diagram for specifics)

Software Restrictions

Do not install Apache or Tomcat software on the host machine prior to the installation of the GC/GP software.

You must disable any web servers or services, such as Microsoft IIS, that use ports 80 or 443.

Synchronized Time Among Load Balancers, GC, and GP

Ensure that the GC and GP servers' and load balancers' time/date are set correctly and are in synch. In an environment when the GP is not joined to the same time source (typically the same AD domain), it is especially important to ensure the GP and GC times are in synch; otherwise the installation can fail at the certificate exchange phase. Consider using the Network Time Protocol (NTP).

Network Requirements

This section describes a standard network integration of the GC and GP servers behind the enterprise firewall.

Intranet Port Configurations

Each GD platform component uses different ports, so you must configure the host machine for each component accordingly. Make sure the following ports are open and available, and ensure that these ports are not in use by other servers or processes.

- The GC server host needs open inbound ports 443 and 17317. Port 443 is required for administrators and users to log into the GC console. The GP and GW installers connect to a GC server over port 443 during the server setup process. GP servers connect to GC servers on port 17317 during policy updates.
- The GP server host needs open inbound ports 17080 and 17433. Additionally, it should have at least 30,000 ports in the dynamic TCP port allocation, which are needed for outbound connections to the GD NOC. (When Direct Connect is configured, however, these ports become inbound.)
- The database host machine needs open inbound port 1521 open for Oracle or port 1433 open for SQL Server.

SSL Ciphers between GC and GP Servers for Direct Connect

By default, SSL communications between the GC and GP servers over port 443 for the Direct Connect configuration uses the following ciphers:

- TLS_RSA_WITH_AES_256_CBC_SHA256 OR
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

One reason you might need to add more ciphers is if you have your own proxy server between your client devices and the GP server configured for Direct Connect. This middle proxy is the one that determines which SSL ciphers to use. You need to ensure that the GP server ciphers correspond to those required by your own proxy.

If you need to add more ciphers, after installation, edit the GP server's configuration file `c:\good\gps.properties` and add the names of the ciphers to the `gps.directconnect.supported.ciphers` key. See List of Supported SSL Ciphers between GC and GP Servers for Direct Connect.

Opening and Checking Outbound Firewall Configurations

If you limit outbound requests through your enterprise firewall, you need to permit access to the following IP ranges in order for the GC and GP servers to connect to the GD Network Operations Center (NOC):

- 206.124.114.1 through 206.124.114.254 (206.124.114.0/24) on port 443
- 206.124.121.1 through 206.124.121.254 (206.124.121.0/24) on port 443
- 206.124.122.1 through 206.124.122.254 (206.124.122.0/24) on port 443

You may alternatively wish to permit access to the specific network host names:

- gdentgw.good.com on port 443
- gdrelay.good.com on port 443
- gdweb.good.com on port 443
- gdmdc.good.com on port 443
- bxenroll.good.com on port 443
- bxcheckin.good.com on port 443

If you make connections through a web proxy server, please make sure to enter the proxy information in both the GC and GP installers when asked to do so.

Note that no inbound ports through the enterprise firewall are required for the Good Dynamics platform.

The following diagram details the ports and connections between the components of the GD platform. Keep the following in mind as you read the diagram:

- All connections are TCP, not UDP.
- Arrows originate at the point from which communications are established. The direction of the arrows neither reflects the flow of data nor the end which initiates commands.
- The selection of high or low port numbers for clients connecting to Good Technology NOC servers is configurable for each enterprise.
- "Secure Communication" refers to data that is sent by using the GD Socket and GD HTTP Request APIs.

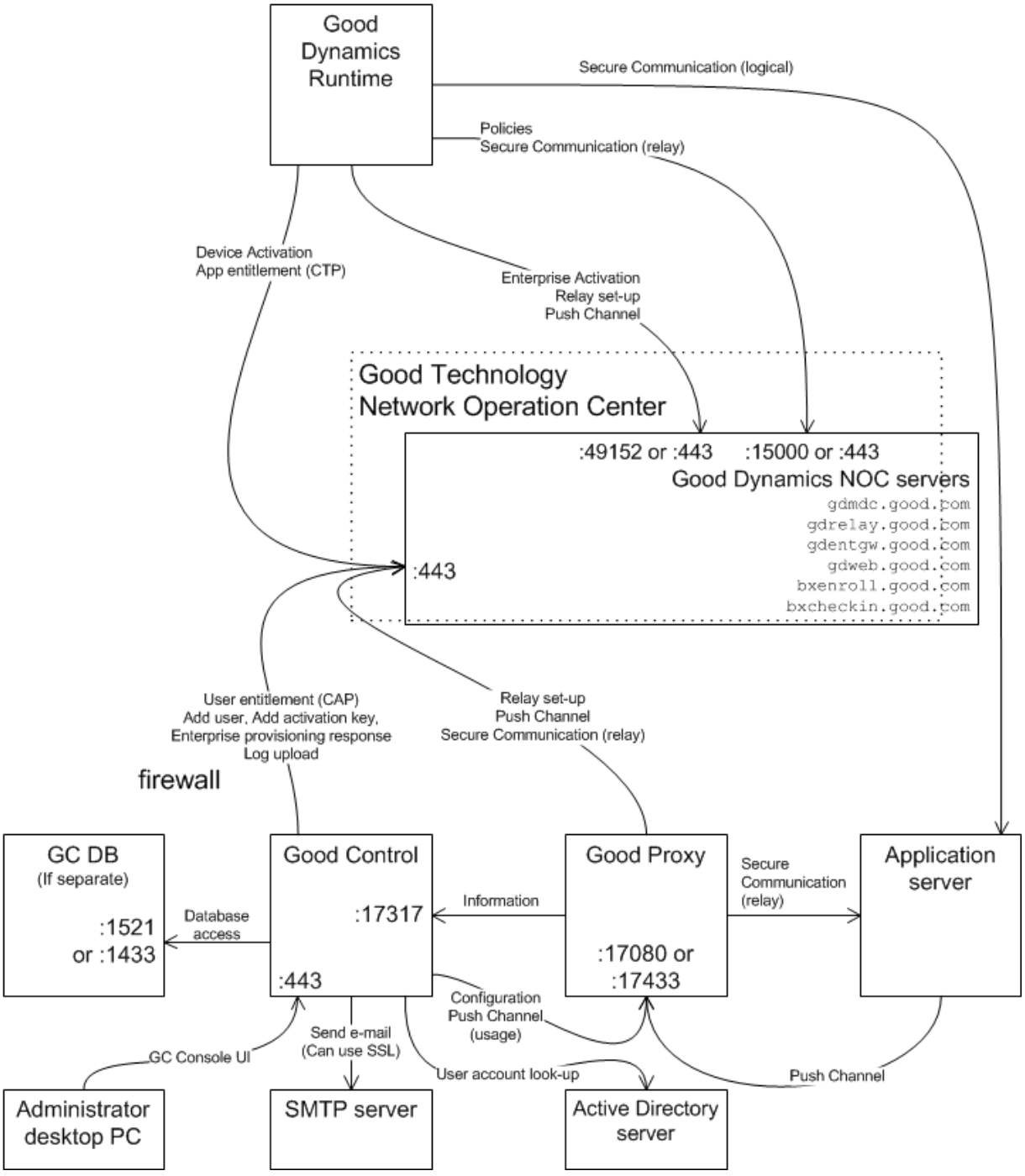


Figure 1: Server and port diagram

Microsoft Port Query Tool

You might want to verify the outbound connections with a network tool like Microsoft PortQry command-line tool or other networking utility to be sure that your firewalls are configured properly.

Latency between GC and GC Database: Maximum 10 ms

Make sure that the latency on the network connecting the Good Control server and its associated database is no greater than 10 milliseconds.

Preparing the Database

A database must be installed and prepared before you start the GC installation. GC supports the following database software:

- All editions of Microsoft SQL Server 2012
- All editions of Microsoft SQL Server 2008 and 2008 R2
- The Enterprise or Standard edition of Oracle 10g/11g for production installations
- The Express edition of Oracle 10g/11g for development installations

Note: Oracle 10g/11g XE supports up to 2 GC servers in a server cluster. Additionally, Oracle 10g XE is not compatible with Windows Server 2008 R2.

Email Server Configuration Requirements

The GD platform depends on the proper configuration of other software, such as Microsoft Exchange, which is not subject to the direct control of the GD installation software.

If you are using Exchange, configure your server to include a dedicated receive connector for the GC server's IP address.

Browser Recommendations

You use a web browser to access the GC console. The latest versions of the following browsers are supported:

- Mozilla Firefox 36.x or later
- Google Chrome 40.x or later
- Microsoft Internet Explorer 9.x or later
- Apple Safari 6.2.3; 7.1.3; or 8.0.3

The following browsers are not supported:

- Opera

Good Dynamics (GD) Scalability

This guide should give you sufficient information to help you determine the most efficient configuration of system hardware based on the design of the GD server software.

The scalability of the various GD components can vary depending on hardware and network configuration. The metrics described here are based on a hardware benchmark designed to accurately determine the capacity and performance of GD servers. Confidence is high that this benchmark reflects the server-scale processor performance typically employed in enterprise systems belonging to the same class as Good Dynamics.

The design of the GD benchmark unit is neither a recommendation nor an endorsement of that design for use in production. Moreover, as stated earlier, the actual performance of a given server can vary depending on the processor selected, the amount of RAM, the network configuration, and other factors.

Common Measures of Scalability

Scalability is among GD's most important benefits because it means the system can be deployed with the confidence that it will not be outgrown.

Scalability can be measured in various dimensions. Some common measures of scalability are:

- **Administrative:** Enabling an increasing number of organizations and/or users to easily share the system.
- **Functional:** The ability to enhance the system by adding new functionality at minimal effort.
- **Geographic:** The ability to maintain performance, usefulness, or usability regardless of expansion from concentration in a local area to a more distributed geographic pattern.
- **Load:** Enabling the distributed system to easily expand and contract its resource pool to accommodate heavier or lighter loads or number of inputs. Alternatively, it is the ease with which the system or its components can be modified, added, or removed to accommodate changing demand.

Sizing Recommendations for a Large Deployment

The section GC Database Scalability defines small, medium, large, and extra large deployments based on number of containers.

Note: The sizing recommendations in this document are for a large deployment of Good Dynamics.

Background: GD Components

GD consists of three core components. This section briefly describes some typical deployment configurations and some minimal sizing of the components. For more on the setup and operation the GD components, see the GD Server Installation Guide. For additional details on infrastructure strategies, see Server Clustering and Affinities.

1. The Good Control (GC) server is the Good Dynamics management and configuration component. Every GD enterprise has its own GC server. The functions of the GC include user, policy, application and role management, and infrastructure configuration. Multiple GC servers can be installed for high availability and disaster recovery (HA/DR).

   Depending on connectivity requirements, a typical installation consists of a single GC server and one or more GP servers. Where appropriate, multiple GC servers can be deployed to accommodate the scenarios described briefly below:

   - **Topology:** For instance, an enterprise might want no connectivity between environments or "zones".
   - **Control**: Each zone might want to maintain control over its own user and application policies.
   - **Scale and redundancy**: Different zones might have different numbers of users or HA/DR requirements.

2. The Good Proxy (GP) server is the component that maintains the secure connection between an enterprise and the NOC. The GP server is installed behind the enterprise firewall and establishes the secure connection outward to the NOC. This means there is no need to open an inbound port on the firewall and no need to use a VPN. Multiple GP servers can be installed, for high availability and disaster recovery (HA/DR), or to add more relay capacity.
3. **The Good Control Database** is the repository used by GC to store user, application, and policy information. Both Microsoft SQLServer and Oracle are supported.

   It is perfectly acceptable to share this database on an existing DB server, especially in environments where Good for Enterprise (GFE) with Good Mobile Control (GMC) and Good Share are deployed. In fact, running multiple Good product databases on a common SQL database server yields significant savings in total cost of ownership.

Good Control Scalability

GC scalability is driven by two primary factors:

- Number of users being administered, discussed in User Scalability.
- Number of “containers” being administered, discussed in Secure Container Scalability and Clustered Servers Scalability.

A fundamental fact is that a single hardware server can process only a finite number of users and containers. If that limit is exceeded, additional capacity is required.

User Scalability

The number of users that can be administered by a single GC server is not limited.

However, the response time of certain elements of the GC console UI is directly correlated to the number of users and containers administered by GC.

Testing by Good Technology has proven that with 100,000 users, the Good Control console UI is adequately responsive.

Secure Container Scalability and Application Policy Updates

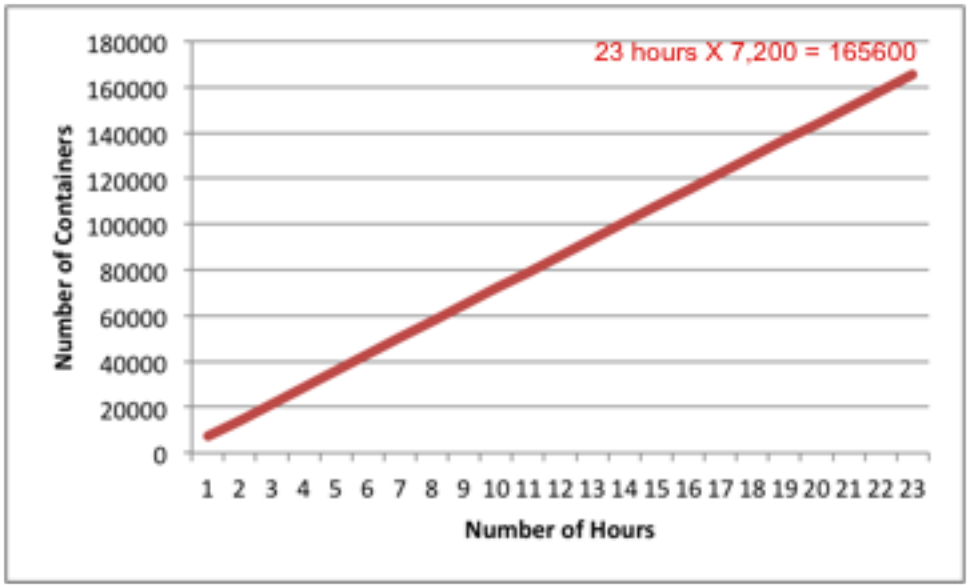
Any single GD-SDK-based application on a device represents a single container. Thus, if a user has five GD-SDK-based applications installed on a single device, there are five containers. If a single user has two GD-SDK-based applications installed on six devices, there are 12 distinct containers in total.

When a GD administrator changes an application policy (for instance, setting a security policy to enable data loss prevention), GC queues that policy for download. As the corresponding applications on client devices authenticate with the GC server, they download any new policies. For policy updates, Good Control accepts connections from applications on client devices at the maximum rate of two connections per second. This throttling is in place to keep GC serviceable in the event that hundreds of thousands of applications attempt to connect at the same time. In all cases when the GC server is performing at two connections per second, any additional applications requesting connectivity must retry later.

Consequently, at a rate of two connections per second, or 7,200 connection per hour, the scalability curve is a constant slope. The determining factor is simply what you accept as response time for an application policy update on all client devices on which the application is running.

Thus, you must decide what constitutes an acceptable update response time for your enterprise. For example, a corporate endpoint antivirus application on 300,000 workstations might take 48 hours to deploy and update. Create a similar policy for your mobile applications so you can determine the scalability of your infrastructure and a reasonable expectation of when all policy updates will be completed.

The graph below plots the linear scalability of a single Good Control server by time in hours required to update all container policies.



Tuning Container-related Performance with GC Server Properties

Good Control has many container-performance-related properties that can be configured to tune or adjust performance. These properties are documented in the topic **Servers > Basic Server Settings > GC Property Reference** in the Good Control online help and the printable guide of the help at *Good Control Online Help*.

Note: Good Technology recommends that you do not modify these settings without consulting Good Technology technical support. Misunderstandings of these settings can adversely impact performance.

Clustered Servers Scalability

GD supports the clustering of server components together to form high availability, highly scalable systems. GD clustering requires no Microsoft Clustering or third-party components. For more on GD's HA/DR capabilities in general, see the Good Dynamics feature overview on Server Clustering and Affinities.

GD clustering follows an active-active model: all servers actively participate all the time. If a particular server goes down, its workload is redistributed automatically to the other servers in the cluster. Strongly recommended, the clustering benefits of GD not only provide for highly scalable enterprise deployments, but a dependably fault-tolerant operational environment as well.

Good Proxy Scalability

The GP server runs the 64-bit Java Runtime Engine (JRE), version 1.7.

Tests were run on Microsoft Windows 2008 R2.

These tests have shown that a GP server running on Xeon E5606 with four 2.13 GHz cores and eight GB of RAM supports up to 15,000 connections. In our tests, Good Technology observed a throughput rate of approximately 15 Mbps (megabits per second).

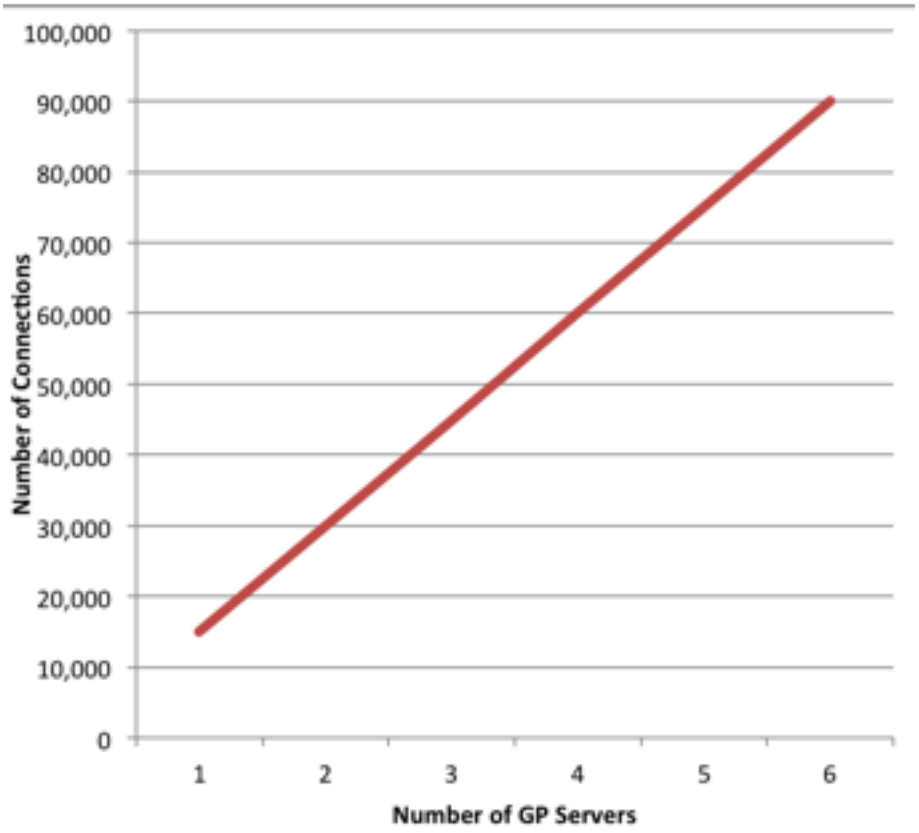
Because the GP server's architecture scales horizontally, every additional GP server can increase the number of concurrent connections by 15,000. (For more information on clustering, see Server Clustering and Affinities.) Multi-server capacities thus equate to the following numbers:

- 2 GP servers support a maximum of 30,000 concurrent connections.
- 3 GP servers support a maximum of 45,000 concurrent connections, and so on.

Some observations:

- In general the lower the number of connections the higher the throughput.
- The network bandwidth for your deployment depends on the nature and characteristics of the workload of the applications. The larger the amount of data, the larger the bandwidth needed.

The following is a graph of the horizontal scaling of the GP server.



Increasing Number of Sockets on MS Windows

Important: For GP servers on Microsoft Windows, you must increase the range of sockets (called "dynamic port range for TCP"). Good recommends that you allocate at least 50,000 ports. The default range allows only 16,384 dynamic ports, which is not sufficient for the Good Proxy, which requires two ports for each connection (one to the device and one to the application server). Without this setting, connections might be blocked when the GP runs out of available sockets.

For specific commands to configure Microsoft Windows, see the Microsoft article at http://support.microsoft.com/kb/929851.

Note: The exact for parameters for the `netsh` command's start and num parameters depend on your precise system configuration and which ports on the server might already be allocated. The values shown below are only examples:

- `start=15000`
- `num=50000`

GC Database Scalability

The best way to size a GD database deployment is based onthe projected number of containers. (For background, see Secure Container Scalability and Application Policy Updates

Four deployment sizes are defined:

- Small Deployment: Less than 10,000 managed containers
- Medium Deployment: Typically between 10,000 to 30,000 managed containers
- Large Deployment: Typically between 30,000 to 150,000 managed containers
- Extra Large Deployment: Typically between 150,000 up to 300,000 managed containers

Summary of Recommendations

Good Technology estimates that initial database for a large GD deployment requires 2 GB for data files and 100 MB for redo log and estimates that data files grow by 2 GB a year.

Good Technology recommends that 10 GB of disk space be allocated for data files. Good Technology has experienced adequate response time with 7200 RPM disks.

Installing the GD Servers and Database

Log on to the Good Developer Network (GDN) portal. If you do not have an active account, click **Login** or **Register** to sign up.

The following topics explain how to download, install and configure the GD server side components and how to find and download the GD SDK:

- Pre-installation Checklist
- Installing the First Good Control Server in Server Cluster
- Installing Additional Good Control Server in Server Cluster
- Installing Good Proxy Server

Upgrade or Install GC and GP In Parallel

Whenever a new version of Good Control is released, it is accompanied by a new version of Good Proxy. Likewise, a new release of Good Proxy always has a companion new release of Good Control. The two servers have mutual support for common features.

The installed versions of Good Control and Good Proxy must be kept in synch. You should always install the latest Good Proxy that accompanies the latest Good Control and vice versa.

Installing the Good Control Database

As part of the database setup process, you need to create an account and set a password for the GC database user. If you do not have corporate security policies that govern the strength of account passwords, we recommend the following minimum requirements for password strength:

- At least 8 characters in length
- At least one numeric character
- At least one special character
- No character used more than twice

See the GC online help for instructions on how to change the database user password after installation is complete.

Setting Up Oracle XE Database

The following instructions explain how to set up an Oracle XE 10g or 11g database user for GC.

1. Start the Run SQL Command Line program:
   Start Menu > All Programs > Oracle Database Express Edition > Run SQL Command Line.

   Enter `connect system`. When prompted, provide the `system` user's password.

2. Run the following command:

```
create user gc_db identified by password;
grant connect to gc_db;
grant resource to gc_db;
alter user gc_db default role all;
alter user gc_db default tablespace USERS;
```

This creates the `gc_db` user with a password of *password*. If you want to use a stronger password, replace *password* with a stronger value when you run this command. You can set any password for this user, but do keep in mind that you will need to enter the password value correctly when the GC installer asks for GC database information.

Optional Caching Configuration for Performance Increase

In addition to the database grants list above, you might want to issue the following. This allows Good Control to receive notifications about changes to the database, instead of having to poll for such changes:

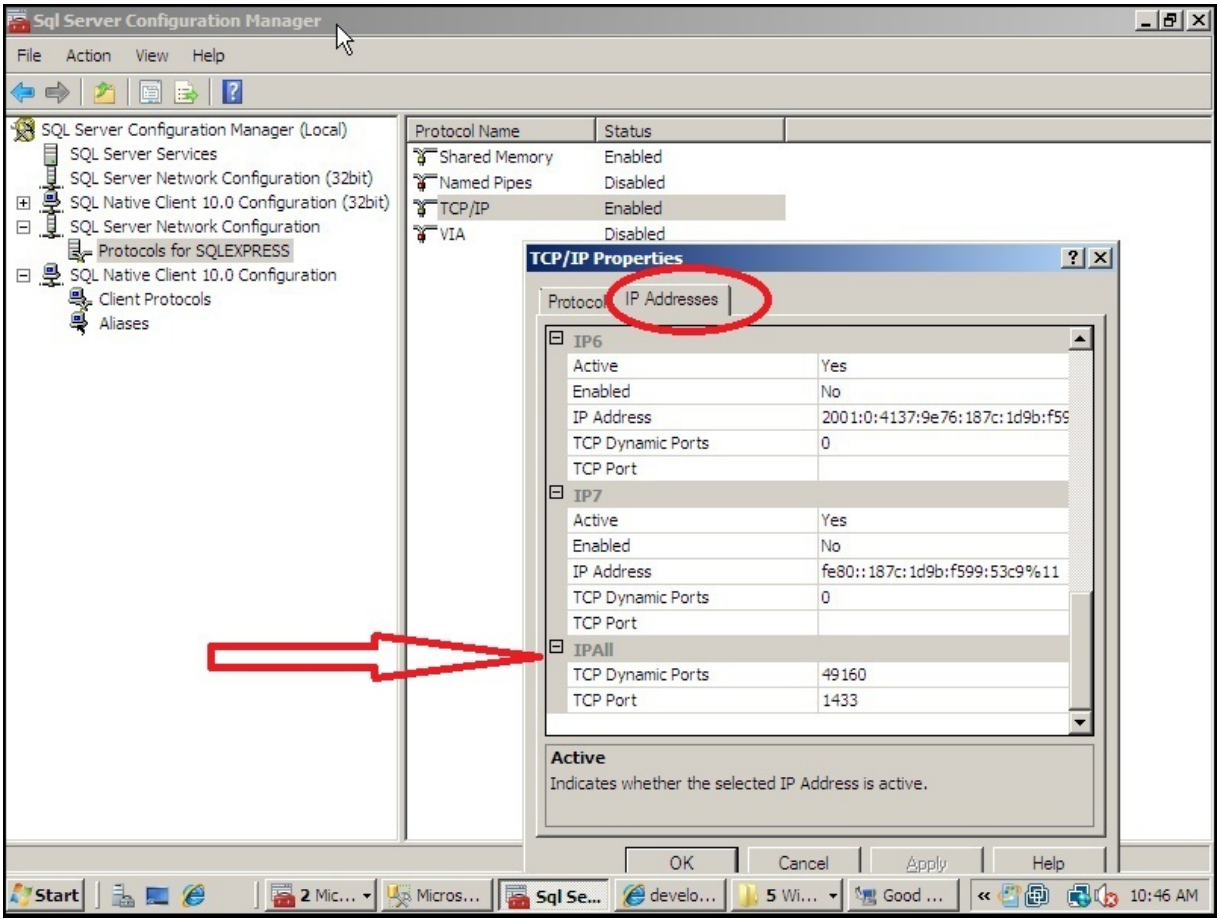
```
grant change notification to $USER;
```

Setting Up SQL Server Database

SQL Server Management Studio is required for GC database setup; if your SQL Server installation does not include the SQL Server Management Studio software, it is available as a separate download from the Microsoft web site.

The following instructions explain how to set up a SQL Server database for GC:

1. Install the SQL Server database per the directions in the installation wizard. You can specify either "Windows Authentication mode" or "SQL Server and Windows Authentication mode" under the Security section of the Server Properties.
2. After installation, launch SQL Server Management Studio and log in. You will perform steps 3 and 4 through the SQL Server Management Studio console.
3. Set up the login that will be used to manage the GC database. Expand the Security item in the Object Explorer pane, then right-click **Logins** and select **New Login**.
   a. If you selected "SQL Server and Windows Authentication mode" in the Server Properties and wish to have a SQL Server login to manage the GC database, enter gc_db as the Login name. Select **SQL Server authentication** and set a password for this login. Keep in mind that you will need to enter the password value correctly when the GC installer asks for GC database information. Click **OK** to add the login.
   b. If you want to use a Windows account to manage the database, select "Windows authentication". Enter the Windows account username in domain\username format as the Login name. This account should be the same as the service or administrator account set up to run the GC service. Click **OK** to add the login.
4. Right-click the **Databases** item in the Object Explorer pane, then select **New Database**. Enter gc as the Database name and set the login you configured in the previous step as the database Owner. Click **OK** to add the database.
5. Launch the SQL Server Configuration Manager:
   Start > All Programs > Microsoft SQL Server 2008 > Configuration Tools > SQL Server Configuration Manager.
6. Select Protocols for SQLEXPRESS. Enable TCP/IP and add port 1433 for IPAll.



7. Restart the SQL Server service.

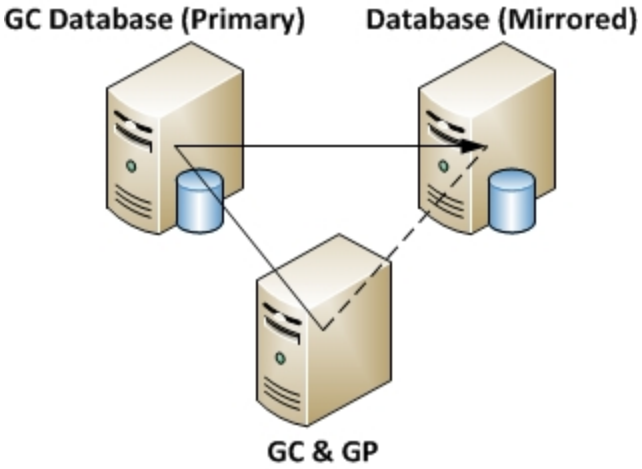
Optional: JDBC Connection String for SQL Mirroring

By default, Good Control is not configured for SQL mirroring, which is an optional deployment configuration. In a Good Dynamics deployment with SQL mirroring, when configured properly, Good Control automatically switches between the primary and mirrored DB when there is a failure on the primary DB. The SQL server used for mirroring (called the `failoverPartner`) must be configured in the Good Control JDBC (Java Data Base Connection) connection string. If the `failoverPartner` is not configured, Good Control cannot failover to the mirrored database.

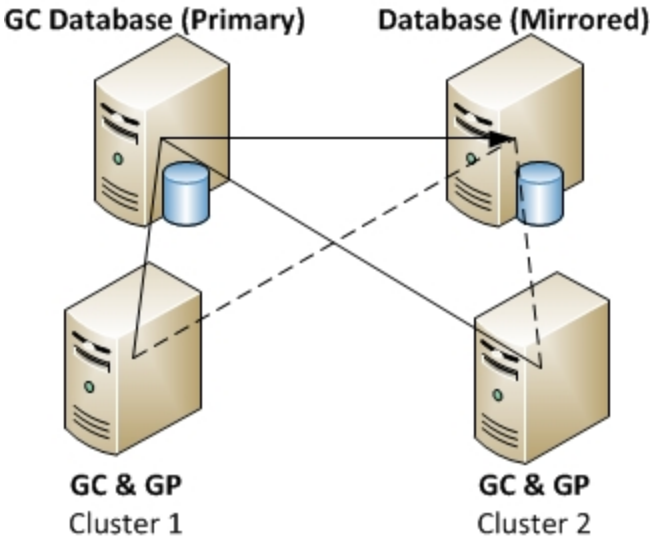
Your database software must be configured for mirroring; GC controls only the connections to the database in case of failover, not the mirroring itself.

SQL mirroring is an active/passive technology; the primary DB is active and the mirrored DB is passive. Reading and writing to a mirrored DB is forbidden by SQL. As such, all Good Control servers must point to the principal SQL DB as the primary and the mirrored DB as the failover.

Here is a logical view of the GC connections to the database with SQL mirroring:



With clustered GCs, a logical view of the GC-to-database connections with SQL mirroring is as follows:



Configuring the JDBC connection string

To enable mirroring, you need to change the value of the `db.connection.url` key in the GC server's `C:\good\gc-server.properties` file to include the `failoverPartner` parameters (the details of the mirroring SQL server).

Here is an example of the default `db.connection.url` value:

```
db.connection.url=jdbc\:sqlserver\://
server1
\\\\MSSQLSERVER\:1433;databaseName\=gc;selectMethod\=cursor;integratedSecurity\=true
```

Here is the changed value to include the `failoverPartner` parameters (the mirroring SQL server):

```
db.connection.url=jdbc\:sqlserver\:/
server1
\\\\MSSQLSERVER\:1433;databaseName\=gc;
failoverPartner\=
server2\\\\MSSQLSERVER\:1433;selectMethod\=cursor;integratedSecurity\=true
```

Notes:

- The example strings above are all on a single line.
- Be sure to use the backslash \ to escape the special characters : \ and = as shown in the examples: \: \\ and \=. For two backslashes, you need four: \\\\.
- Be sure to substitute your own server names and configured port numbers for the *server1* and *server2* shown in the example.
- Update this value on all Good Control servers that are SQL-mirroring.
- Restart all Good Control servers for changes to take effect.

Migrating the Good Control Database

Good Control supports databases from several different vendors. However, Good recommends that you migrate only among different database servers from the same vendor, such as Microsoft SQL Server to Microsoft SQL Server, not across vendors such as Microsoft SQL Server to Oracle. Migrating across different types of databases requires more extensive work than is discussed here.

Do Not Change the GC Login Authentication Method

When you migrate from one database to another, do not change the authentication method used for logging in to Good Control. For instance, if the first database was set up to use SQL authentication for login, do not switch to use Windows authentication in the new database.

About Migrating the Database of a GC Cluster

Important: If you are running a cluster of Good Control servers, you must:

- Shut down all GC servers in the cluster before migrating the database.
- Follow the procedures detailed here for all GC servers in the cluster.
- After finishing the procedures, restart all GC servers in the cluster.

GC's Server Properties File and Database Connections

The property file **C:\good\gc-servers.properties** includes database-related properties.

Important: You should make a backup copy of this file before you change the original.

After migrating your data, be sure to verify the following settings.

1. DB connection string. This is the value of `db.connection.url` property, as shown in the examples below.
2. DB user/owner. This is the value of the `db.connection.user` property, as shown in the examples below.

gc-servers.properties File with Details for Oracle Database

Note: In this example of `gc-servers.properties`, in the value of the `db.connection.url` property, the host name is `localhost`: `(HOST\=localhost)`. However, if your database is installed on a machine separate from your Good Control server, make sure you use the fully qualified domain name of your database server so your GC server can resolve the hostname of the database server.

Also make sure that the value of `SERVICE_NAME` (`GC`, in this example) matches what is required by your Oracle instance.

```
db.connection.url=jdbc\:oracle\:thin\:@(DESCRIPTION\=(ADDRESS\=(PROTOCOL\=tcp)(HOST\=localhost)
(PORT\=1521))(CONNECT_DATA\=(SERVICE_NAME\=GC)))
db.connection.user=gc_db
db.dialect=org.hibernate.dialect.Oracle10gDialect
db.driver=oracle.jdbc.driver.OracleDriver
gd.product.hostname=gc-server-123
gd.product.licensekey=5649-8E49-C9C7-C1D7-78EF-116B
gd.product.serialnum=GD1000001
```

gc-servers.properties File with Details for Microsoft SQL Server Database

Note: In this example of `gc-servers.properties`, in the value of the `db.connection.url` property, the host name is `localhost`: `sqlserver\://localhost`. However, if your database is installed on a machine separate from your Good Control server, make sure you use the fully qualified domain name of your database server so your GC server can resolve the hostname of the database server.

Also make sure that the value of `databaseName` (`gc`, in this example) matches name of your Oracle database.

```
db.-
con-
nection.url=jdbc\:sqlserver\://localhost\\\\SQLExpress\:1433;databaseName\=gc;selectMethod\=cursor

db.connection.user=gc_db
db.dialect=org.hibernate.dialect.SQLServerDialect
db.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
gd.product.hostname=gc-server-456
gd.product.licensekey=1A51-6D26-8469-6AEF-6361-7298
gd.product.serialnum=GD1000002
```

Changing the Database Password

Depending on how it was set up, your new database might have a different password than was set for the database you are migrating from.

Good supplies a script to update the password that Good Control needs to access the database. This script prompts you for the new database password, obfuscates the password, and then stores the obfuscated password for best security.

To change the database password:

1. On the Good Control server host machine, open a command window as administrator and enter the following command:

   gc_install_dir\tools\password\changepwd.bat
2. You are prompted to select which password you want to change:
   - Enter: **GC_DB**
   - Enter the new password.
3. For changes to these properties to take effect, you need to restart the Good Control service. Go to the **Services** window.
4. Select the GC server from the list of services.
5. Click the **Start** command.

Verification

Good recommends that with either SQL commands or Microsoft SQL Server management console, log in to the applicable database with the GC database user and password to verify that the account has full access.

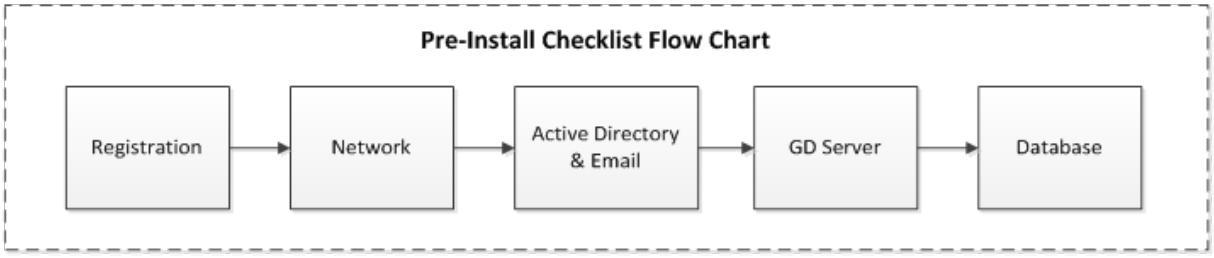
Possible Errors

If the above issues have not been addressed, the following errors might be recorded in the GC server logs when the GC is started.

- `org.hibernate.exception.GenericJDBCException: Cannot open connection at org.hibernate.exception.SQLStateConverter.handledNonSpecificException (SQLStateConverter.java:140)`
- `java.sql.SQLException: Connections could not be acquired from the underlying database! at com.mchange.v2.sql.SqlUtils.toSQLException(SqlUtils.java:106)`
- `com.mchange.v2.resourcepool.CannotAcquireResourceException: A ResourcePool could not acquire a resource from its primary factory or source.`

Pre-installation Checklist

It is highly recommended that the following checklist be completed before implementation takes place. This checklist is meant for a POC install of Good Dynamics (GD).



	Registration	Check
1.1	Register with the GDN Portal.	☐
1.2	Download software from the GDN (Good Control & Good Proxy)	☐
1.3	Create your Good Control license on the GDN.	☐
1.4	Request your trial Apps.	☐

	Network, Server Hardware, Server Software	Check
2.1	Double-check the following list against the canonical server port document at https://community.good.com/docs/DOC-1121 Ensure that the GD server has outbound (egress) access to the Good NOC on TCP port 443. The Good NOC has the following IP ranges: • 206.124.114.1 through 206.124.114.254 (206.124.114.0/24) on port 443 • 206.124.121.1 through 206.124.121.254 (206.124.121.0/24) on port 443 • 206.124.122.1 through 206.124.122.254 (206.124.122.0/24) on port 443 You may alternatively wish to permit access to the specific network host names: • gdentgw.good.com on port 443 • gdrelay.good.com on port 443 • gdweb.good.com on port 443 • gdmdc.good.com on port 443 • bxenroll.good.com on port 443 • bxcheckin.good.com on port 443 You should probably verify that these ports are open with a network utility like Microsoft PortQry or Microsoft's version of traceroute.	☐

	Network, Server Hardware, Server Software	Check
2.2	If the Good server requires a Proxy server for external access. Please note the following: Proxy Server make/model: Authentication method:	☐
2.3	Ensure that the GD server has access to the database server if it is remote. The default port for MS SQL is TCP 1433. **Note:** The port must be static; GC does not support dynamic SQL port connections.	☐
2.4	Make sure that the Domain Name System (DNS) has been properly setup on the GD servers themselves and that the DNS properly resolves the servers' fully qualified domain names (FQDNs): • You must use the fully qualified domain name for the server, like `hostname.somedomain.com`. Do *not* use just the bare `hostname`. • The FQDN must be an A record in your DNS; that is, a canonical address record for this server. You must not use a DNS alias.	☐
2.5	Minimum server hardware for Proof-of-Concept (more for production), as follows: • Good Control: Pentium 2GHz dual core, 4GB RAM, 100GB disk • Good Proxy: Pentium 2GHz dual core, 4GB RAM, 100GB disk • Database: Pentium 2GHz dual core, 4GB RAM, 10GB disk If you want to install all software (GC, GP, and database) on a single machine for POC, ensure the following: • Quad core / 2.4 GHz CPU or higher • 16 GB RAM / 100 GB HDD • 100/1000 Ethernet Card	☐
2.6	The GD servers require one of the following operating systems, real or virtualized: • Windows Server 2012 or Windows 2012 R2 • Windows Server 2008 or 2008 R2, 64-bit versions • Windows 7 **Note:** Although Good Technology supports Windows 7 for development and testing, do not use Windows 7 as a production platform.	☐

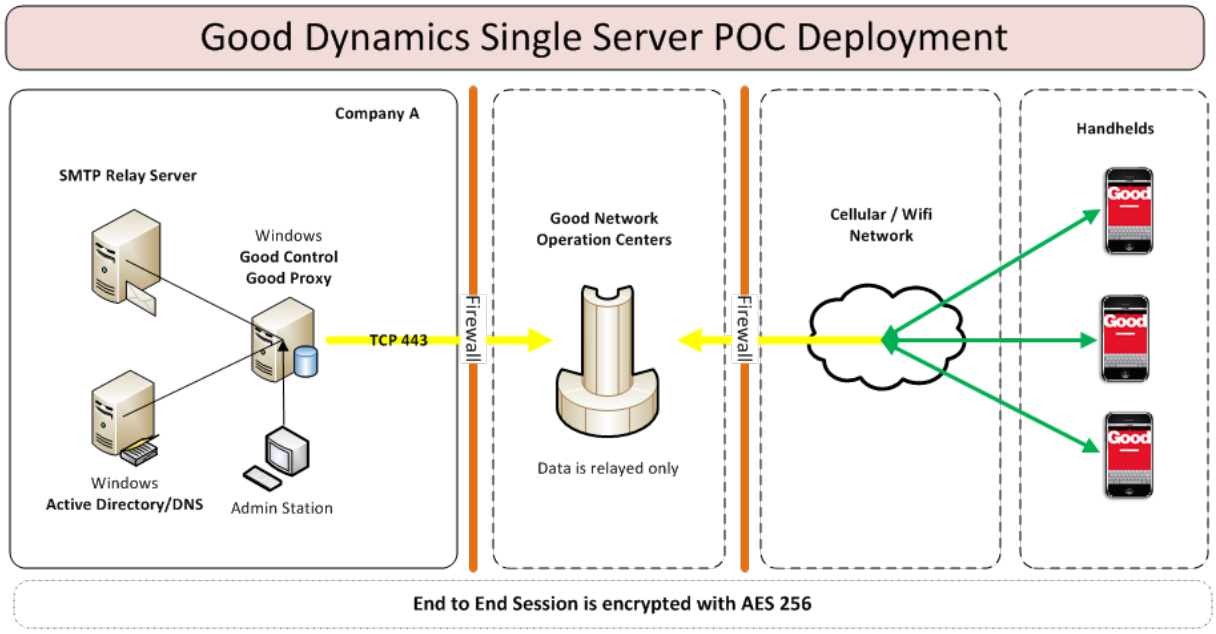
	Active Directory and Email	Check
3.1	Create an AD service account for the Good Dynamics software, or if the GP is to be installed without AD, create a local MS Windows administrative service account.	☐
3.2	The Good Dynamics server needs to send notification emails. Make sure SMTP relay is enabled for the Good Dynamics server on your email server.	☐

	Good Dynamics Server: MS Windows System Administration	Check
4.1	Verify OS support. Windows Server 2008, 2008 R2 (32 or 64 bit) & 2012 is supported.	☐
4.3	Disable TCP auto tuning. Recommended if large file transfers are expected. "Run as Administrator" the following CMD command to disable TCP auto tuning: `netsh interface tcp set global autotuninglevel=disabled` A restart of the server is required for the setting to take effect.	☐
4.4	Ensure that the GD Service account is a member of the local administrator group on the server	☐
4.5	Ensure that the GD Service account has “Logon As a Service” right	☐
4.6	Ensure that the GC and GP servers' and load balancers' time/date are set correctly and are in synch. In an environment when the GP is not joined to the same time source (typically the same AD domain), it is especially important to ensure the GP and GC times are in synch; otherwise the installation can fail at the certificate exchange phase. Consider using the Network Time Protocol (NTP).	☐
4.7	Ensure that the server has been joined to the AD domain, unless you plan on installing the GC and GP servers in separate domains (the "GP as workgroup only" option). For GP as workgroup only, in the GP server's TCP/IP network setup, set the DNS suffix to `local`.	☐
4.8	Ensure that the Windows firewall is off	☐
4.9	Ensure Antivirus/backup and backup software are stopped during the install	☐
4.10	Ensure TCP port 80 and 443 are not already in use on the server.	☐
4.11	Ensure that all GD software is installed with the GD service account.	☐

	Database	Check
5.1	Verify Database server support. The following database servers are supported: • All editions of MS SQL Server 2012	☐

	Database	Check
	• All editions of MS SQL Server 2008 & 2008 R2 • MS SQL Express 2008 R2 with Management Tools • Enterprise or Standard edition of Oracle 10g/11g • Express Edition of Oracle 10g/11g For your convenience, here is the link to download MS SQL Express http://www.microsoft.com/en-us/download/details.aspx?id=23650".	
5.2	Manually create a database for the Good Dynamics software on the database server. Default attributes are sufficient. The name of the database is arbitrary, but "GC" is recommended. This must be done prior to the install (very important!)	☐
5.3	Ensure that the GD service is the actual owner of the GD database, either a service account or SQL account.	☐

Architecture Diagram



Installing the First Good Control Server in Server Cluster

Here are the steps for installing Good Control on the first server in a cluster. The steps for remaining servers are described in Installing Additional Good Control Server in Server Cluster.

Required: Install or Upgrade All Servers in the Cluster

You must install or upgrade all the servers in a cluster to the same version of Good Control before resuming operations. Servers in a cluster that are not upgraded will be in an inconsistent state which can cause problems in operation.

Steps

To install the GC server, make sure you have the following:

- The GC installer (**gcsetup.exe**), which is available through the GDN portal.
- A License Key and Serial Number, which are available through the Licenses & Servers section on the GDN portal.
- Your Active Directory domain information.
- The fully qualified domain name (FQDN) of the GC server itself, which is a DNS A (or "address") record, not a DNS alias. For example, not just the bare `hostname` but `hostname.domain.com`.
- An administrator or service account set up on the host machine to run the GC service.
- Your SMTP server's connection information.
- Your database connection information.

This section provides information on the installation steps.

1. Run **gcsetup.exe** to begin the installation.
2. **Introduction** screen
3. **Good License Agreement** screen
4. **Third Party License Information** screen
5. **Host Information**: verify that the hostname and fully qualified Internet domain name (FQDN) displayed in the panel is correct for the target machine, like `gc.mydomain.com` (for example):
   - The hostname must not be an IP address.
   - The fully qualified domain name must be a DNS A record (canonical "address" record), not a DNS alias.
   - Only valid Internet domain names allowed, in conformance with RFC 1123 (https://tools.ietf.org/html/rfc1123).
   - No wildcard, control, or special characters allowed.
   - No underscores allowed.

Choose either to accept or to modify the values displayed by the installer.

6. **Choose Install Folder** screen: Accept the default, which is `C:\Program Files\Good Technology\Good Control`, or specify the desired folder path.
7. **Choose Logs Folder** screen: make sure that the directory you specify is writable only by the service account installing the software. For security, the logs folder must not writable by anyone else.

   Note: If you are installing multiple GC or GP servers, such as in a cluster, each server must have a unique folder for logging. The servers must not share the same folder.
8. **Proxy Information** screen: the installer attempts to detect whether or not the host machine uses a proxy server to access the Internet. Follow the installer prompts as appropriate for your environment, and enter any additional requested information.
9. **Administrator Information** screen: enter the user name, domain, and password of the administrator account which will be used to run the GC server service.

   The administrator must be a valid user in Active Directory. This user must be in the administrator group for the machine that will host the GC server and must have the "Log on as a service" privilege. This account is used to run the GC server as a Windows service and also serves as the first administrator login for the GC server.

   By default, only users within the administrator account's Active Directory domain can be given GC accounts: however, you can whitelist additional domains after the server is installed. See the Basic Server Settings section of the GC console's built-in help for complete instructions.
10. **Database Information** screen: select the database type and enter the connection information.

    Choose Oracle or SQL Server for the database type. Select the Advanced connection type to provide the full connection string or select the Basic connection type and provide the host (fully qualified Internet domain name or bare hostname as long as the GC server can connect), service and port for your database connection. If you are using a database other than the default, you can also specify the database name, like this (for example): `mysql.mydomain.com\SQLEXPRESS`. Enter the GC database username and password that you created previously per the instruction in Installing the Good Control Database.

    If you are using an Enterprise database for GC, you can contact your DBA for connection information.
11. **Domain Information** screen: check the "Use trusted domains" box if you need to add users to GC from additional domains.
12. **Mailbox Information** screen: select the option which describes how your Active Directory and email services are set up in your environment.
13. **Registration Information** screen: provide the license information for the new GC server.

    Enter your company name and the Serial Number and License Key that you acquired from the Good Developer Network.

Note: Each license you generate on the Good Developer Network will allow you to deploy one GC server cluster. You enter this license on this screen, during the installation of the first GC server in the cluster. Once a license has been used to deploy a server cluster, it cannot be used to deploy additional server clusters.

12. **SMTP Information** screen: enter the SMTP server information.

    In most cases, the SMTP server is your Microsoft Exchange server. GC uses the SMTP server to send welcome emails and provision emails to users. Users activate GD applications on their devices using the access keys contained in provision emails, so it is important to make sure the SMTP connection information is correct.

13. **Pre-Installation Summary** screen: review the information and if everything is correct, click Install.
14. **Certificate Information prompt**: review the certificate generated by the installer.
15. **Install Complete** screen: click **Done** to quit the installer.
    The installation logs are located in the C:\good\ialogs directory.
16. You can access the GC console by pointing your browser to https://localhost/ or https://*server_name* on the GC server host machine.

    The GC console login screen appears, and you can log in using the account credentials you supplied to the installer in step 8.

    Also, browse to https://*server_name* from another machine on your enterprise network to verify that you can access the GC server. Make sure your firewall settings allow incoming connections from domain networks.

    If the GC console does not load in your browser, check the status of the Good Control Windows service, and start the service if it is not already running. The GC server logs are located in the C:\good\gclogs directory.

Installing Additional Good Control Server in Server Cluster

In order to successfully install the GC server, make sure you have the following:

- The GC installer (**gcsetup.exe**), which is available through the GDN portal
- A host machine that belongs to the same domain as the other GC servers in the cluster
- An administrator or service account, already added to the list of GC administrators, set up on the host machine to run the GC service
- Your SMTP server's connection information
- Connection information for the database used by other GC servers in the cluster
- A GC server license which was generated through the console of a GC server that already resides in the cluster

This section provides information on the various installation steps.

1. Run **gcsetup.exe** to begin the installation.
2. **Introduction** screen
3. **Good License Agreement** screen

4. **Third Party License Information** screen
5. **Host Information**: verify that the hostname and fully qualified Internet domain name (FQDN) displayed in the panel is correct for the target machine, like `gc.mydomain.com` (for example):
   - The hostname must not be an IP address.
   - The fully qualified domain name must be a DNS A record (canonical "address" record), not a DNS alias.
   - Only valid Internet domain names allowed, in conformance with RFC 1123 (https://tools.ietf.org/html/rfc1123).
   - No wildcard, control, or special characters allowed.
   - No underscores allowed.

   Choose either to accept or to modify the values displayed by the installer.
6. **Choose Install Folder** screen: Accept the default, which is `C:\Program Files\Good Technology\Good Control`, or specify the desired folder path.
7. **Choose Logs Folder** screen: make sure that the directory you specify is writable only by the service account installing the software. For security, the logs folder must not writable by anyone else.

   Note: If you are installing multiple GC or GP servers, such as in a cluster, each server must have a unique folder for logging. The servers must not share the same folder.
8. **Proxy Information** screen: the installer attempts to detect whether or not the host machine uses a proxy server to access the Internet. Follow the installer prompts as appropriate for your environment, and enter any additional requested information.
9. **Administrator Information** screen: enter the user name, domain, and password of the administrator account which will be used to run the GC server service.

   The administrator must be a valid user in Active Directory. This user must be in the administrator group for the machine that will host the GC server and must have the "Log on as a service" privilege. Also, the user must already be a GC administrator; if this condition is not met, you can log into the console of any GC in the cluster and add the user to the list of GC administrators on the **Roles** > **Administrators** screen.
10. **Database Information** screen; select the database type and enter the connection information.

    You must enter the database connection information used by other GC servers in the cluster. Choose Oracle or SQL Server for the database type. Select the Advanced connection type to provide the full connection string or select the Basic connection type and provide the host, service and port for your database connection.

    Confirm when the installer asks if you want to install the new GC in the existing server cluster.
11. **Registration Information** screen; provide the license information for the new GC server.

If you have not already obtained a license for your new GC server, log into the console of a GC server that already resides in the cluster and go to the **Server Configuration** > **Licenses** screen, then click Generate License. GC creates a new license and displays it on the console.

In the installer, enter your company name, if requested, and provide the license key that was generated by the GC console.

Note: Each license you generate through the GC console allows you to install one additional GC server in the cluster. Do not attempt to reuse a license that has already been consumed.

12. **SMTP Information** screen; enter the SMTP server information.

    In most cases, the SMTP server is your Microsoft Exchange server. GC uses the SMTP server to send welcome emails and provision emails to users. Users activate GD applications on their devices using the access keys contained in provision emails, so it is important to make sure the SMTP connection information is correct.

13. **Pre-Installation Summary** screen; review the information and if everything is correct, click **Install**.
14. **Certificate Information prompt**; review the generated certificate.
15. **Install Complete** screen; click **Done** to quit the installer.

    The installation logs are located in the C:\good\ialogs directory.

14. You can access the GC interface by pointing your browser to https://localhost/ or https://*server_name* on the GC server host machine.

    The GC console login screen appears, and you can log in using the account credentials you supplied to the installer in step 8.

    Also, browse to https://*server_name* from another machine on your enterprise network to verify that you can access the GC server. Make sure your firewall settings allow incoming connections from domain networks.

    If the GC console does not load in your browser, check the status of the Good Control Windows service, and start the service if it is not already running. The GC server logs are located in the C:\good\gclogs directory.

Installing Good Proxy Server

You must have the following in order to install the GP server:

- The GP server installer (**gpsetup.exe**), which is available through the GDN portal
- The GC server and GP server fully qualified domain names. Not just the bare `hostname`, but the full `hostname.somedomain.com`.

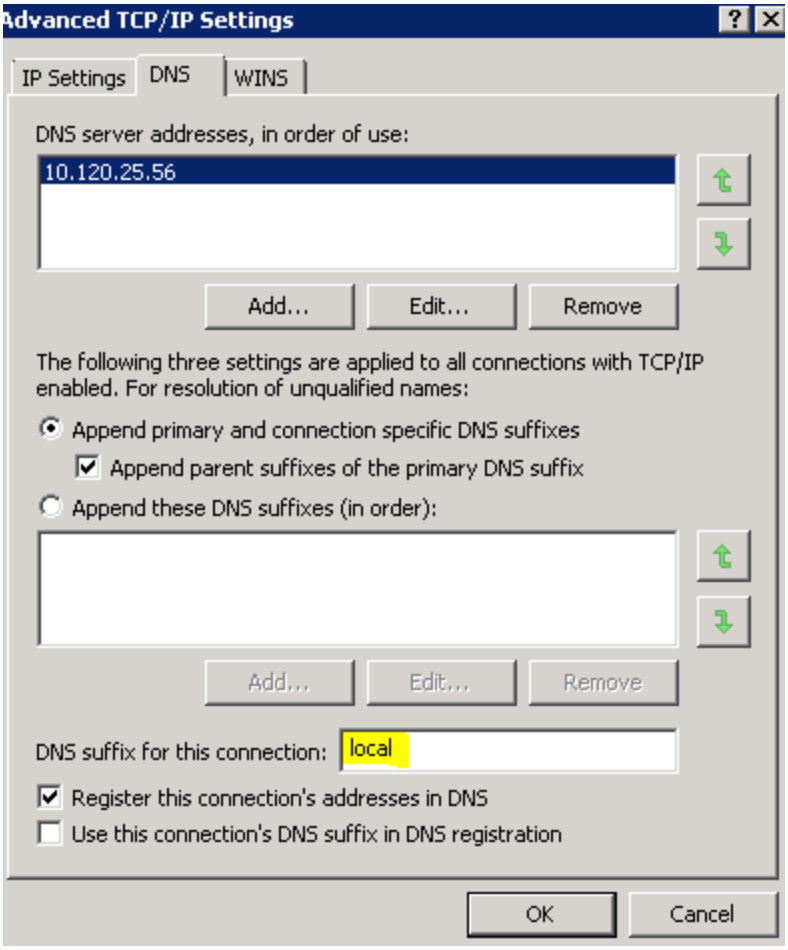
Network Setup for GP as "Workgroup Only"

The default installation option is that the GP and the GC are in the same domain. Installing GP as "workgroup only" (that is, in a domain that is distinct from the domain of the GC server) is a deployment option for GP detailed in the step for **Registration Information**, below.

The installation for GP as workgroup only is essentially the same as the default, with some slight differences in settings.

If you are deploy GP as workgroup only, before beginning the actual GP installation, make sure of the following on MS Windows:

1. Grant “Log on as service” right to a local user account on the Good Proxy server's Local Security Policy.
2. In MS Windows Advanced TCP/IP settings, enter `local` for the DNS suffix.



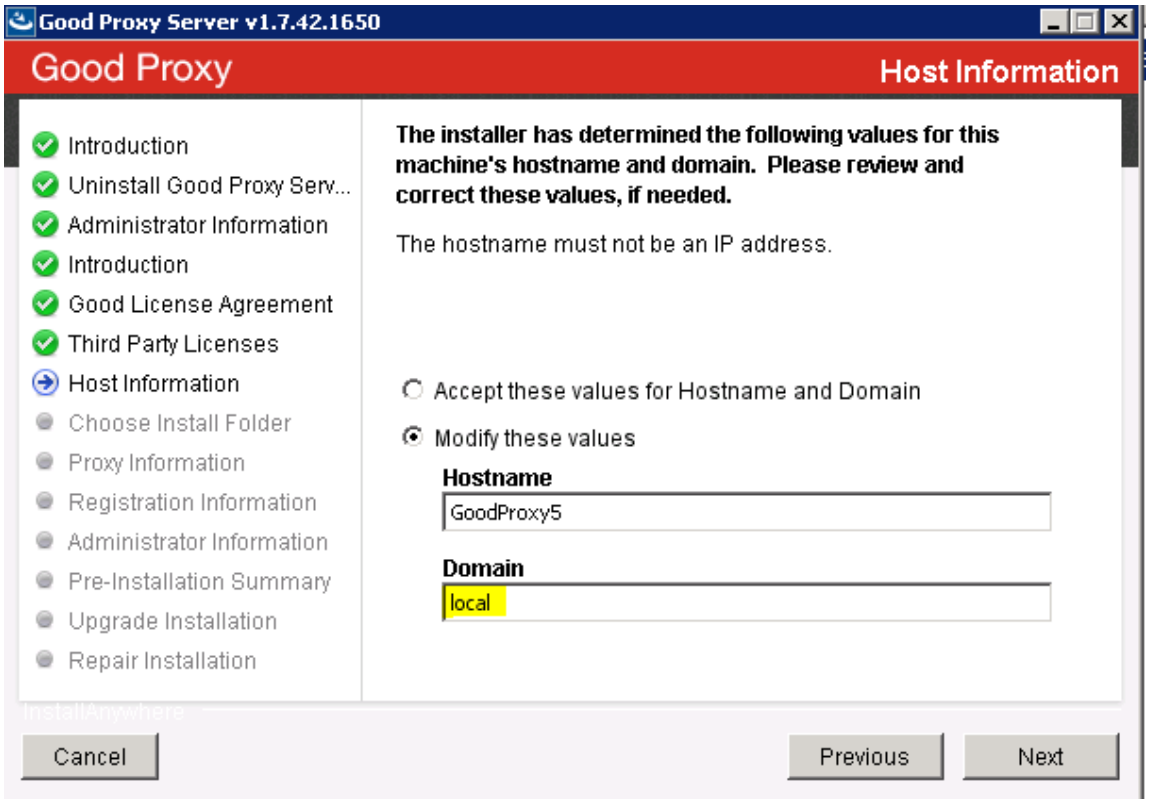
Now follow the actual installation steps and see the rest of the configuration for GP as workgroup only under the **Registration Information** step below.

Installation Steps

1. Run **gpsetup.exe** to begin the installation.
2. **Introduction** screen
3. Accept the **Good License Agreement**.
4. Accept the**Third Party License Information**.
5. **Host and domain information**: verify that the the hostname and fully qualified Internet domain name displayed in the panel is correct for the target machine, like mygc.mydomain.com (for example). The hostname must not be an IP address. Either accept the supplied values or modify the values provided by the installer.
6. **Choose destination folder**: you can change the drive designator but do not change the installation location. Accept the default.
7. **Web proxy information:** If you are using a web proxy server to access the Internet, click **Use a Web Proxy** and then supply the necessary information appropriate for your environment. If your web proxy requires authentication, supply the necessary credentials.
8. For the **Registration Information**, unless you are installing "GP as work group only", accept the default value or enter the fully qualified domain name of the GC server, like gc.mydomain.com. Do not enter an IP address.

   The default configuration assumed by the Good Proxy installation program is to have the Good Proxy and the Good Control servers in the same domain, but you can install them in different domains, if needed. This is sometimes referred to as "GP as workgroup only." This configuration is particularly useful with Direct Connect, where joining directory service domains can be complicated.
   The additional steps for GP as workgroup only at this point are detailed below.

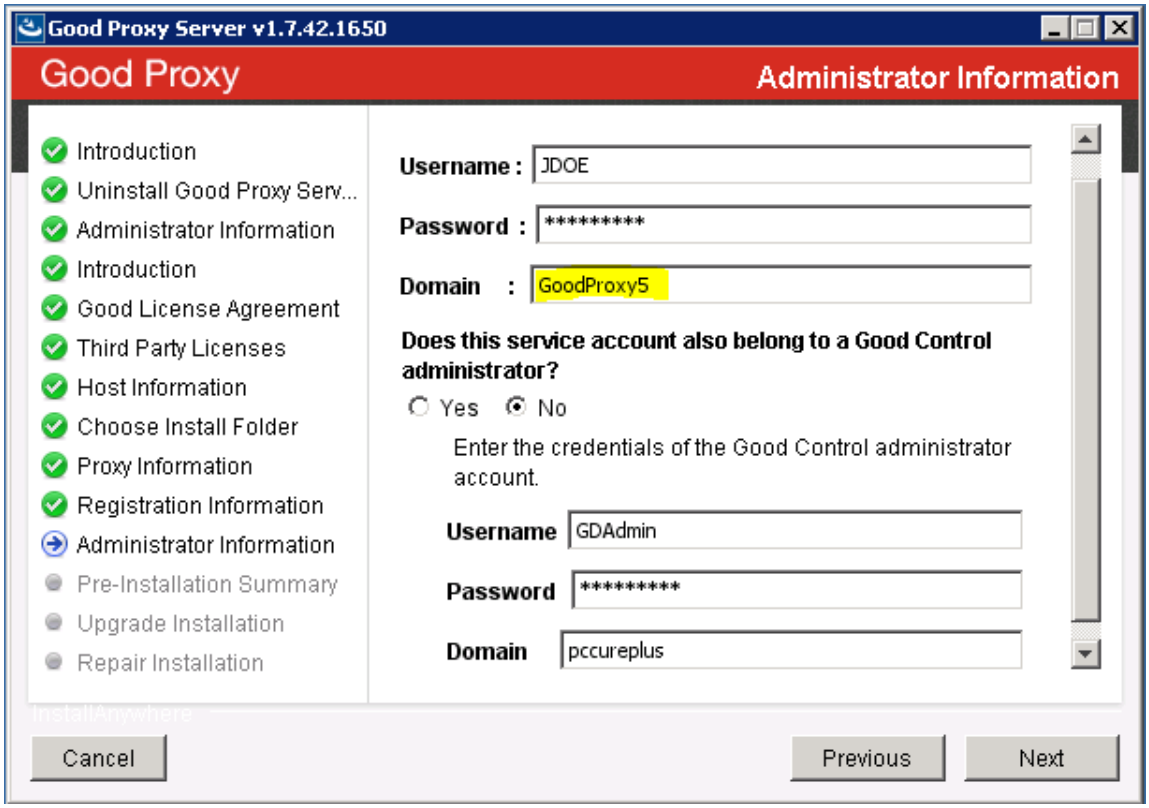
a. For the **Domain** field enter `local`, and continue.



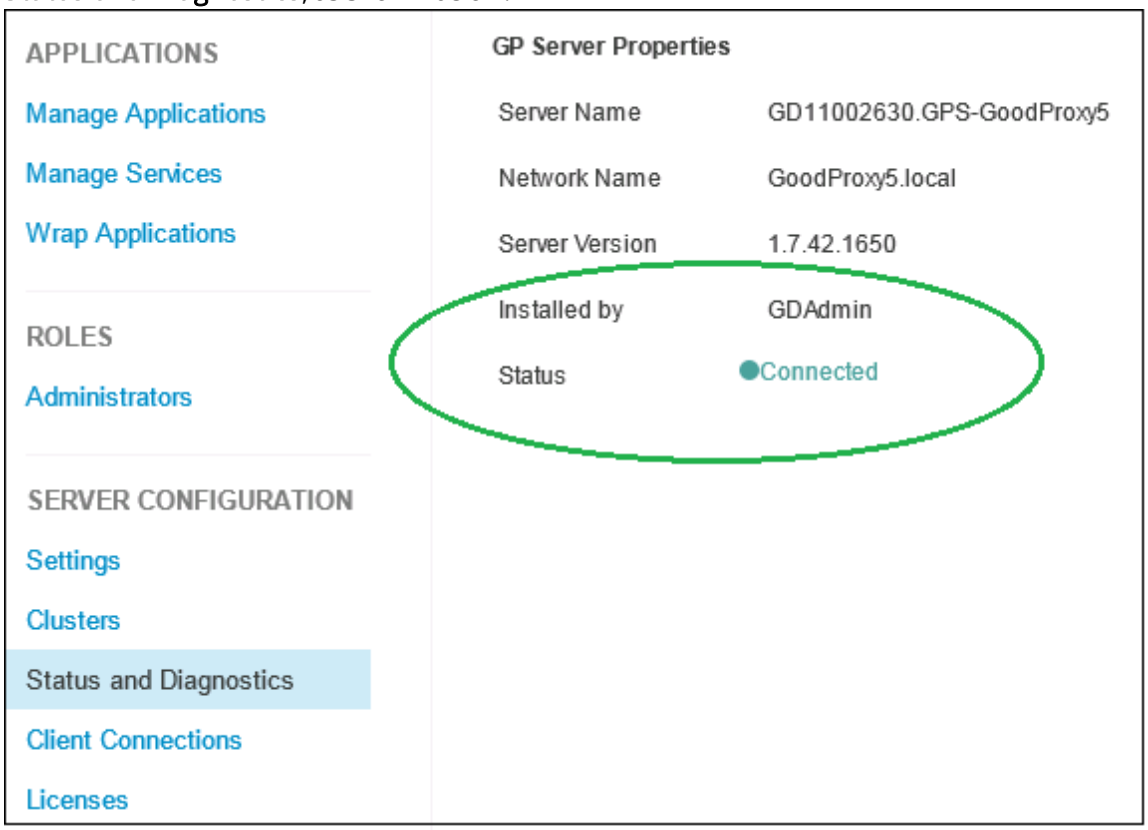
b. Supply the details for the fields in the top section of **Administrator Information** (the next screen of the installer):

Field or Prompt	Details
Enter the Credentials for the Good Proxy service account	This is the `local` user on the Good Proxy server.
Good Proxy domain field	Must be the host name of the Good Proxy
Does this service account also belong to a Good Control Administrator?	Select **No**.
Enter the credentials of the Good Control Administrator	Supply the correct values.

Below is an example of the fields and values:


Good Proxy Server v1.7.42.1650
Good Proxy
Administrator Information
Introduction
Uninstall Good Proxy Serv...
Administrator Information
Introduction
Good License Agreement
Third Party Licenses
Host Information
Choose Install Folder
Proxy Information
Registration Information
Administrator Information
Pre-Installation Summary
Upgrade Installation
Repair Installation
Username : JDOE
Password : *********
Domain : GoodProxy5
Does this service account also belong to a Good Control administrator?
Yes
No
Enter the credentials of the Good Control administrator account.
Username GDAdmin
Password *********
Domain pccureplus
InstallAnywhere
Cancel
Previous
Next

c. Complete the installation and confirm the Good Proxy shows connected in the Good Control **Console** > **Status and Diagnostics**, as shown below:



9. Supply the following information on the **Administrator Information** screen.

   Note: If you are installing GP as workgroup only, see the details in the previous step.

   - Enter the username, password, and domain of the account to be used by the GP service.
   - Depending on whether this service account also belongs to a Good Control administrator, check **Yes** or **No**. If **Yes**, enter the username, domain, and password of the account needed by GP to connect to that Good Control service.

     Note: This user must be in the administrator group for the machine that hosts the GP service and must have the "Log on as a service" privilege. Also, the user must be a GC administrator. This account is used to run the GP server as a Windows service.

10. When asked to trust the certificate retrieved from the GC server, click **Accept** or **Reject**.
11. **Pre-Installation Summary**: verify the information and if everything is correct, click **Install**.
12. **Install Complete** screen: click **Done** to exit the installer.
13. The installation logs are located in the C:\good\ialogs directory, and the GP server logs are located in the C:\good\gpslogs directory.

Additional Windows Configuration

Starting with Windows Vista, Microsoft introduced the Receive Window Auto-Tuning feature which continually monitors network conditions and scales the TCP Receiving Window for optimal performance. Unfortunately, older routers and firewalls may not handle the window scaling correctly and may cause data loss or dropped connections.

We highly recommend that you perform the following steps to avoid this issue if your GP server is installed on a machine running a more recent Windows OS than Vista.

1. Launch a command prompt as an administrator.
2. You must open a 64-bit command prompt if you are running a 64-bit version of Windows. For 32-bit and 64-bit Windows installations, navigate to your Windows System32 directory (usually in C:/Windows/System32), right click the cmd.exe file, and select the **Run as administrator** option.
3. Run the following command:
   `netsh interface tcp set global autotuninglevel=disabled`
4. Restart the machine.

Increasing the GC or GP Server's Java Heap Size

You might find the need to change the size of the memory heap of the Java Runtime Engine (JRE). These are the steps.

The heap size must be set depending on your hardware configuration, such as amount of physical memory or number of CPUs. Overallocating memory can cause other performance issues.

Good Technology recommends that you set the size of the GP server's Java heap to 60% of physical memory, with an upper limit not to exceed 5 GB. If you think it necessary to allocate more than this, get guidance from Good Technical Support.

To increase the GC or GP server's Java heap size:

You should be familiar and comfortable with using the Windows Registry Editor (`regedit` command). The HKEY entries in the registry for GC and GP are as follows:

GC	HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Apache Software Foundation\Procrun 2.0\GoodControl\Parameters\Java
GP	HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Apache Software Foundation\Procrun 2.0\GPS\Parameters\Java

1. Stop the GC or GP server. See Stopping the GC and GP Servers.
2. Login to the GC or GP server as administrator.
3. Start the Windows Registry Editor (`regedit` command).
4. Find the desired entry in the registry tree for either GC or GP.

5. Under the `Java` node, set the values of following keys to the desired settings. These keys correspond to the similarly named options on the JRE command:

   Important: The initial values for both keys should be the same.

   - `JvmMs`: Amount of memory in megabytes to allocate to the Java Virtual Machine at start
   - `JvmMx`: Maximum amount of memory in megabytes to allocate to the Java Virtual Machine
6. Save your changes to the registry.
7. Start the GC or GP server. See Starting and Restarting the GC and GP Servers.

Good DM and AM Deployment Models

Good device management and application management are features of the Good Control server. As such, it is deployed in the same way as Good Control. There are two deployment models, in both of which the servers and other aspects of the management features are hosted by Good Technology:

- **Good Control Cloud**. In this model, an administrator can provision all major components of the solution in Good Technology's cloud, including a Good Control server.
- **On-premise**. In this model, the Good Control and Good Proxy servers are deployed on the customer's premises. Good Control and Good Proxy servers need connectivity to the Good NOC.

Good DM and AM Installation Procedure

DM and AM are deployed along with Good Control itself. No additional installation or configuration is necessary.

After Installation: Additional Configuration Tasks

For a list of related documentation that describes the various features or configurations you might be interested in after installation, see Good Dynamics Documentation and the subsection about **GD Servers**.

In particular for administrative tasks, see the Good Control online help (either in the GC itself or in its PDF form) for information about installing certificates, configuring server properties and more, not limited to the following:

- Configuring Web Proxies in GC and GP
- Changing SSL certificates on the GC or GP
- Starting and Stopping the Servers
- Customizing Email Templates
- Installing Additional Servers
- Removing Servers from Clusters

Uninstalling the GC or GP Server

You can use the installation program for GC or GP to uninstall the server software.

Note: After installation, although the software itself is removed, the GC or GP server software is still listed in the Installed Programs.

High-Level Steps for Upgrading GC or GP

These are the high-level steps for upgrading the GC and GP.

The exact steps for upgrade are version-dependent. The release notes for GC and GP always state what earlier versions can be upgraded.

Details about backing up and stopping/starting servers are in the GC console online help.

Note: The back-ups you make in this procedure are only in case of a problem. You do not need to restore the database as part of the upgrade.

1. Make a back-up of the common shared database.
2. On each GC node, ensure to back up its respective `C:\good` directory and the *`Good_Control_install_dir`*`\jre\lib\security\cacerts` file.
3. Do not delete the current working license on GDN.
4. Shut down the Good Control service on each of the GC nodes.
5. Upgrade one GC and bring it back up to ensure it has migrated and you can log into the console. Then upgrade the remaining nodes.

   Note: if you are upgrading from GC version 1.6.x or earlier, depending on the number of user account records in the database, the upgrade of the first node can take a long time, because of a database schema change.
6. Optional: On each GP node, back up its respective `C:\good` directory and the *`Good_Proxy_install_dir`*`\jre\lib\security\cacerts` file. Because the GP is stateless, it's just as easy to uninstall and reinstall (rather than restore) the GP if something goes wrong. The GC is not stateless; you must back up that server.
7. Upgrade each GP.

Variations on Deployment Configurations

Good Dynamics and its components are highly configurable to adapt to a wide variety of networking and security needs. This configurability can make planning a deployment appear more complex than it is. Here we present conceptual views of the basic deployment configurations to help you see more clearly the options available to you.

In many ways, the different deployment configurations are cumulative and complementary. There are many more possible configurations than are shown here. Here are six of the more common:

1. Basic Development Configuration
2. Basic Distributed Servers
3. Distributed Servers + Web Proxy
4. Distributed Servers + Web Proxy + Direct Connect
5. Distributed Servers with Kerberos Constrained Delegation
6. Clustering, Affinities, and Database Mirroring

Important: The diagrams here are highly simplified, even oversimplified. For instance, the Good Network Operations Center (NOC) component is not pictured. Be sure to consult more detailed diagrams, network topologies, and other material from your Good technical representatives and in other GD server-related separate guides and later sections in this document, including the following:

- Server Clustering and Affinities
- Kerberos Constrained Delegation for Good Dynamics
- Direct Connect Feature Summary
- GD Sizing

1. Basic Development Configuration

Probably the most common development environment or a proof-of-concept configuration has all the GD components on a single machine.

This configuration gives a developer of GD-based applications the independent control to develop, deploy, and test an application without necessarily involving an IT administrator.

2. Basic Distributed Servers

The simplest configuration of distributed servers is a dedicated machine (physical or virtual) for each GD component.

This is the basic building block for all production deployments.

3. Distributed Servers + Web Proxy

Another common deployment configuration of distributed servers includes a web proxy server for access from mobile devices to resources on the Internet that are external to the GD deployment.

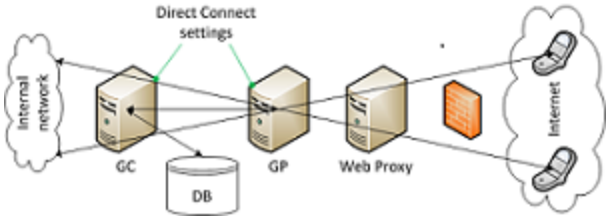
Details about GC properties (settings) for web proxies are in Good Control Administrator Help.

4. Distributed Servers + Web Proxy + Direct Connect

Users of mobile devices in the field often need to access internal resources directly via the GP server, bypassing the Good Network Operations Center (NOC, not shown). This configuration is called *Direct Connect*.

Reasons for enabling Direct Connect include increased performance (by decreased network latency) and location sensitivity (such as avoiding sending traffic through U.S. or other data centers).

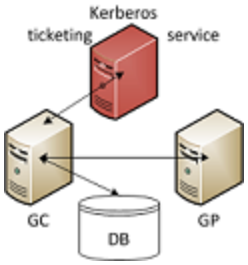
Detailed information about deploying Direct Connect is in the GD Direct Connect guide.



5. Distributed Servers with Kerberos Constrained Delegation

An alternative to user authentication involves integration with the Kerberos security system. This configuration is called *Kerberos Constrained Delegation*.

Detailed information about deploying GD with Kerberos is in Kerberos Constrained Delegation.



6. Clustering, Affinities, and Database Mirroring

Not strictly a distinct deployment topology, the GC/GP servers can be clustered for performance, server affinities (associations between servers) can be established, and the database can be mirrored. Clustering, affinities, and database mirroring are often done after the initial deployment of a chosen configuration, usually based on performance testing or capacity planning.

- Many details about how to create clusters and affinities are in GD Server Clustering and Affinities.
- Configuring GC for database mirroring is discussed in Optional: JDBC Connection String for SQL Mirroring

Good Dynamics™

GD Direct Connect

Direct Connect

Last updated: May 26, 2015

Versions: GP 2.0.*xx.yy*, 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

© Copyright 2015. All rights reserved. All use is subject to license terms posted at www.good.com/legal. GOOD, GOOD TECHNOLOGY, the GOOD logo, GOOD FOR ENTERPRISE, GOOD FOR GOVERNMENT, GOOD FOR YOU, GOOD APPCENTRAL, GOOD DYNAMICS, SECURED BY GOOD, GOOD MOBILE MANAGER, GOOD CONNECT, GOOD SHARE, GOOD TRUST, GOOD VAULT, and GOOD DYNAMICS APPKINETICS are trademarks of Good Technology Corporation and its related entities. All third-party technology products are protected by issued and pending U.S. and foreign patents.

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

Table of Contents

Revision History

Direct Connect

Date	Description
2015-07-15	Version numbers updated for latest release; no content changes.
2015-05-26	Removed a misleading FAQ from Frequently Asked Questionsthat implied that you can control the communication paths of a specific application. Rather, in the GC you can control the priority of GP clusters for an application server, but neither GC nor Direct Connect has the ability at the application level to control the exact path through the network.
2015-04-30	Version numbers updated for latest release; no content changes.
2015-03-23	Version numbers updated for latest release; no content changes.
2015-01-15	Version number updated for latest release; no content changes.
2014-12-16	Styling and page layout updated
2014-11-03	Removed misleading statement about additional filtering possible on the forward proxy.
2014-10-27	Added details about SSL-certificate-based client authentication
2014-10-21	Added note about ability to install the GC and GP servers in separate domains.
2014-10-07	Added complete list of SSL ciphers supported in GP for Direct Connect
2014-09-30 and 2014-10-03	Added clarification about the function of the GD Network Operations Center (NOC) in the Direct Connect configuration. See About the GD NOC and Direct Connect.
2014-09-25	• Updated for latest versions of GC and GP • Version stamp on cover page
2014-08-27	Fixed bad cross-reference to appendix.
2014-08-21	Added standard list of all Good Dynamics documentation
2014-08-13	Corrected graphic for SSL bridging requirements: proxy host field must be null.
2014-07	Completely revised for clarity

GD Direct Connect

GD Direct Connect is a deployment option for the Good Dynamics Secure Mobility Platform. It delivers direct control over application data path, reduces round trip time (RTT), and enhances performance—all resulting in a superior user experience.

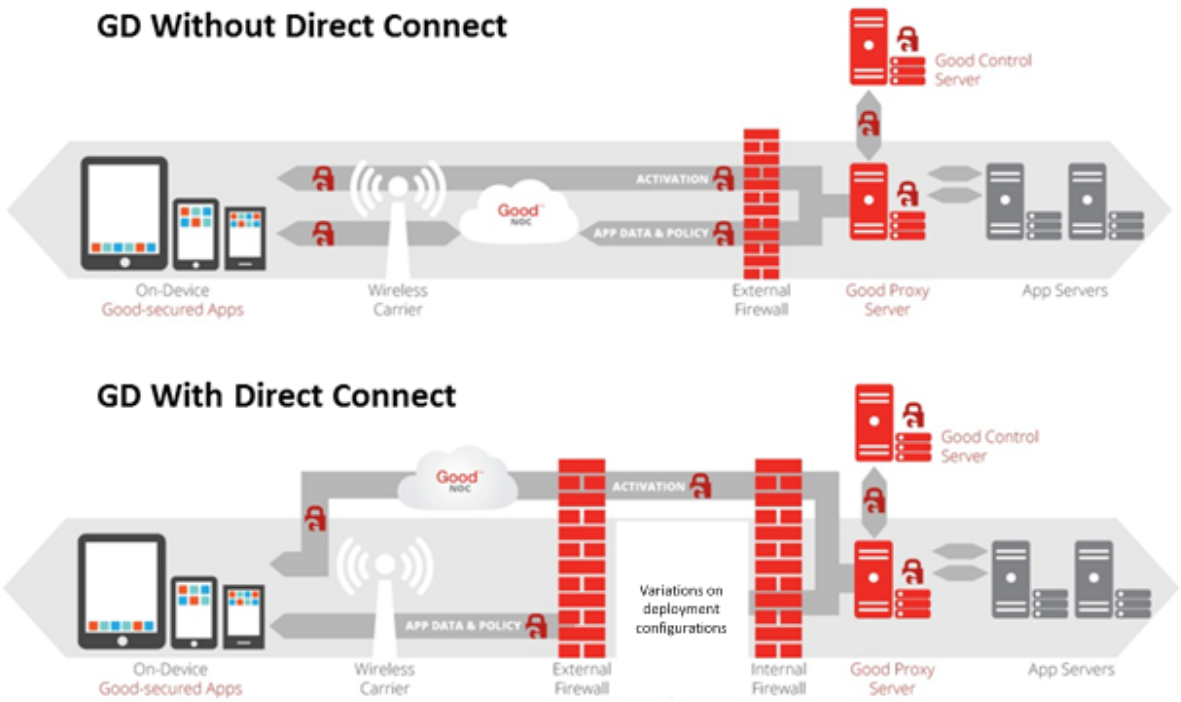
GD Direct Connect has several benefits:

- Enhanced control because application data is always under corporate control, flowing directly to/from the corporate network, an important feature when your enterprise needs its sensitive data restricted to national and/or corporate boundaries.
- Improved network performance because GD Direct Connect is a low-latency configuration allowing Good-secured applications to communicate directly with the Good Proxy server, thereby reducing data round trips to optimize bandwidth utilization for applications like HTTP video streaming.
- Better user experience because the reduced RTT lets applications refresh faster, contributing to a better overall user experience.

Before the introduction of GD Direct Connect, the physical distance of users and organizations in the Eastern US, Europe and Asia from the Good Dynamics NOC servers located on the USA's West Coast potentially meant longer network RTT because of latency in connection establishment.

Direct Connect avoids this latency issue by allowing your enterprise GD clients to establish direct connections with GP servers located behind the internal firewall, bypassing the GD NOC servers to eliminate four long hops—from GD client to GD NOC, from GD NOC to GP, then two hops back to the GD client from the GP, thereby reducing RTT.

Below are high-level views of the GD architecture, with and without Direct Connect. Direct Connect has three basic deployment models, which are detailed in Deployment Configurations.



Depending on your organization's proximity to the GD NOC, and assuming your GD clients are situated closer to your GP servers than to the GD NOC, the Direct Connect feature will likely improve the performance while reducing the latency of your GD platform.

GD Direct Connect does not eliminate the need for the GD NOC, which is still required for application activation and authorization on client devices. Once provisioned and activated, Direct Connect affords you the flexibility to route application directly from your enterprise network to/from the application containers on the device, instead of having to go through the GD NOC.

Direct Connect does not require any new Good components, although you can optionally use a standard commercial off the shelf HTTP proxy server to enable Direct Connect, rather than connecting to a GP server if so desired.

Direct Connect is not designed to provide better performance than a VPN. You should also not expect to see improvements when the GD client is in close proximity to the GD NOC.

About the GD NOC and Direct Connect

Even with the Direct Connect configuration, it is important to know that the GD Network Operation Center (NOC) is still a critical part of the architecture. It is always relied on for the following functions:

- Provisioning of applications on mobile devices.

- Notification of policy updates to active (currently open) GD containers. For inactive containers, the policy update takes place the next time the container opens, but realtime notification requires a connection to the NOC.
- Applications that rely on the Secure Push Channel require connectivity to the NOC.

Other reliance on the GD NOC with Direct Connect (or not) is pointed out in other sections of this document.

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.

Relationship to Cloud GC: Feature Not Applicable

The feature, service, server type, or software described in this guide is not available on Good Control Cloud because it is not applicable in a hosted environment.

Deployment Configurations

Regardless of which DC deployment option is used, the following statements are always true.

- GD NOC:
  - Provisioning of applications on mobile devices.
  - Notification of policy updates to active (currently open) GD containers. For inactive containers, the policy update takes place the next time the container opens, but realtime notification requires a connection to the NOC.
  - Applications that rely on the Secure Push Channel require connectivity to the NOC.
- SSL/TLS : Communication between a client and the Good Proxy server is always secured over SSL/TLS.
- Access: By default all clients are denied access to the Good Proxy server.
- Good Control: An administrator retains all device management capabilities in Good Control.

Direct Connect does not change the security of the system. It simply provides an alternate way to deliver data from the client to the Good Proxy server. In the following section we will take a closer look at the various ways that DC can be deployed.

There are three main ways to deploy DC.

- Port Forwarding
- Forward Proxy
- SSL Bridging and a variation SSL-Certificate-based Client Authentication

About the GP and GC in Separate Domains or "Workgroup Only"

The default configuration assumed by the Good Proxy installation program is to have the Good Proxy and the Good Control servers in the same domain, but you can install them in different domains, if needed. This is sometimes referred to as "GP as workgroup only." This configuration is particularly useful with Direct Connect, where joining directory service domains can be complicated.

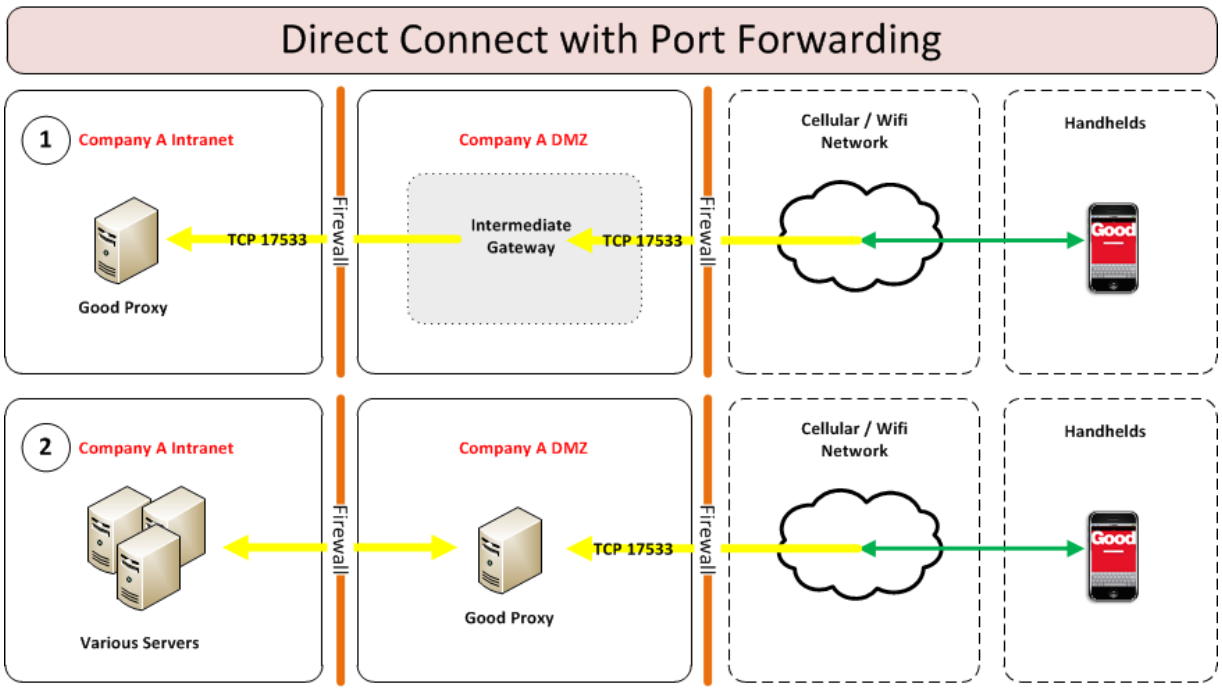
See the step for **Registration Information** in "Installing Good Proxy Server" in *GD Server Installation*.

Port Forwarding

This is the simplest deployment option for DC. In this approach we simply port forward all incoming client traffic to the Good Proxy server. There are two variations of this approach. The first variation is to port forward from the edge of the perimeter network directly into the corporate network where the Good Proxy resides. The Good Proxy server only requires one inbound port, TCP 17533. As long as the perimeter firewall is configured to only allow this port to the Good Proxy server then access is secured. As noted above, security policies are setup and managed in Good Control to allow access to the system.

The second variation of the port forwarding approach is to place the Good Proxy server in the corporate DM Z. The benefit of this approach is that you don't need to port forward directly from the edge of the perimeter network directly into the corporate network. Instead, you only need to port forward from the edge to the corporate DM Z network. However, additional ports will need to be open between the DM Z network and the corporate network in order to facilitate traffic between the Good Proxy and internal resources.

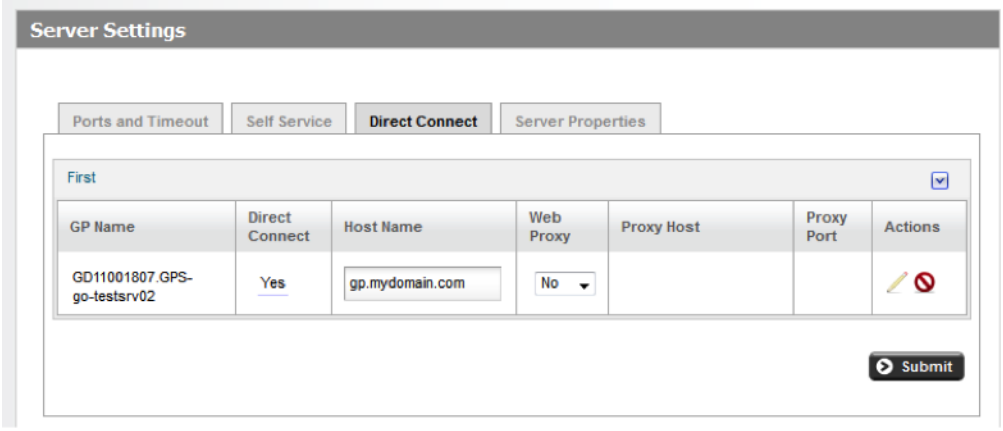
Both variations of the port forwarding approach are shown below.



Port forwarding requirements

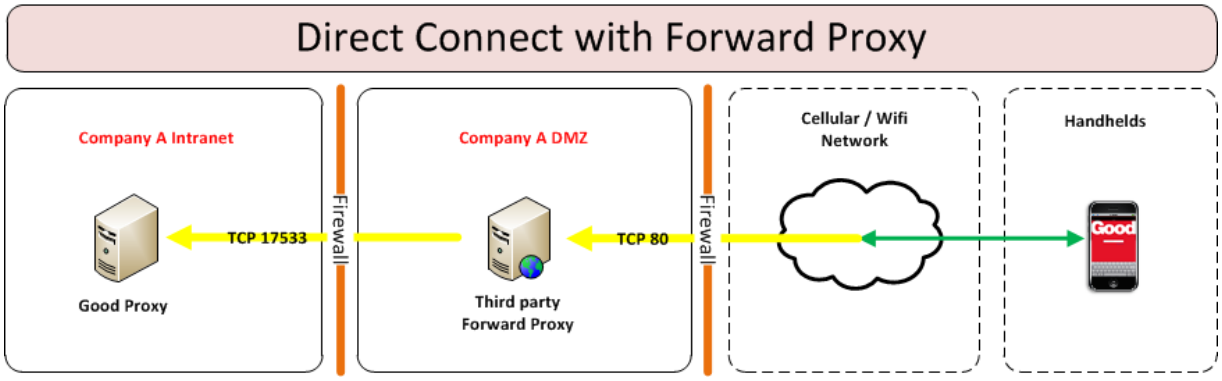
Regardless of which variation is used, a publicly routable DNS name is required for each Good Proxy server, for example, `gp.mydomain.com`. Depending on which method is used, the firewalls must be adjusted accordingly to forward TCP 17533. If you chose to place the Good Proxy server in the DMZ, then additional ports, including port 17433, need to be open between the DMZ and the corporate network. Other ports between the DMZ and corporate network vary depending on the resources that are required.

In Good Control, the Direct Connect configuration is accessible in the following menu on the left navigation area: **Server Configuration -> Settings -> Direct Connect tab**. The following is an example of the settings.



Forward Proxy

In this DC deployment option, a forward proxy web server is used to proxy client requests to the Good Proxy server. The following diagram illustrates how this is deployed. For simplicity, only the vital components are shown.



The major benefits of this approach are:

- No need to port forward directly from the edge network to the internal corporate network.
- The forward proxy can load-balance incoming client traffic across multiple Good Proxy servers.

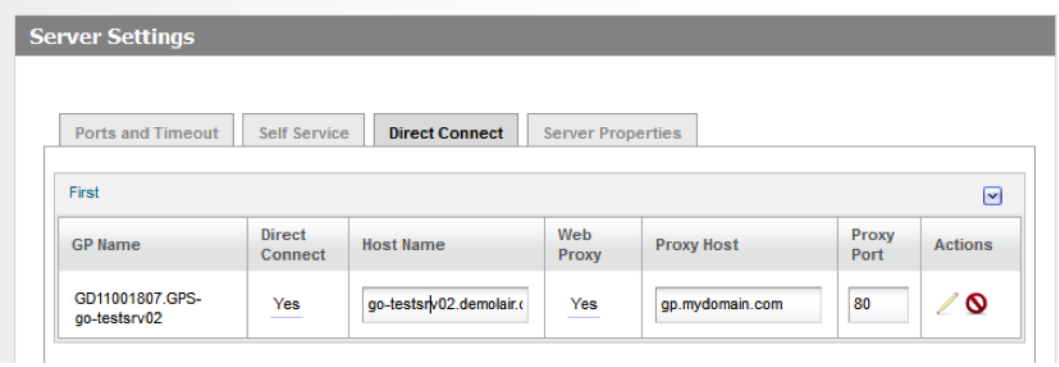
Forward proxy requirements

Direct Connect is agnostic with respect to forward proxying as long as the configuration meets the following requirements.

1. The forward proxy server must support the “HTTP CONNECT” method
2. The forward proxy must be able to communicate with the Good Proxy server via TCP port 17533
3. The forward proxy must be able to resolve the Good Proxy server's hostname.
4. An inbound port must be allowed to the Forward Proxy server. This port is arbitrary.
5. A publicly resolvable DNS hostname must be assigned to the Forward Proxy server.

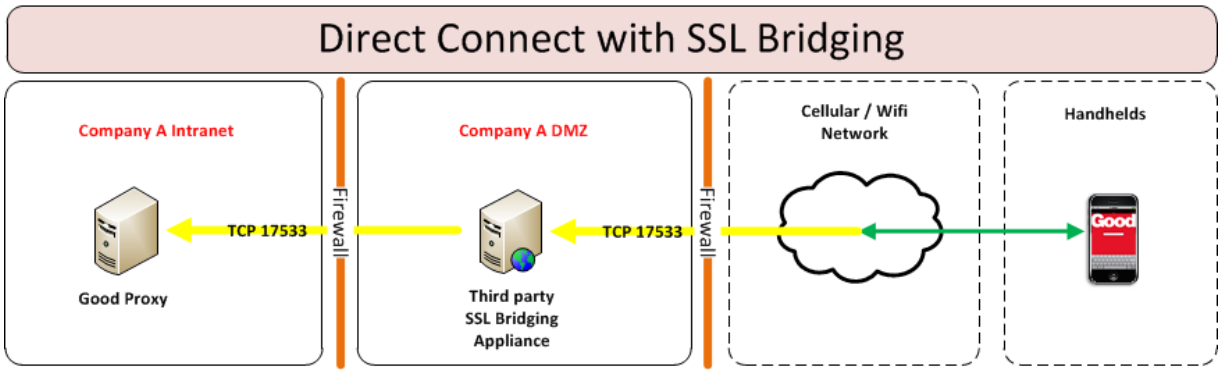
As long as the above requirements are met any Forward Proxy servers can be used for direct connect. Examples of how to configure an Apache Forward Proxy and F5 BIG-IP LTM Forward Proxy with direct connect is included in the appendix.

In Good Control, the Direct Connect configuration is accessible in the following menu on the left navigation area: **Server Configuration -> Settings -> Direct Connect tab**. The following is an example of the settings.



SSL Bridging

This deployment option is the most complex. This option involves using a third-party appliance to terminate the SSL/TLS connection from both the client and the Good Proxy server. The third-party appliance then bridges the two connections. The architecture is as follow:



The benefit of this approach is that the third-party appliance may be able to do additional filtering of the incoming traffic before sending it to the Good Proxy server. Load balancing of the incoming client traffic can also be achieved. However, these functions are highly dependent on the third-party appliance that is used. Configuration of these features is beyond the scope of Direct Connect. Consult your appliance manufacturer's documentation.

SSL Bridging Requirements

Direct connect is agnostic to the third-party SSL bridging appliance that is used as long as it meets the following requirements:

1. The bridging appliance must be able support the following ciphers

a. TLS_RSA_WITH_AES_256_CBC_SHA256 OR

b. TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

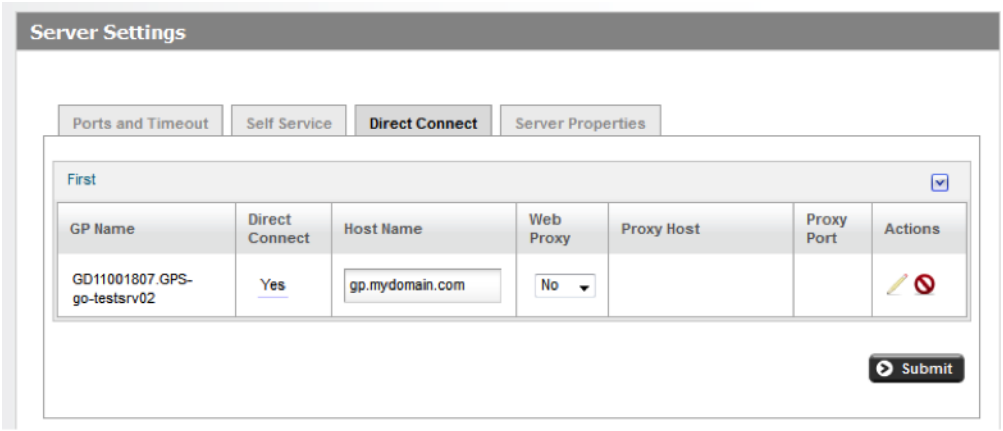
2. Inbound TCP 17533 must be opened to the appliance.

3. Outbound TCP 17533 must be open from the appliance to the Good Proxy server.

4. A publicly resolvable DNS hostname must be assigned to the appliance for the purpose of DC.

As long as the above requirements are met any third-party SSL bridging appliance can be used for DC. An example of how to configure a F5 BIG-IP LTM appliance for SSL bridging for DC is in Appendix: Direct Connect with SSL Termination at Reverse Proxy.

In Good Control, the Direct Connect configuration is accessible in the following menu on the left navigation area: **Server Configuration -> Settings -> Direct Connect tab**. The following is an example of the settings.



Note: This Good Control configuration is exactly the same as the Port Forwarding deployment model.

SSL-Certificate-based Client Authentication

This deployment option is a variation on SSL Bridging. In SSL bridging, an SSL termination device (or "appliance", such as Netscaler or F5) terminates the incoming direct connect connections and acts as a bridge between users' mobile devices and the GP. The SSL listener or connection point on the bridge contains the same public and private key that resides on the GP server the connections are destined for. In this configuration, connections from the mobile devices are terminated on the appliance, and a new secure channel is created from the appliance back to the GP. This allows true termination at the edge for incoming connections.

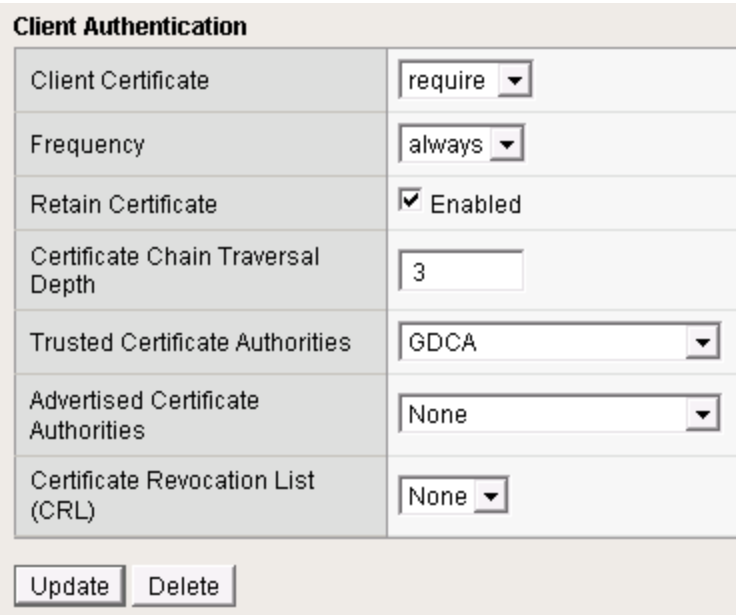
During the provisioning process of each Good Dynamics secured application, a certificate signing request (CSR) is generated by the application and signed by the GDCA (Good Dynamics Certificate Authority). After being signed, this Inter-Container Communication certificate (or "ICC cert") is sent to the application, where it is secured inside the secure container of the application. The GDCA that signs the ICC certificate is the root CA of a client's specific GD environment, and is the same root CA that signs the certificates on the GP servers and also the certificate exported to the listener of the SSL termination device where the SSL connection for Direct Connect terminates in this configuration.

Every application has an ICC certificate. This certificate is specific to the application, its specific GD environment used for key generation, and device on which the application is installed. The certificate is used for Good's patented Shared Services Framework (also know as AppKinetics) for inter-application transfer of files/data, such as "open-in" functionality. The SSL termination device's listener (endpoint) is responsible for issuing the challenge for presentation of a client certificate during the Direct Connect TLS 1.2 channel establishment from the application to the appliance. *Therefore, the appliance must have a copy of the GDCA certificate.* In the negotiation phase, once challenged, the application presents the ICC certificate to the appliance's listener, which then validates the ICC certificate against the GDCA certificate authority. The application also validates that the certificate presented by the appliance during the negotiation is signed by the GDCA, because the application has a copy of this root certificate in its secure container, as well. After both certificates are validated, the connection is considered authenticated and the TLS channel is successfully established. If the ICC cert is signed by any certificate authority other than the same GDCA that is configured on the appliance's listener, the authentication fails and no TLS channel can be established.

Setup Requirements

1. Make sure that your communications appliance supports SSL-certificate-based client authentication.
2. Set up Direct Connect with the SSL Bridging deployment configuration.
3. Configure the SSL listener on your appliance to require client certificate authentication. The device must also be configured to validate the presented client certificates against the GDCA root certificate, which must be exported from the GC and imported into your appliance. The exact steps on establishing this appliance requirement vary from one vendor to another. Consult your appliance vendor's documentation.

   Below is an example of the relevant settings on the F5, which come at the bottom of the Client-SSL Profile section. Note the name of the Certificate Authority: GDCA.



Testing GD Direct Connect

To test and verify your Direct Connect connectivity, we recommend using a custom application built with the latest GD SDK for iOS or for Android, or you can verify using one of the Good Dynamics sample applications included in the downloaded SDK bundle; for instance, the RSSFeed sample app.

Additional Considerations

The Good Proxy "external" address only needs to be reachable from the Internet if no HTTP proxy is used. If a HTTP proxy is configured, then only the HTTP proxy address needs to be Internet accessible. The GP "external" address, in this case, would only need to be accessible from the HTTP proxy.

Direct Connect is configured on an individual Good Proxy basis. This means you won't be able to configure Direct Connect at the cluster level. It is therefore recommended as a best practice to make sure all GPs in a cluster are configured for Direct Connect, since GPs in a cluster are chosen at random. Consequently, if some GPs in the cluster are DC while others are not, you will continue to have some connections going through the GD NOC arbitrarily.

Frequently Asked Questions

Included here are some of the most commonly asked questions regarding the GD Direct Connect feature.

Q. Are there any special requirements when na HTTP proxy is used for implementing GD Direct Connect?

A.A customer can use a standard off the shelf (OTS) HTTP proxy server as long as it supports the HTTP connect command and does not require separate authentication.

Q. Why would I choose to use the optional HTTP proxy?

A. You should choose the configuration that makes the most sense for your organization and environment. For instance, you may opt to use a HTTP proxy in the DM Z to reduce the maintenance cost of adjusting the internal firewall to allow connections between the Good Proxy and newly white-listed app servers. Even so, in cases where all connections to on-premise servers go through a proxy on campus, using Good Proxy may be the more suitable installation option.

Q. Can a reverse proxy be used when implementing Direct Connect?

A. Yes. See the details for configuring Direct Connect with a reverse proxy in this document.

Q. If there is no authentication at the HTTP proxy server level, is the Direct Connect as secure as the standard configuration, which relays data through the Good Dynamics NOC?

A. Generally when a HTTP proxy is put in the DM Z, authentication is required because the proxy is the access point to anything within the enterprise. This stricture is accommodated by configuring the DM Z-based HTTP proxy to only allow a path to the behind-the-firewall Good Proxy server using the specified port and address. Anything identified that is not explicitly configured on the DM Z-based HTTP proxy will not be allowed to go through the enterprise firewall, thereby restricting external access to the GP server, which performs the authentication, then allows the perimeter infrastructure to do its business and take care things like DPI, DOS detection/prevention, and so forth.

Q. Is Good Dynamics Direct Connect supported in HA/DR scenario?

A. Not only can GD Direct Connect be enabled in a HA/DR scenario, it is a recommended configuration so you can take advantage of your designated fail-over path. You can configure one DM Z-based HTTP proxy server for multiple Good Proxy instances or distinct DM Z-based HTTP proxy servers for each Good Proxy server.

Essentially, you could set up the primary cluster of Good Proxy servers to use the GD Direct Connect feature and point those Good Proxy servers to a single DM Z-based HTTP proxy server address. You can then designate a secondary cluster of GP servers to use another DM Z-based HTTP proxy server. Or, you can choose not to enable Direct Connect for that secondary cluster of GP servers.

Q. If the HTTP proxy in the DM Z fails and HA/DR has not been used, will clients fail over to the GD NOC? Can this failover be turned off for regulatory reasons??

A. With Direct Connect, connectivity to App Servers will adhere strictly to configurations set in the Good Control server. If you don't provide a non-Direct Connect path to an App Server, the client app will never connect through the NOC. However, for connectivity to Good Dynamics servers, if only Direct Connect paths are configured and the GD Library is unable to reach any Good Dynamics server via these Direct Connect paths, then it will fail over to connecting through the NOC. This is done to ensure that any policy updates, server configurations/addresses, and proxy configurations/addresses remain current.

Q. Can settings for the DMZ-based HTTP proxy be updated? If so, how quickly can these setting be received by a client app?

A. You can update the addressing information for the DM Z-based HTTP proxy at any time from the Good Control management console. Receipt of that new addressing information by the client app is immediate if the app connected to the network. If not connected, then the new addressing information is received immediately upon the next connection.

Q. Does the client app always need to connect through the Good Dynamics NOC when there is a connection failure to the DM Z-based HTTP proxy?

A. The complete GD Direct Connect addressing information is sent to the client at activation and again whenever this information is changed. If more than one HTTP proxy server is in use in an HA/DR scenario, the client does not need to reconnect to the Good Dynamics NOC after a connection failure in order to get the address of additional HTTP proxy servers, since it already has all of this information.

Q. If I implement Direct Connect, do I need to restart the Good Dynamics servers?

A. You do not need to restart the Good Dynamics servers if you change the Good Dynamics Direct Connect setting. Changes are transparent to the end user.

Q. Do the Good Dynamics servers monitor the health of the proxies used for GD Direct Connect?

A. The Good Dynamics servers do not monitor the health of any HTTP web proxies used for Direct Connect. You are therefore encouraged to configure multiple DM Z-based proxies, as well as to monitor proxy health using other off the shelf network monitoring tools.

Q. How is app data secured with Direct Connect?

A. Not only is traffic end-to-end encrypted but in the Direct Connect case, where SSL is used to secure the link between the client and the Direct Connect relay (or load balancer) the client only accepts certificates that come from a CA that is under the control of the customer and so is not subject to attacks through the coercion of a commercial CA.

Q. For SSL bridging or proxying, how can I change/add to the SSL ciphers that can be allowed?

A. By default, SSL communications between the GC and GP servers over port 443 for the Direct Connect configuration uses the following ciphers:

- TLS_RSA_WITH_AES_256_CBC_SHA256 OR
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

If you need to add more ciphers, after installation, edit the GP server's configuration file `c:\good\gps.properties` and add the names of the ciphers to the `gps.directconnect.supported.ciphers` key.

One reason you might need to add more ciphers is if you have your own proxy server between your client devices and the GP server configured for Direct Connect. This middle proxy is the one that determines which SSL ciphers to use. You need to ensure that the GP server ciphers correspond to those required by your own proxy.

Q: Some of our GD applications were compiled with older versions of the GD SDKs that do not support Direct Connect. Will they have issues connecting if we move to Direct Connect?

A: Yes. This will cause an issue and older apps will not be able to connect. One way to mitigate this problem is to configure Direct Connect at the application level, so some applications communicate via Direct Connect(those applications those that support Direct Connect), and other applications connect via GD NOC. However, to support this configuration, you will need multiple GP clusters.

Appendix: Direct Connect with SSL Termination at Reverse Proxy

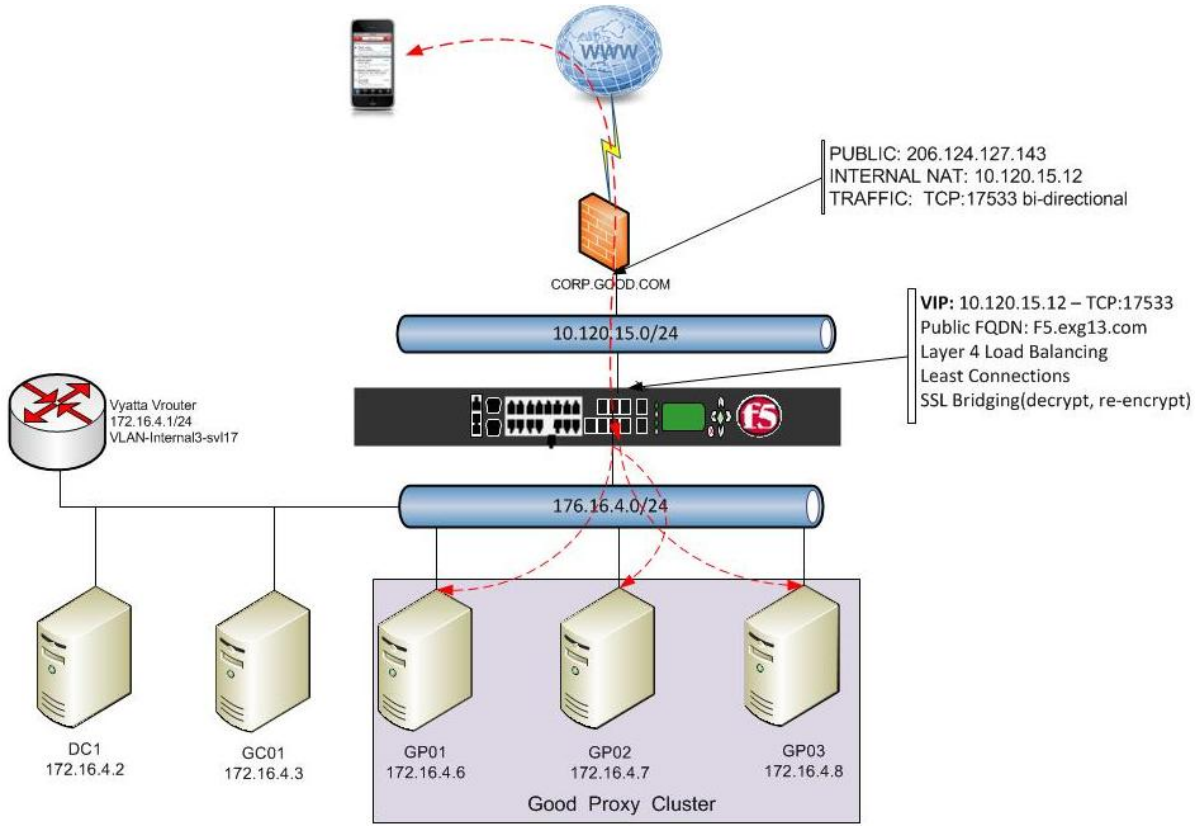
GD Direct Connect is currently supported in two main deployment models:

- Direct Connect with no Web Proxy
- Direct Connect with a Web Proxy

While both of these methods are supported, many enterprises prefer to use an edge network device that will terminate the SSL connection from the device as it ingresses into the corporate network at internet edge. Upon connection to the edge device, the application can establish connections to any of the Good Proxy servers defined in the cluster specified for Direct Connect.

The following diagram showsthe network architecture, subnets, location of reverse proxy(F5), and traffic flow as applicable to Direct Connect.

This architecture allows a single publically exposed IP address to accept connections for all servers within a GP cluster.



Creating the Key Pair for External Listener on F5

The Good Dynamics platform uses a proprietary method for signing, securing, and distribution of certificates used in the communication process between GC and GP servers, along with communications between GD secured applications and GP servers.

For this initial testing, it is required to use some open source SSL key tools to create the necessary key pair required for utilizing the F5 as a reverse proxy.

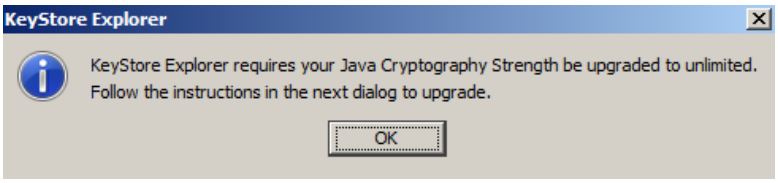
Installing the Key Store Explorer

Some recommendations before you begin:

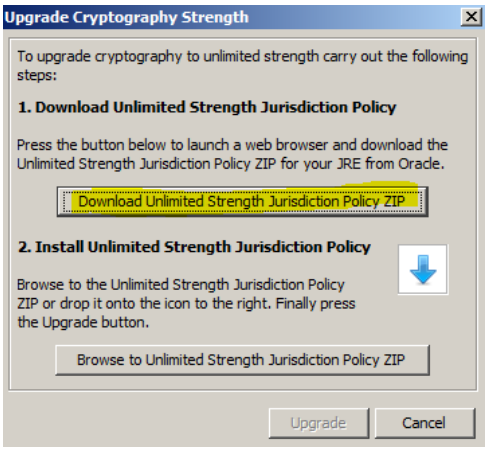
- Make a backup of the GC server's *installation_directory*\jre\lib\security\cacerts file.
- Good Technology recommends that for testing you install the key store explorer on a server external to your intranet and not on the GC server. GC has its own copy of the Java Runtime Engine (JRE) that is probably different from that required by the key store explorer.
- From this separate server, make sure you have read/write access to the GC server's file system.
- If you must install the key store explorer on the GC, install Java for it in a directory that is separate from the GC Java directory. If the values of the GC's JAVA_HOME and JRE_HOME environment variables have to be modified, after testing make sure to reset the variables to their original values.

Steps

1. Download the required version for your operating system onto your separate external-to-the-intranet server or the Good Control server:
   http://keystore-explorer.sourceforge.net/downloads.php
2. After launching the application you will be required to install Java if it is not already installed. Clicking "OK" if Java is not found will direct you to the appropriate download site.
3. Upon re-launch of the Keystore-Explorer application you will be prompted to upgrade your Java Cryptography Strength to unlimited.

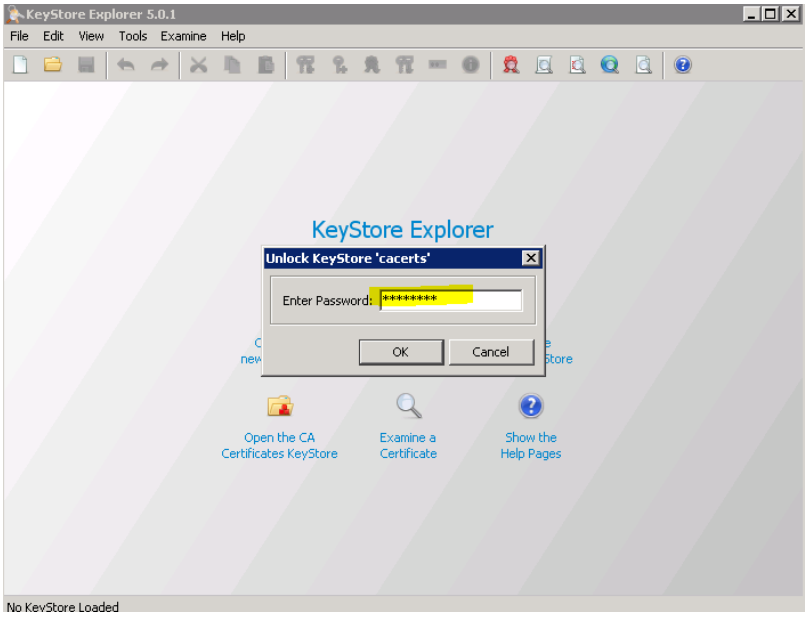


4. Pressing OK will direct you to appropriate site to download the appropriate zip file.

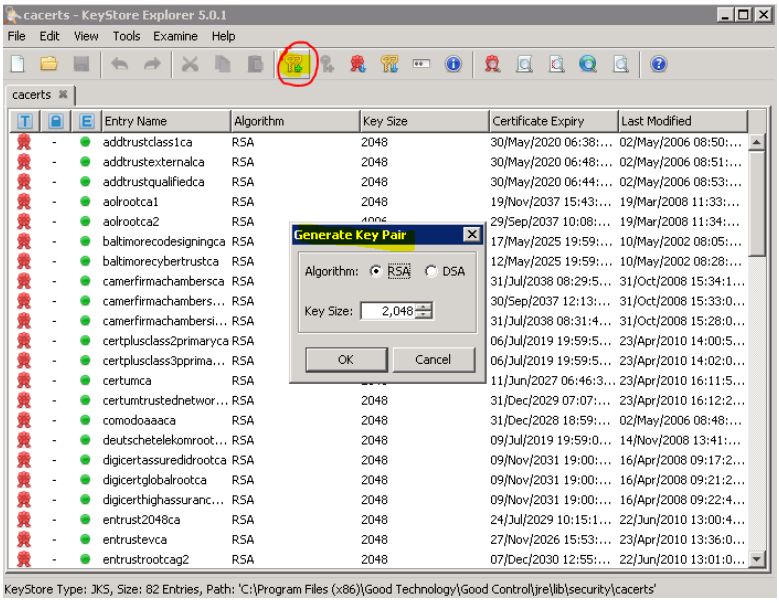


5. Save the zip file in a known location on your local PC and then browse to the saved zip file for Keystore-Explorer to import required files and click "Upgrade"
6. In the Keystore tool, select "Open an Existing Keystore" and browse to the following installation directory on your Good Control server: *installation_directory*\jre\lib\security.

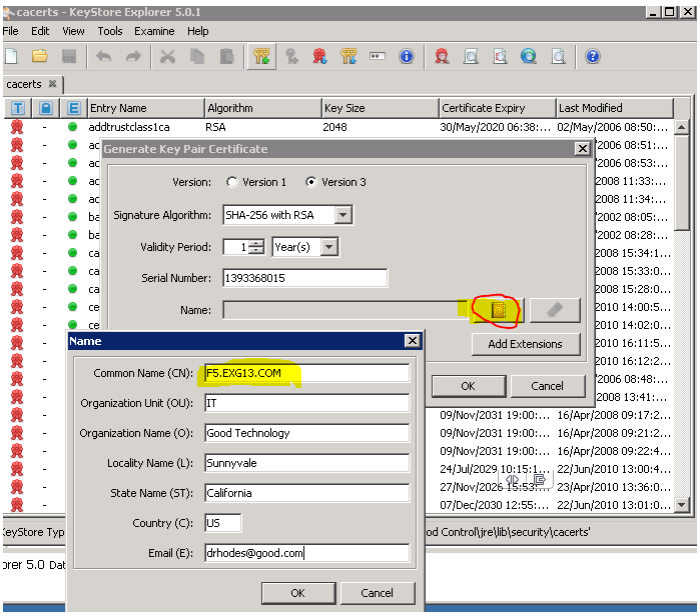
7. Select the file `cacerts` to open in the Keystore Explorer. When prompted for password, the default password is `changeit` all lowercase.



8. Generate a Key Pair by clicking on the icon as shown below and then selection OK, accepting the auto populated values.



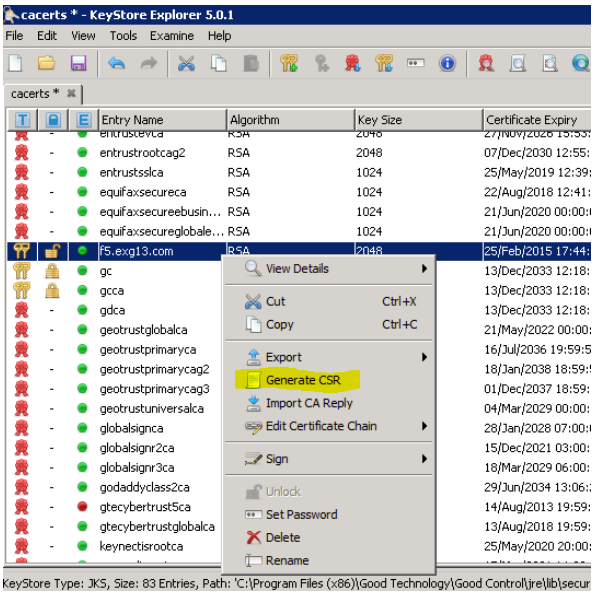
9. After the key pair is generated, you will be prompted with the screen below, Click on the highlighted section to change the certificate attributes to match your environment. The "Common Name (CN)" field must be populated with the FQDN of the F5 listener. This is the name resolvable from the public Internet for which this certificate will be valid. In this example F5.EXG13.COM is the name of the listener.



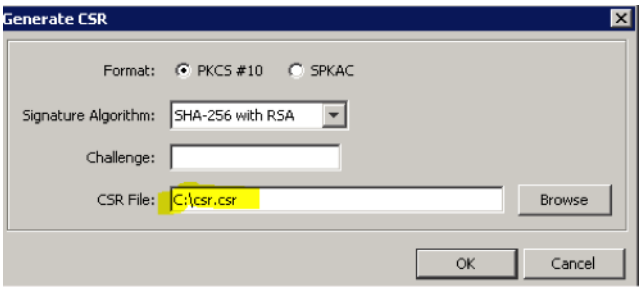
10. Click OK, then OK to accept the default alias which should be the CN you populated in above step, and input the password and confirm password.

    Important: You must use the password `changeit` because the web server associated with the GC expects this password.

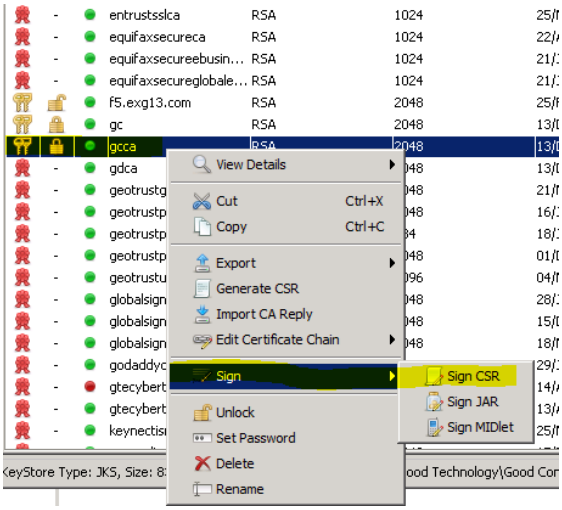
11. Next generate a CSR from this key pair by right-clicking and selecting **Generate CSR**.



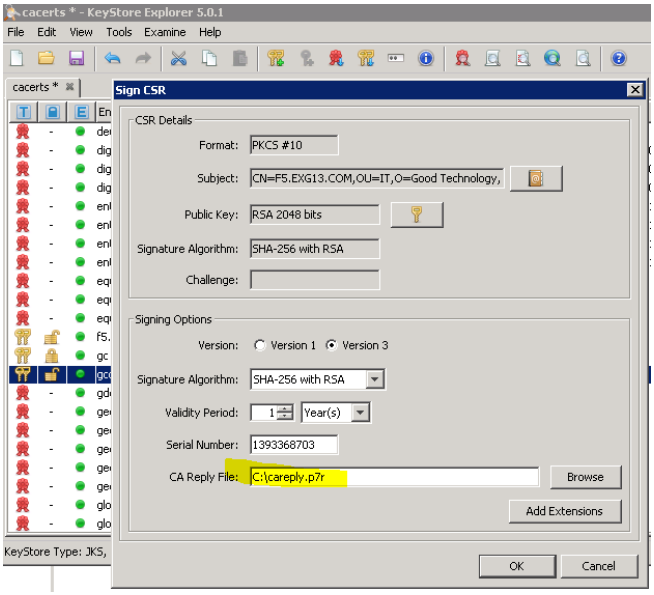
12. Leave the default Format and signature Algorithm and enter a location and name for the CSR.



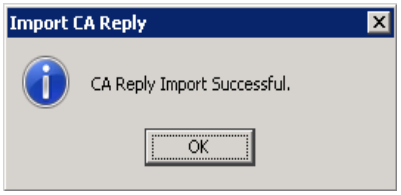
13. Right-click the "gcca" key and choose to sign the csr file you just created. You are prompted for the same password to unlock the gcca key to allow it to be used in the signing of the CSRfile.



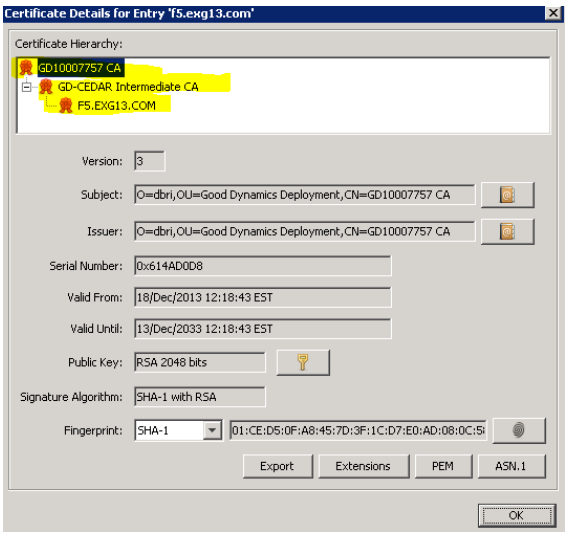
14. Browse to the CSRfile generated, select it, and then fill in the field shown below to have the tool output a CSR reply file. Choose the location and name in the field shown.



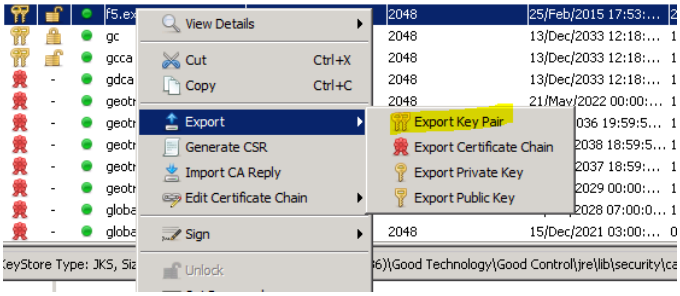
15. Next, right-click on the newly created keypair, and select "Import CA Reply" as shown below. Browse to the saved .p7r file from the previous step and select OK. The certificate is now complete with exportable public and private key.

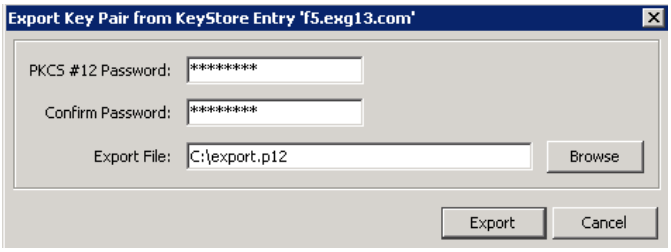


16. By right clicking on the final key-pair you and selecting certificate chain details, you should see details similar to the following screenshot, showing the Root CA, the intermediate CA, and the final certificate you just completed.



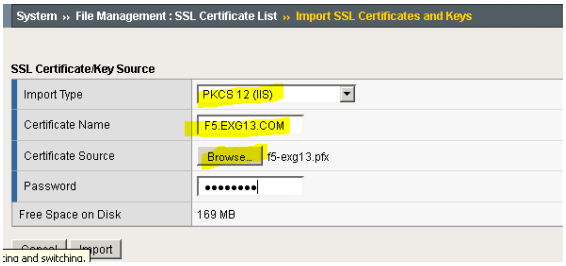
17. Final step is to export the Key Pair and save the .p12 file for import to F5.



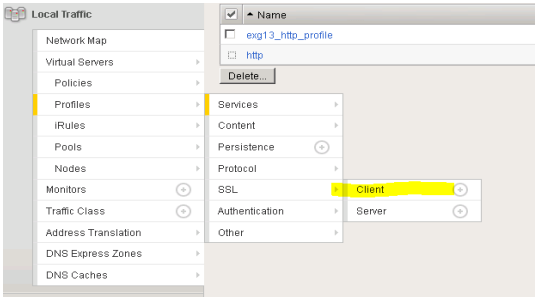


Configuring the F5 Client-Side SSL Profile

1. In the F5 GUI, select System, File Management, SSL Certificate List, Import SSL Certificates and Keys. Browse to the previously export .pfx file, provide a recognizable Certificate Name, the password used to export, and click "import" – this saves the certificate and private key into the F5 repository for use in setting up the server SSL profile.



2. Select Local Traffic, Profiles, SSL, Client – this will allow you to create a client profile for SSL authentication.



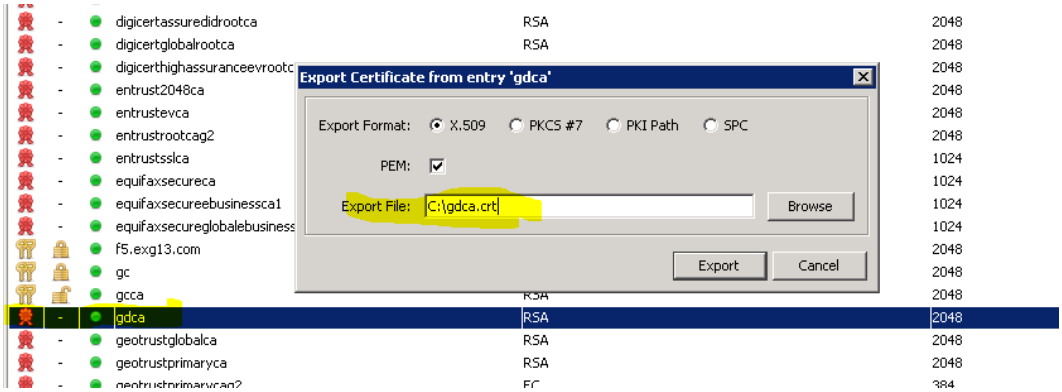
3. Select "Create" in the upper right side of the screen to create a new Client SSL profile.

   Do not change the parent profile clientssl, select the "custom" box on the right, and select the Certificate and Key file that match what was Imported in the previous step.

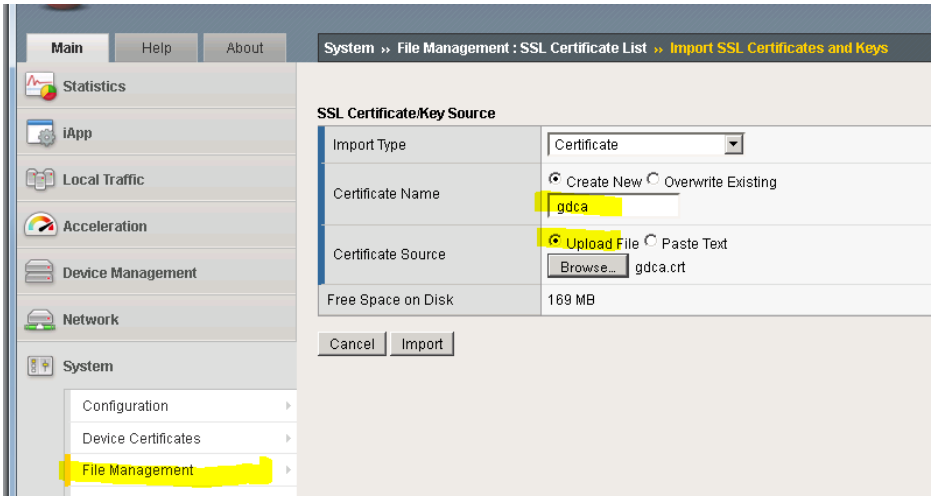
   In this example the name chosen was F5.EXG13-client: All other settings should not be altered on this page.

Configuring the Server-Side SSL Profile

1. In the Keystore Explorer tool, select the gdca public certificate and export to .crt file as shown below.

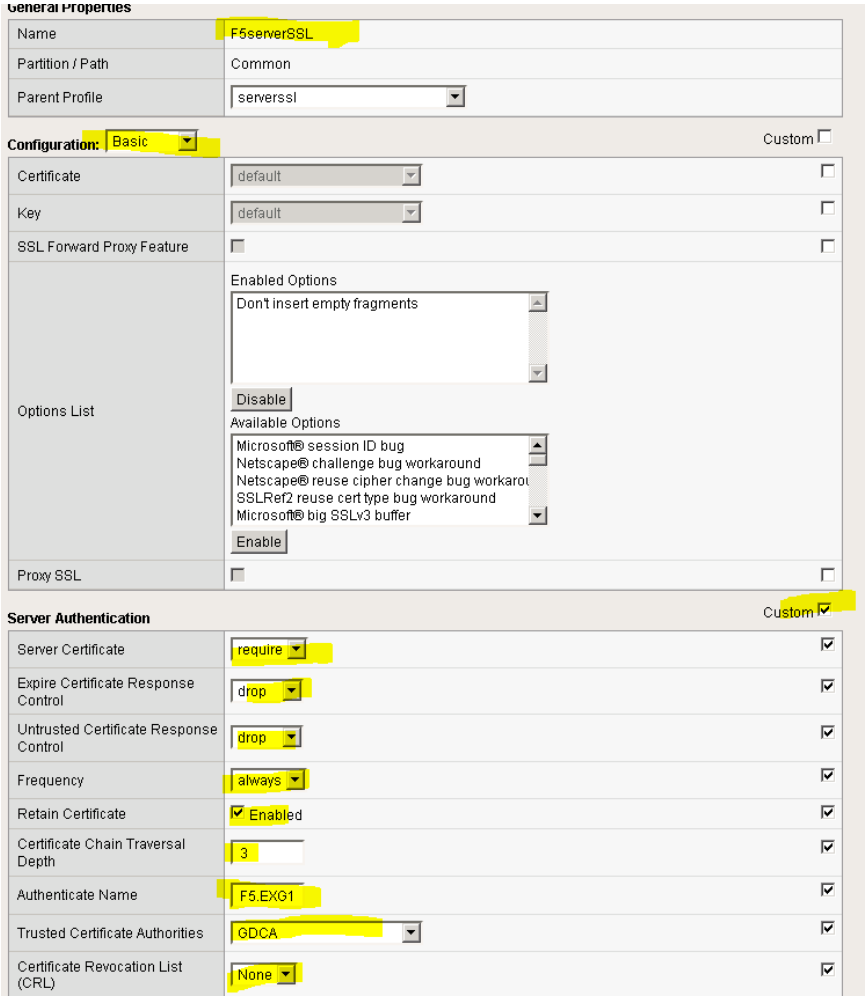


2. Import this file into the F5 SSL certificate store. This certificate is used as the Trusted Certificate Authority instead of the default cacerts bundle included with default F5 profile.



3. Select Local Traffic, Profiles, SSL, Server and create a New Server SSL profile, and name accordingly
4. Modify the server authentication details to include:
   - Server Certificate: Require
   - Expire Certificate Response Control: drop
   - Untrusted Certificate Response control: drop
   - Frequency: once – (can be set to always)
   - Retain Certificate: enabled
   - Certificate Chain traversal depth: 3
   - Authenticate Name – This must be the CN of your created certificate, in this example F5.EXG13.COM

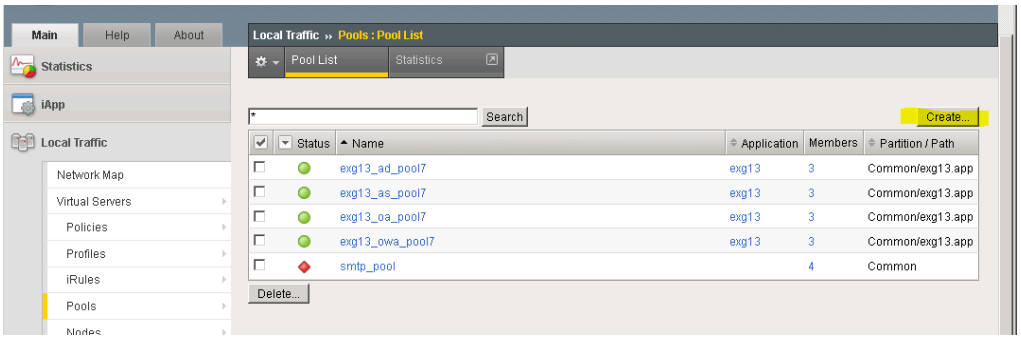
- Trusted Certificate Authority: GDCA. This must be the certificate you uploaded in a previous step. This allows the F5 to verify the certificate presented by the GP server(s) it establishes connections with to be validated against the GD systems proprietary CA.



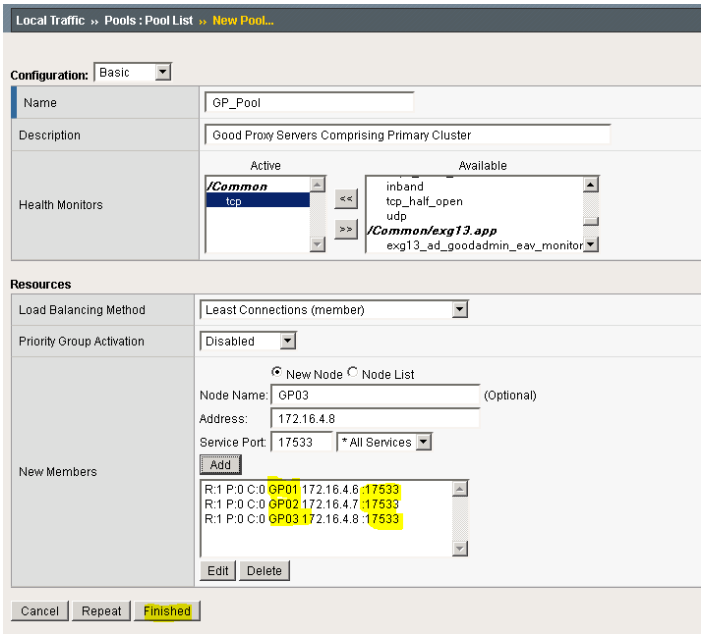
Configuring the F5 Server Pool

Each member of the GP cluster must also be a member of a pool of servers that F5 will distribute connections to. The method of distribution or load balancing used in this guide is “least-connections” although the actual method the client can choose can vary.

1. From the F5 console, navigate to Local Traffic, Pools, and then select “Create” in the top right.



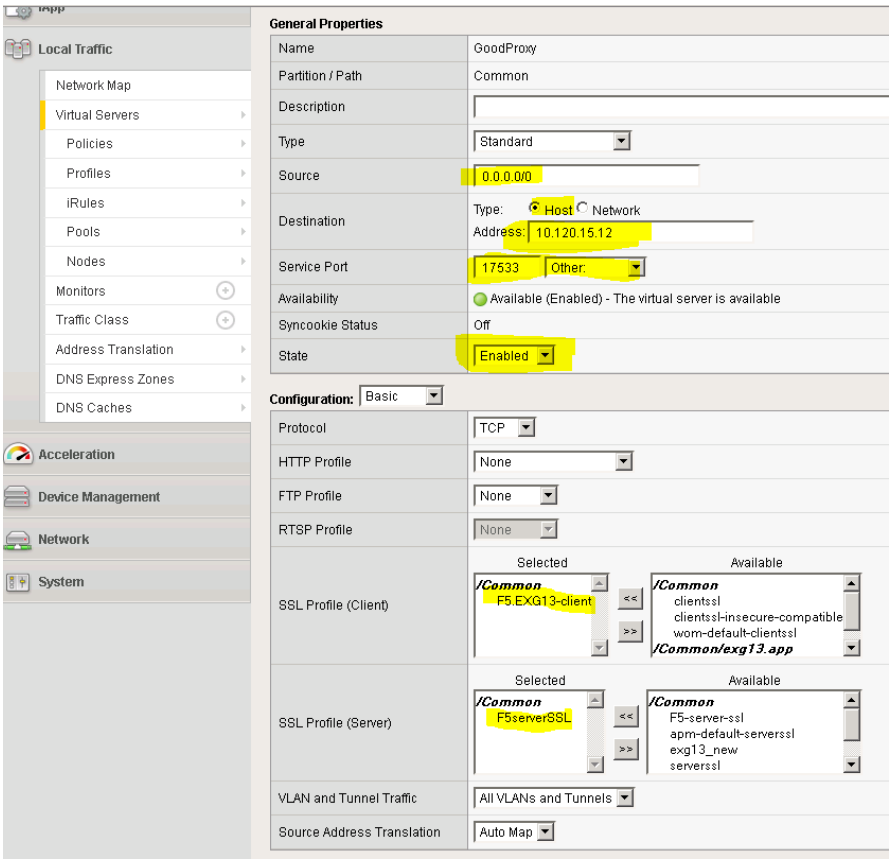
2. The pool name used in this example is GP_Pool, the health monitor is simple TCP, Load Balancing method “least connections” .
3. Each GP server in the cluster was given identifiable name and associated IP address, along with service port of 17533.
4. Do this for each member of your GP cluster for which the F5 will balance connections.



Configuring the F5 Virtual Server

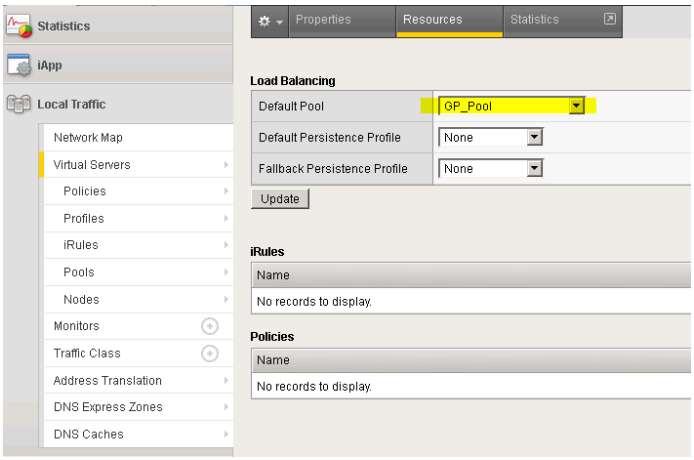
1. From the F5 GUI, go to Local Traffic, Virtual Servers, and select “Create” to create a new Virtual server. This Virtual Server will be the perimeter facing IP address which is NAT’d to from Public IP, or in some cases this could be the actual public IP address which the GD secured applications will make their initial connection to.
2. Source is 0.0.0.0/0 because we will be accepting connections from IP addresses anywhere on the public Internet

3. Destination will be the perimeter IP address of the F5, in this lab this is the IP address on the internal LAN which is NAT'd to from the Public Interface of Internet Edge Router.
   - Service port must be 17533.
   - Configuration = Basic.
   - Protocol = TCP.
   - SSL Profile (Client) = select the profile created in step 5 above.
   - SSL Profile (Server) = select the custom SSL server profile created in step 6 above.
   - Choose Source Address Translation = Auto-Map (could vary depending on configuration).
   - ALL other values can be left at defaults.



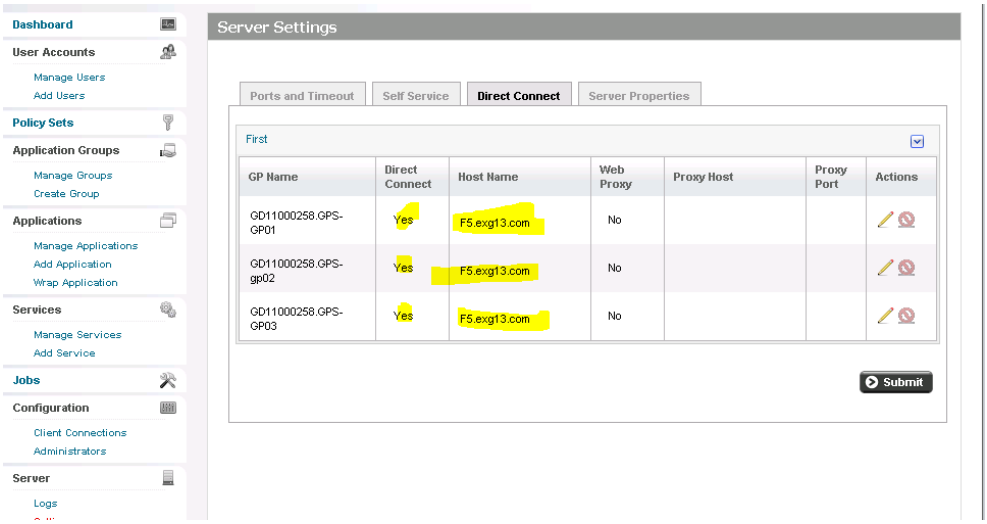
4. Next select Local Traffic, Virtual Servers, GoodProxy (name chosen in previous step), and select "Resources".

5. Select the Default Pool to be associated with this Virtual Server you created previously.



Configuring Good Console Settings

1. Any GP server that is a member of a cluster must be configured identically. All members of a cluster must be either Direct Connect enabled or disabled. Broken connections and undesired behavior will result if settings are not uniform.
2. Each member of the GP cluster should be set to Direct Connect = Yes
3. Each member of the GP cluster should have its "Host Name" set to the name identified as the public FQDN of the listener on the F5 reverse proxy – i.e. the Common Name of the Certificate created in the beginning of the configuration.
4. Do not enter anything for the Proxy Host field.



List of Supported SSL Ciphers between GC and GP Servers for Direct Connect

The complete list of supported ciphers is below. These are valid values for the GP server's property file `c:\good\gps.properties` and the `gps.directconnect.supported.ciphers` key.

SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA

SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA

SSL_DHE_DSS_WITH_DES_CBC_SHA

SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA

SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA

SSL_DHE_RSA_WITH_DES_CBC_SHA

SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA

SSL_DH_anon_EXPORT_WITH_RC4_40_MD5

SSL_DH_anon_WITH_3DES_EDE_CBC_SHA

SSL_DH_anon_WITH_DES_CBC_SHA

SSL_DH_anon_WITH_RC4_128_MD5

SSL_RSA_EXPORT_WITH_DES40_CBC_SHA

SSL_RSA_EXPORT_WITH_RC4_40_MD5

SSL_RSA_WITH_3DES_EDE_CBC_SHA

SSL_RSA_WITH_DES_CBC_SHA

SSL_RSA_WITH_NULL_MD5

SSL_RSA_WITH_NULL_SHA

SSL_RSA_WITH_RC4_128_MD5

SSL_RSA_WITH_RC4_128_SHA

TLS_DHE_DSS_WITH_AES_128_CBC_SHA

TLS_DHE_DSS_WITH_AES_128_CBC_SHA256

TLS_DHE_DSS_WITH_AES_256_CBC_SHA

TLS_DHE_DSS_WITH_AES_256_CBC_SHA256

TLS_DHE_RSA_WITH_AES_128_CBC_SHA

TLS_DHE_RSA_WITH_AES_128_CBC_SHA256

TLS_DHE_RSA_WITH_AES_256_CBC_SHA

TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

TLS_DH_anon_WITH_AES_128_CBC_SHA

TLS_DH_anon_WITH_AES_128_CBC_SHA256

TLS_DH_anon_WITH_AES_256_CBC_SHA

TLS_DH_anon_WITH_AES_256_CBC_SHA256

TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

TLS_ECDHE_ECDSA_WITH_NULL_SHA

TLS_ECDHE_ECDSA_WITH_RC4_128_SHA

TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

TLS_ECDHE_RSA_WITH_NULL_SHA

TLS_ECDHE_RSA_WITH_RC4_128_SHA

TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA

TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA

TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256

TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA

TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384

TLS_ECDH_ECDSA_WITH_NULL_SHA

TLS_ECDH_ECDSA_WITH_RC4_128_SHA

TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA

TLS_ECDH_RSA_WITH_AES_128_CBC_SHA

TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256

TLS_ECDH_RSA_WITH_AES_256_CBC_SHA

TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 = Default

TLS_ECDH_RSA_WITH_NULL_SHA

TLS_ECDH_RSA_WITH_RC4_128_SHA

TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA

TLS_ECDH_anon_WITH_AES_128_CBC_SHA

TLS_ECDH_anon_WITH_AES_256_CBC_SHA

TLS_ECDH_anon_WITH_NULL_SHA

TLS_ECDH_anon_WITH_RC4_128_SHA

TLS_EMPTY_RENEGOTIATION_INFO_SCSV

TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5

TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA

TLS_KRB5_EXPORT_WITH_RC4_40_MD5

TLS_KRB5_EXPORT_WITH_RC4_40_SHA

TLS_KRB5_WITH_3DES_EDE_CBC_MD5

TLS_KRB5_WITH_3DES_EDE_CBC_SHA

TLS_KRB5_WITH_DES_CBC_MD5

TLS_KRB5_WITH_DES_CBC_SHA

TLS_KRB5_WITH_RC4_128_MD5

TLS_KRB5_WITH_RC4_128_SHA

TLS_RSA_WITH_AES_128_CBC_SHA

TLS_RSA_WITH_AES_128_CBC_SHA256

TLS_RSA_WITH_AES_256_CBC_SHA

TLS_RSA_WITH_AES_256_CBC_SHA256 = Default

TLS_RSA_WITH_NULL_SHA256

Good Dynamics Documentation

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

Category	Title	Description
Cross-	• *Getting Started Guide for Marketplace*	Overviews of the Good Dynamics system

Category	Title	Description
platform	*Partners* • *Good Dynamics Platform Overview for Administrators and Developers* • *Good Cloud Deployment*	
	• *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 • 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.	

Category	Title	Description
	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*

Server Clustering and Affinities

Last updated: January 15, 2015

Versions GP 2.0.*xx.yy*, GC 2.0.*xx.yy*

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

© Copyright 2015. All rights reserved. All use is subject to license terms posted at www.good.com/legal. GOOD, GOOD TECHNOLOGY, the GOOD logo, GOOD FOR ENTERPRISE, GOOD FOR GOVERNMENT, GOOD FOR YOU, GOOD APPCENTRAL, GOOD DYNAMICS, SECURED BY GOOD, GOOD MOBILE MANAGER, GOOD CONNECT, GOOD SHARE, GOOD TRUST, GOOD VAULT, and GOOD DYNAMICS APPKINETICS are trademarks of Good Technology Corporation and its related entities. All third-party technology products are protected by issued and pending U.S. and foreign patents.

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

Table of Contents

Server Clustering and Affinities Introduction

Server Clustering and Affinities are two separate yet dependent Good Dynamics features.

Clustering is a widely known technology in which a group of servers, a cluster, is deployed as a single node that provides a service. From the client point of view, all servers in the cluster are interchangeable and any server could be used to access a required service. The Good Dynamics server clustering feature enables this type of deployment in the Good Dynamics enterprise infrastructure.

Each of the following servers in the Good Dynamics (GD) enterprise infrastructure can be deployed in clusters:

- Good Proxy (GP)
- Good Control (GC)
- Application servers

This means that enterprises can deploy additional instances of these servers in a way that contributes to the strategic benefits of clustering by:

- Scaling up the capacity for concurrent users.
- Delivering high availability and disaster recovery (HA/DR).

Use of GP clustering is also a prerequisite for use of the Affinities feature.

Affinities feature enables an enterprise to allocate its Good Proxy servers between its Good Control servers and its application servers. Allocation of Good Proxy (GP) servers can be an absolute division, or based on a priority order, or both. Either kind of allocation is referred to as an affinity.

Affinities with server clustering can be used to scale up the number of users that can be serviced, in addition to being part of an enterprise HA/DR strategy. There are other reasons for allocating GP servers between GCs and application servers.

For instance, an enterprise's GC and application servers may be installed in a number of different geographic locations. Affinities can be used to allocate GPs to servers that are in the same geographic location to improve the speed of data transfer.

Or, an enterprise may assess that some of its data is especially sensitive and wish to segregate the servers that handle this data by putting additional protections in place. For example, the servers could be installed in a special secure server room. Affinities could then be used to allocate GPs to these servers, and the GPs could be segregated and protected in the same way.

Allocation of GP servers can also align with the enterprise organization's customary division of resources, for example by budget holder or line of business.

Whatever the motivation for allocation, the feature works in the same way.

Because the server clustering and affinities features are enabler-type features, configuration is required at the enterprise in order to realize their benefits. The GC management console is the user interface in which the server clustering and affinities features are configured.

Once the configuration is in place, mobile GD applications will follow the configuration when setting up connections with servers that are behind the firewall. This is mostly implemented in the GD runtime, although support for clustered application servers also requires implementation in the GD application code layer.

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.

Server Clustering and Affinities Terminology

A group of GD servers deployed as a single node is referred to as a **GD server cluster**. All servers in a cluster must be instances of the same server. For example, all the servers in a cluster could be GC servers, or all could be GP servers, or all could be instances of the same application server.

Any GD server can be in only one server cluster.

A server cluster of GPs can be referred to as a **proxy cluster**, or a GP cluster. Proxy clusters enable affinity relationships, see below.

The capabilities of the server clustering and affinities features apply equally to GC servers and application servers. Either a GC server or an application server can be referred to as a **host server**. This term is used because these servers can be thought of as hosting services, with Good Control being an example of a service. A GP server cannot be referred to as a host server.

A cluster of host servers can be referred to as a **host cluster**. A GD mobile application can connect to any of the servers in the host cluster in order to access the service provided. This means that the service can still be accessed when some of the servers in the cluster are over capacity, offline or otherwise inaccessible.

Any host server can be assigned a **server priority**, typically stated as primary, secondary, etc. Server priorities apply relative to other instances of the same server. A GD mobile application will attempt to connect to higher priority servers before lower priority servers.

An **affinity** is an allocation of a proxy cluster to a host server. Connection from a GD mobile application to a host server that is behind the firewall can only be conveyed by a GP in a proxy cluster with which the host server has an affinity. Within the proxy cluster, any GP can be used to convey the connection. This means that a connection can still be conveyed when some of the GPs in the cluster are over capacity, offline or otherwise inaccessible.

A single host server can have affinity relationships with multiple proxy clusters, and vice versa. An **affinity relationship** can be assigned an affinity priority, typically stated as primary, secondary, etc. Affinity priorities apply relative to other affinity relationships of the same host server. A GD mobile application will attempt to connect via GPs in clusters with which the host server has a higher affinity priority before attempting to connect via proxy clusters with lower affinity priority.

For connections to the GC, the GD runtime implements both selection of the host server by priority, and selection of the GP according to affinity. For connections to application servers, the GD runtime implements only the GP selection part. The application code implements its own selection of host server. The details of the application server's cluster and priority configuration are made available to the application layer within the runtime API.

Relationship to Kerberos Constrained Delegation

Kerberos Constrained Delegation (KCD) is supported with Good Dynamics clusters, subject to the details in the Good KCD documentation and described below:

1. For normal Kerberos (not KCD), the GC and the GP are installed in the same domain. This is not required for KCD.
2. In Good Control's **Clusters** > **GP Clusters** tab, define the cluster using only GPs in the same domain as the GCs.
3. In Good Control's **Clusters** > **GC Clusters** tab, specify only the first GP cluster. Do not specify a secondary GP cluster.

These settings are for the simplest, most easily defined multi-realm KCD. Other settings are possible, with many different permutations affecting the complexity of the settings.

Feature Highlights

The server configuration diagrams that follow quickly highlight key aspects of GD clustering and affinities:

- GC Servers with Simple Affinities and Priorities
- GC Servers with Multiple Affinities and Shared Priorities
- Component Interaction
- Application Servers

GC Servers with Simple Affinities and Priorities

Figure below shows a fairly simple deployment. There are two GC servers, each of which has an affinity to a GP cluster. In accordance with the table, the two GC servers each have a different priority.

Server	Address	Server Priority	Affinities
GC 1	gc1.intranet.example.com	Primary	GP Cluster A
GC 2	gc2.intranet.example.com	Secondary	GP Cluster B

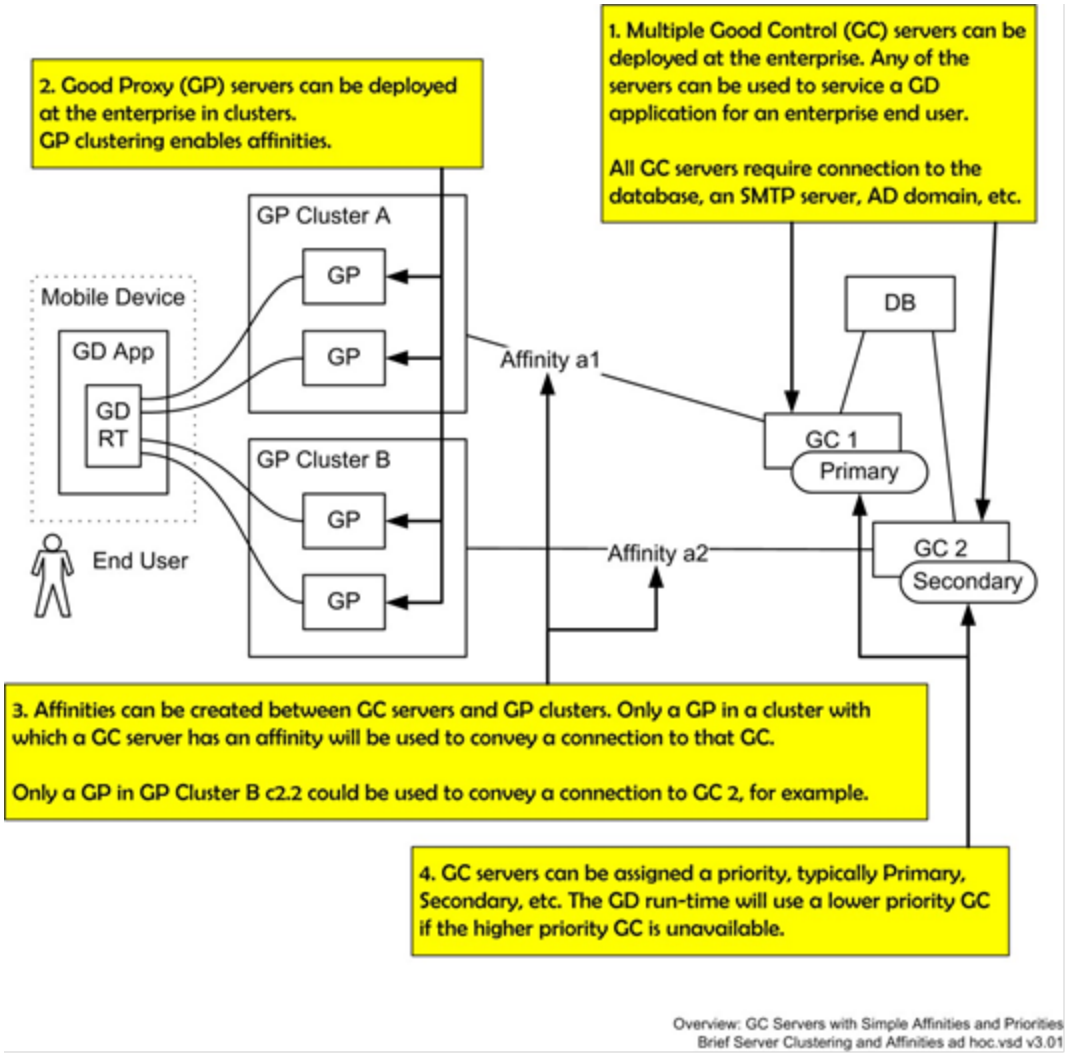


Figure 1: GC Servers with Simple Affinities and Priorities

GC Servers with Multiple Affinities and Shared Priorities

Figure below shows a more advanced deployment than the clustering scheme. In this case, there are three GC servers. Two of the GC servers have primary priority, the other has secondary priority. One of the GC servers has affinities to two GP clusters and the affinities have a priority order.

Server	Address	Server Priority	Primary Affinity	Secondary Affinity
GC 1	gc1.intranet.example.com	Primary	GP Cluster A	GP Cluster B
GC 2	gc2.intranet.example.com	Secondary	GP Cluster B	
GC 3	gc3.intranet.example.com	Secondary	GP Cluster B	

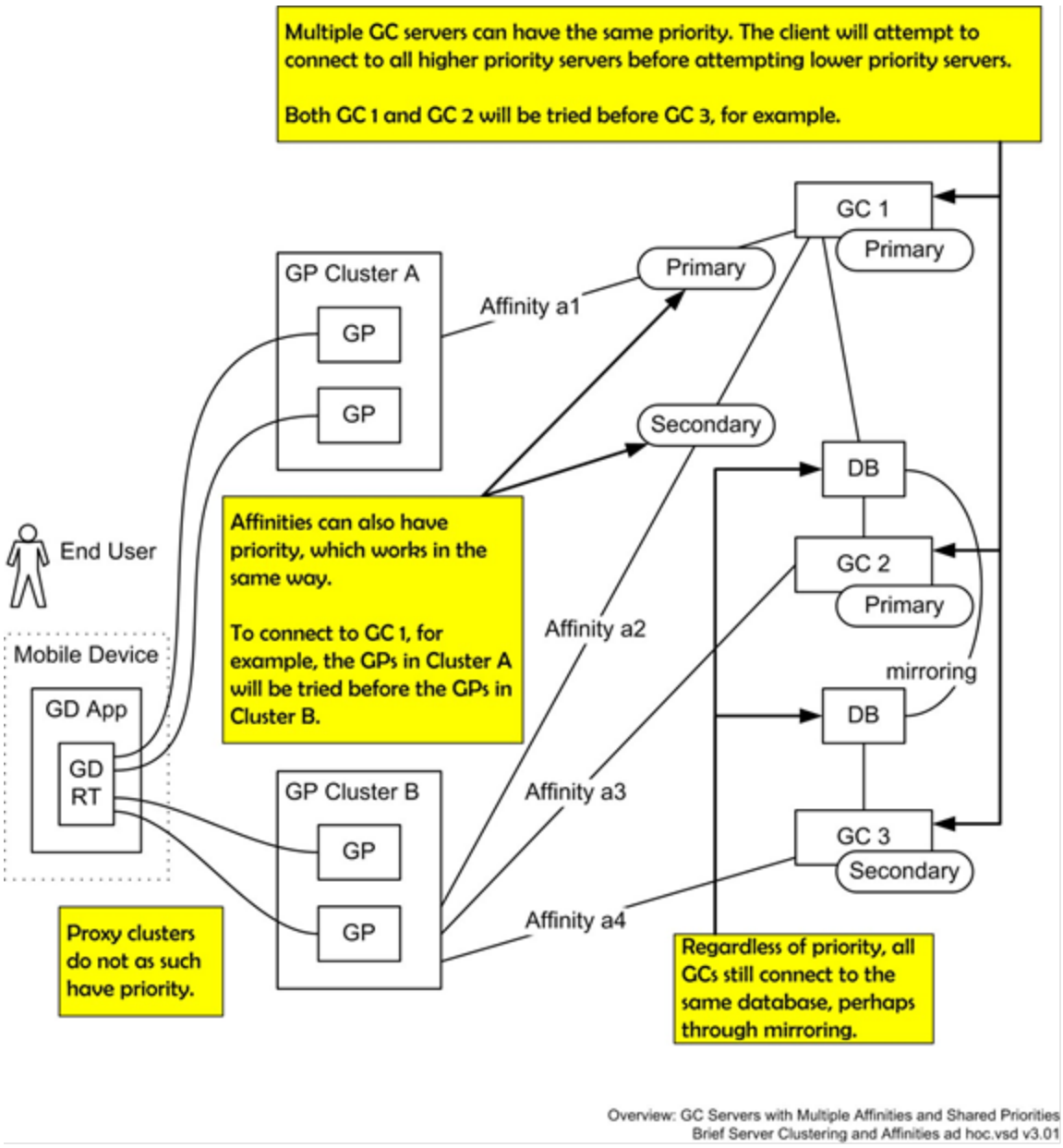


Figure 2: GC Servers with Multiple Affinities and Shared Priorities

Component Interaction

Figure below shows the component interactions that take place when connecting to the GC from the mobile GD application, in accordance with the priorities and affinities in the table.

Server	Address	Server Priority	Primary Affinity	Secondary Affinity
GC 1	gc1.intranet.example.com	Primary	GP Cluster A	GP Cluster B

Server	Address	Server Priority	Primary Affinity	Secondary Affinity
GC 2	gc2.intranet.example.com	Secondary	Not shown	
GC 3	gc3.intranet.example.com	Secondary	GP Cluster B	
GC 4	gc4.intranet.example.com	Secondary	Not shown	

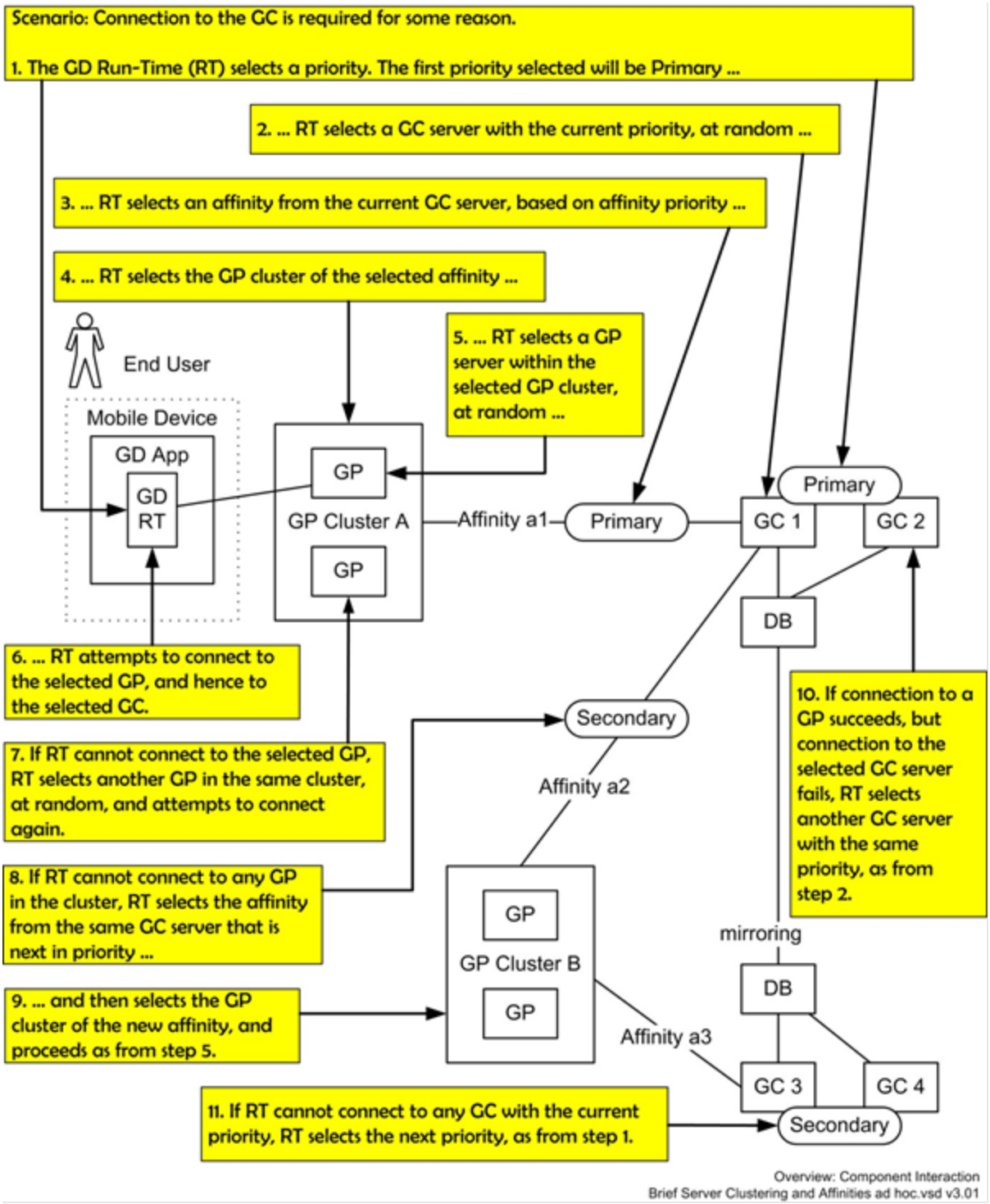


Figure 3: Component Interaction

Application Servers

Figure below shows deployment of clustered application servers. The configuration is similar to the configuration of the GCs shown in the previous diagram. The processing by the mobile GD application is different, however, in that selection of server now takes place in the GD application code, not in the GD run-time. Selection of proxy still takes place in the GD run-time.

Server	Address	Server Priority	Primary Affinity	Secondary Affinity
AS 1	as1.intranet.example.com:80	Primary	GP Cluster A	GP Cluster B
AS 2	as2.intranet.example.com:80	Primary	GP Cluster B	
AS 3	as3.intranet.example.com:80	Secondary	GP Cluster B	
These configuration details can be used by the GD application code layer.			These configuration details are only used by the GD runtime.	

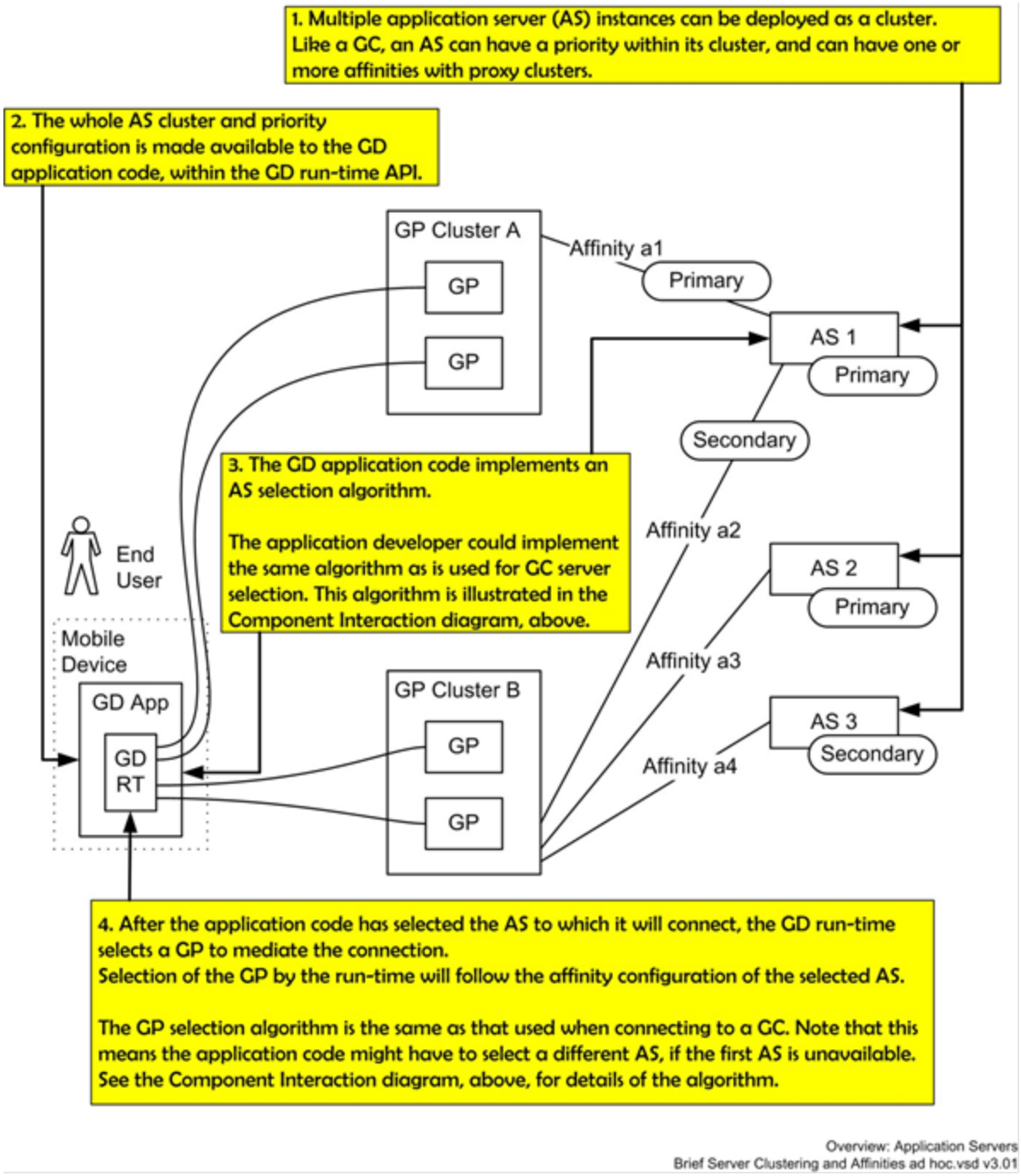


Figure 4: Clustered Application Servers

The balance of this document describes in additional detail the kinds of configurations supported, and describe how communication setup is affected.

Relationship to Cloud GC: Feature Not Applicable

The feature, service, server type, or software described in this guide is not available on Good Control Cloud because it is not applicable in a hosted environment.

Server Clustering

Server clustering is the deployment of a group of servers working as a single node. In Good Dynamics, a server cluster can be one of the following:

- Group of GC servers
- Group of GP servers
- Group of instances of the same application server

In GD, a server cluster does not consist of a mixture of servers that fulfil different roles.

To use server clustering, the enterprise must commission extra servers, then configure groups of the same servers into clusters. Once the configuration is in place, mobile applications will behave differently when setting up communication with servers.

A description of the configuration and communication setup for a number of clustering scenarios, starting with the simplest:

- Basic Good Control Clustering
- Good Control Clustering with Server Priorities
- Application Server Clustering
- Good Proxy Server Clustering
- Server Clustering: Limitations

Basic Steps to Create a Good Control Cluster

With a basic topology shown below, to deploy the simplest GC cluster, you install installs multiple GC servers that communicate with the same GC database.

Basic Steps:

1. Install the first GC server.
2. As administrator, login to that first GC server's console.
3. To cluster the other GC servers, follow the remaining steps detailed in the consoles online help topic **Advanced Server Configuration > Defining and Managing GP Server Clusters**.

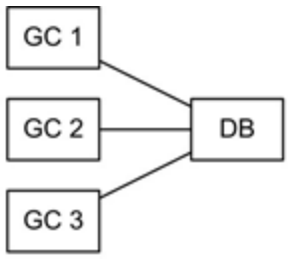


Figure 5: Basic GC Clustering

In a single GC cluster:

- Communication with GD applications is distributed between the servers in the cluster. Load distribution is generally random. This has a benefit for scale.
- If some of the GC servers in the cluster are over capacity, offline or otherwise inaccessible, the GC service as a whole is still accessible, through the remaining GC servers. This has a benefit for HA/DR.

There are some differences in the installation and other administrative procedures between a clustered GC and a single-server GC. There are also differences in the activation of, and communications with, GD applications.

Basic Clustering: Installation and System Administration

Installation of GCs in a cluster is similar to, though not exactly the same as, installation of a single GC server.

The license key for the first GC that an enterprise installs is always obtained from the Good Developer Network (GDN) website, whether or not clustering is going to be used. The license keys for subsequent GC installations that will form part of a cluster are obtained from the GC console user interface, not from the GDN. Any of the enterprise's GCs can be used to generate license keys for clustered installation. A copy of the key will be stored in the central GD Network Operation Center (NOC), so that the new GC will be able to connect to the GD infrastructure.

The license key is used in the GC installation procedure in the usual way. After installation, the operation of a GC cluster differs from the operation of a single server GC in a number of respects.

Each GC in the cluster has its own instance of the GC console user interface. The same information will generally appear in the consoles of all clustered GCs. This is because policies, entitlements, client connections and all other information that appears in the GC console is either stored in the GC database (DB), or retrieved from the NOC.

When a change is made in one GC console in a cluster, the GC that hosts the console updates the DB.

Some changes require provisioning to the NOC or the enterprise GP servers. For example, changing an application server's address or port number requires provisioning to the GP servers. For configuration changes that require provisioning, all GCs in the cluster will attempt to send the change to all GP servers. Only one attempt per GP needs to succeed for the change to be fully provisioned. This means that every GP needs a viable communication link with at least one GC. It is not necessary that all GPs are able to receive communication from all GCs.

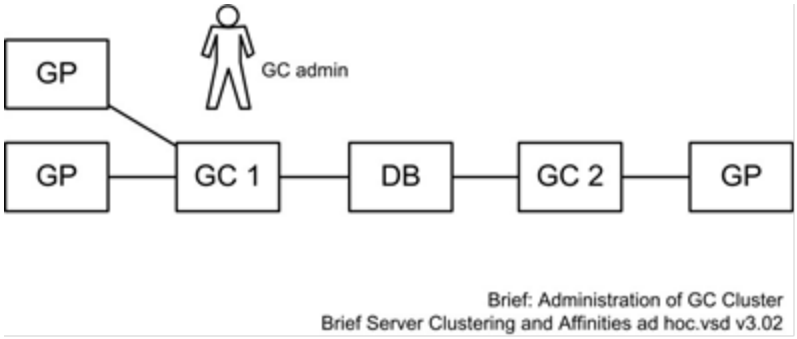


Figure 6: GC Cluster Administration

Important: There is no locking in the GC console. Actions taken in different console instances will be applied to the database in order of completion. For example, if two GC administrators edit the same policy set in two different GC consoles at the same time, one change will be overwritten by the other.

In the same way that every GC hosts an instance of the GC console user interface, every GC also hosts an instance of the Good Control web services. See Applications Programming Interface for more details of how to access GC web services when server clustering is in use.

From a general network administration point of view, servers in a GC cluster do not have any special relationship. The cluster does not require a shared IP address, for example.

During installation of a GP, the address of a GC server must be entered. Where a GC cluster is in use, the address of any server in the cluster can be used for this purpose.

Note that deployment of a GC cluster does not deliver any direct scale or HA/DR benefits for the GC database. The DB could still be protected using a solution offered by the DB vendor. For example, the vendor might offer database clustering or mirroring solutions.

Basic Clustering: Application Communication

The first communication between a newly installed GD application and a GC server will be enterprise activation. Enterprise activation takes place just after the end user has entered their access key in the GD runtime library user interface.

If the enterprise has a GC cluster then one server in the cluster will be chosen at random to process each enterprise activation. The chosen GC server will send the relevant policy and other configurations to the GD runtime (GD RT in figure below) as part of this processing. The GD runtime will then establish a Push Channel connection with a GC server. The Push Channel will be used to notify the runtime of any updates to policies, entitlement or other configuration.

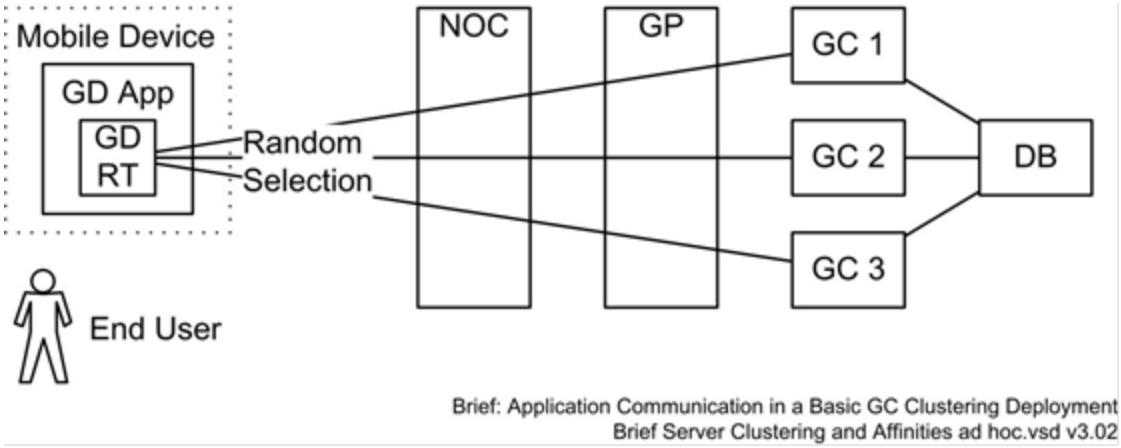


Figure 7: Application Communication in a Basic GC Clustering Deployment

Each time the GD runtime makes a new connection, a different GC server could be chosen. There is no permanent association between the GD runtime and any particular GC server in the cluster. The association lasts only for the lifetime of the Push Channel.

Note that the GD runtime instances in different GD applications on the same device could be communicating with different GC servers in the cluster.

Basic Clustering: Resilience

GC clusters are more resilient than single-server GCs. Cluster resilience is implemented as follows.

Connection setup from a GD runtime to the GC server that it chooses randomly will fail if the GC server is offline. Connection would also fail if the randomly chosen GC server rejected the connection. The server could do this if the server was experiencing a high processing load, for example.

Note: A single-server GC could also reject connections due to load. In that scenario, Good Control services, such as enterprise activation, would be unavailable to applications that had not already connected to the GC.

Whatever the reason, when connection to the first chosen GC server fails, the GD runtime chooses another GC server and attempts connection to that server instead. This choice is also made at random, between the untried servers. This continues until a connection attempt succeeds, or until there are no more untried servers.

When all servers have been tried, and no connection attempt has succeeded; i.e., all the GC servers are off line, the GD runtime will enter a back-off and retry cycle. The algorithm used to determine the back-off period is the same as the algorithm that would be used if there was a single-server GC with which a connection attempt had failed.

The foregoing scenario, wherein a GC is offline when a connection attempt is being made, is one possible failure. Another possible failure is the scenario that a GC goes off line sometime after a connection attempt has succeeded, and a Push Channel is in place.

In the scenario that a GD runtime has established a Push Channel with a GC instance, and that GC then goes off line, the GD runtime will be notified. Notification, and detection, of this condition uses the Ping Fail capability of the GD Push Channel framework.

When the GD runtime receives the Ping Fail notification, it will attempt to re-connect to a GC server. This attempt will be to a GC server chosen at random, as described above for a new connection.

These resilience features are implemented by the GD runtime and infrastructure platform, and do not require special coding in the application layer.

Good Control Clustering with Server Priorities

GC servers in a clustered deployment can be assigned server priorities. The server priority dictates the order in which the GC servers will be tried when connection is required by a GD application. Typically referred to as primary, secondary or tertiary, server priorities can also be considered numeric. Each GC server can be assigned a different priority, or multiple GC servers can share the same priority, as illustrated in figure below.

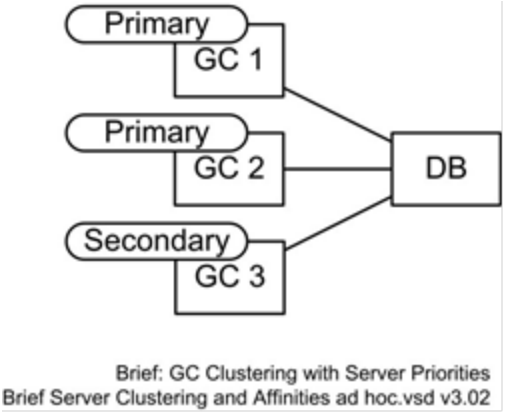


Figure 8: GC Clustering with Server Priorities

Assigning server priorities has no impact on the need to connect all GC servers to the same GC database. However, it is perhaps more likely that database mirroring would be used here. For example, all primary GC servers could connect to the main database instance, and secondary GC servers could connect to a mirror.

There are some differences in the installation and other administrative procedures when server priorities are being used. There are also differences in the activation of, and communication with, GD applications.

Clustering with Server Priorities: Installation and System Administration

Initial installation of a GC server that will have an assigned server priority is the same as the installation of a GC server in a basic cluster with no server priorities.

Configuration of server priorities is carried out in the GC console user interface, like any other configuration of the GD infrastructure within the enterprise. Server priority configuration consists of such tasks as:

- Assigning the priority of a new GC server.
- Changing the priority of an existing GC server.

Note: Reassigning server priority might necessitate editing the address of the database server in the configuration of the GC itself, if a different database mirror with a different address is used by servers of different priority.

Any GC console in an enterprise can be used to configure server priorities.

Clustering with Server Priorities: Application Communication

Shown in figureApplication Communication in a GC Deployment with Server Priorities, communication between GD applications and clustered GCs with server priorities works as follows.

When connection to a GC is required, a GD runtime will first try the GC with the highest server priority. If connection to that server fails, then the GD runtime tries the GC with the next highest server priority, and so on. If there is more than one GC at a particular priority, then the GD runtime tries all of them, in a random order, before moving on to try GCs at the next lower priority.

In other respects, application communication is the same as it is for a basic GC clustered deployment.

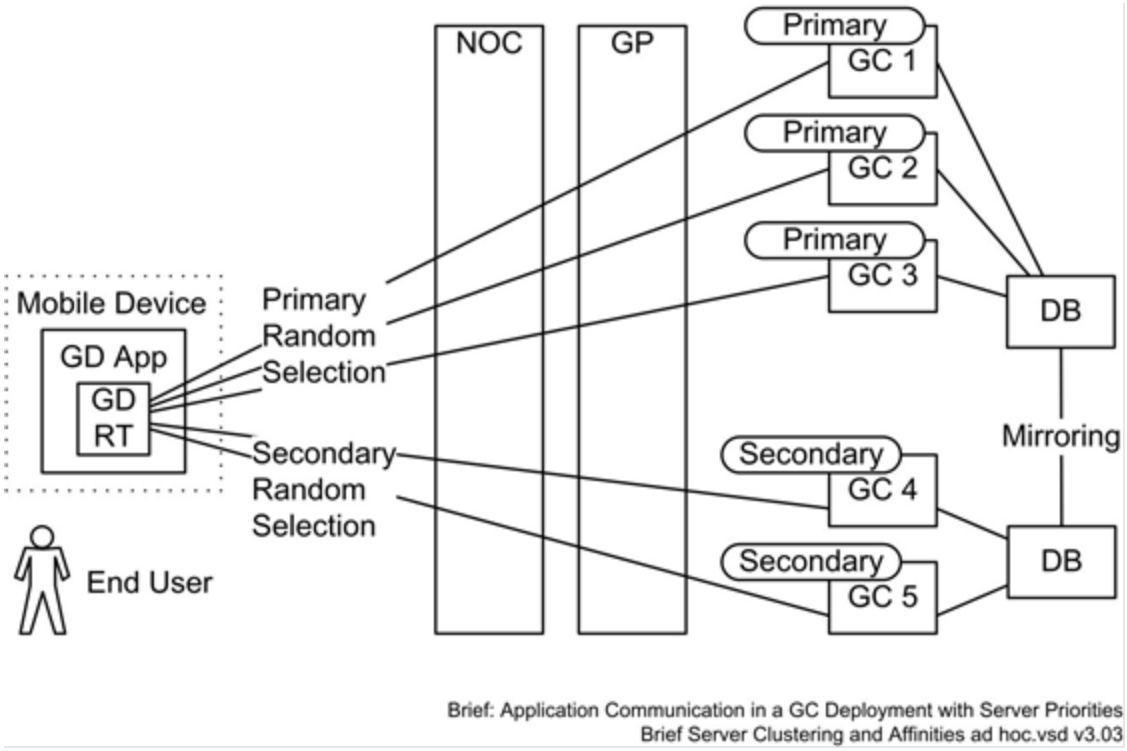


Figure 9: Application Communication in a GC Deployment with Server Priorities

Application Server Clustering

The software for an application server is provided by the vendor of a GD application. The vendor may implement their own HA/DR and scaling mechanisms for the application server. However, application server clustering capabilities are also provided as part of the server clustering feature of GD.

Application servers can be deployed in the same ways as GC servers: as single-server nodes, or in a cluster. If an application server is deployed as a cluster, then the individual servers can be assigned server priorities.

As for the GC DB, the GD server clustering feature does not specifically provide HA/DR of any databases to which the application servers connect. The application provider should typically put in place database mirroring or some other HA/DR measures at the back end of their application.

Application server clusters and server priorities are configured in the GC console, in the same way as GC clusters (see Basic Clustering: Installation and System Administration and Clustering with Server Priorities: Installation and System Administration on installation and system administration). The GD runtime does not implement server selection for application servers. Instead, the mobile application is given access to a structured representation of the application server clustering configuration. The structure includes server priorities, addresses and port numbers. The application could then implement the same prioritized server selection algorithm as the GD runtime implements for communication with GC servers (see Basic Clustering: Application Communication and Clustering with Server Priorities: Application Communication on application communication).

If the application uses Push Channel notifications, then the Ping Fail capability could be used for resilience. See Basic Clustering: Resilience for a description of how the GD runtime uses the capability with server clustering.

Note: The application could instead implement its own variant of the GC cluster communication setup algorithm. If the application's own variant requires additional configuration, then the application-specific policy feature of Good Dynamics could perhaps be used to provide that.

Good Proxy Server Clustering

A single implicit GP cluster has been supported since the first release of GD. If more than one GP server was installed in a deployment, the GD runtime would select one at random when establishing a connection to a server behind the firewall.

The server clustering feature adds support for multiple clusters of GP servers. The main use of this is for configuring affinities, see Affinities for details.

A GP cluster differs from a GC or application server cluster in a number of ways.

Good Proxy Selection

Selection of a GP can be based on a priority, but this is part of the affinities feature. GP servers and clusters do not as such have priorities. See Affinities for details.

The membership of a GP server in a GP cluster is explicit in the configuration, as is the existence of multiple GP clusters. GP clusters have names. This means that the GC console offers options such as:

- Create a new GP cluster.
- Assign a new GP to a GP cluster.
- Move a GP from one cluster to another.

By contrast, the configuration of GC clustering is to some extent implicit. Individual GC and application servers are configured and assigned priorities, but there is no configuration at the group level.

Like the servers in a GC cluster, the servers in a GP cluster can be considered interchangeable. All GP servers in a cluster offer the same services.

Good Proxy Services

The main function of the GPs at an enterprise is to proxy connections to servers that are located behind the enterprise firewall. However, the GPs at an enterprise have a secondary function, which is to act as an access point for a number of services that can be used by application servers. For example, the Push Channel server-side API can be accessed in this way.

Access to these services is different when server clustering is in use. See Applications Programming Interface (API) for details.

Server Clustering Limitations

The feature supports clustering of the principal application server for a GD application, but does not support clustering of any additional servers. A single mobile application can communicate with multiple back-end application servers, but can only retrieve the application server cluster configuration for its principal server.

Only three levels of server priority are supported: primary, secondary and tertiary. There is no limit on the number of proxy clusters.

Affinities

Affinities are allocations of GP servers to GC servers and application servers. The benefits and motivations for making these allocations are outlined in Server Clustering and Affinities Introduction.

In Good Dynamics, an affinity can be configured between:

- A GC server and a cluster of GP servers.
- An application server and a cluster of GP servers.

The affinities feature works in the same way for GC servers and application servers, so a server of either type may be referred by the same term, host server, in the following description. Conversely, a cluster of GP servers may be referred to as a proxy cluster or GP cluster.

To use affinities, the enterprise must organize their GP servers into multiple clusters, then configure the allocation of proxy clusters to host servers. Once the configuration is in place, mobile GD applications will behave differently when setting up connections with servers that are behind the firewall.

The sections that follow describe the configuration and communication setup in a number of affinity scenarios, beginning with the simplest.

Simple Affinities

The most simple affinity configuration is to allocate each proxy cluster to a single host server, and vice versa. In this type of simple affinity configuration, connections are made as follows:

- Connections to a host server via a GP from the proxy cluster with which the host server has an affinity.
- A GP server to use is selected at random from within the cluster.

If the initially selected GP is over capacity, offline or otherwise inaccessible, then another GP in the same cluster is selected, again at random. This continues until an accessible GP is found, or until there are no more untried GP servers in the cluster.

If all the GPs in the proxy cluster are tried, and none are accessible, then a connection cannot be made to the required host.

The above connection setup procedure is followed by the GD runtime for all host servers. This is different to server clustering where the GD runtime only implements the whole procedure for the GC, not for application servers.

Note: The GD runtime for a single application might connect to a number of different GPs, in different clusters, in order to connect to the GC and one or more application servers.

Affinities: Resilience

Larger clusters, those with more servers, are expected to be more resilient than smaller clusters. A host server with a simple affinity to a larger proxy cluster may have higher effective availability than a host server with a simple affinity to a smaller proxy cluster.

As illustrated in figure below:

- The GC has an affinity with GP Cluster A, and could receive connections from either GP in that cluster.
- The application server (AS) has an affinity with GP Cluster B, and could receive connections from any of the GPs in that cluster.
- The GD runtime on the mobile device could make connections to any of the GP servers in either cluster.
- More GP resources are dedicated to AS than to GC. AS therefore has higher availability, other factors being equal.

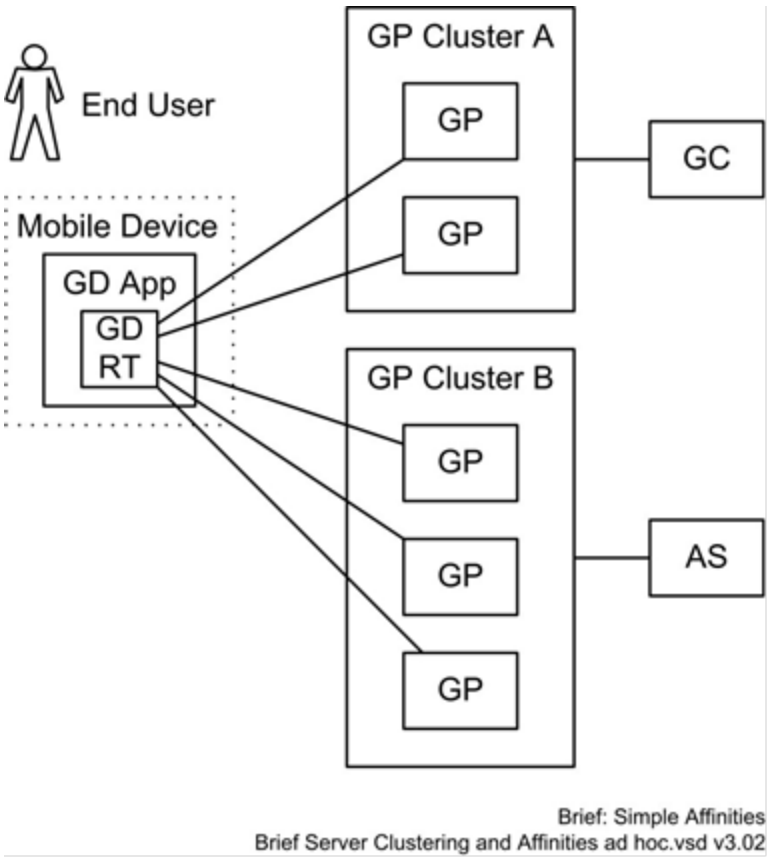


Figure 10: Simple Affinities

Simple Affinities: Configuration

Affinity setup is carried out in the GC console user interface, like any other configuration of the GD infrastructure within the enterprise.

The GC server sends the whole affinity configuration to the GD runtime of every connected GD application. The configuration is first sent at activation time, and then sent again whenever the configuration changes.

Shared Affinities

A proxy cluster can be allocated to more than one host server. This is a shared affinity.

The rules for connecting through a shared affinity configuration are the same as the rules for a simple affinity configuration, above. The GD runtime selects a GP from a proxy cluster that has an affinity with the host server to which connection is being attempted. The difference is that the same proxy cluster could also be used for connection to a different host.

As illustrated in figure below:

- The GC has an affinity with GP Cluster A, and could receive connections from either GP in that cluster.
- The application servers AS 1 and AS 2 both have an affinity with GP Cluster B, and could receive connections from any of the GPs in that cluster. This is a shared affinity.
- The GD runtime on the mobile device could make connections to any of the GP servers in either cluster.
- GC has dedicated proxy server resources. This means that the route to the GC need not become congested when the application servers are busy.
- The configuration could reflect that AS 1, AS 2, and the servers in GP Cluster B are all in one geographic location, whilst the GC and servers in GP Cluster A are in a different geographic location.

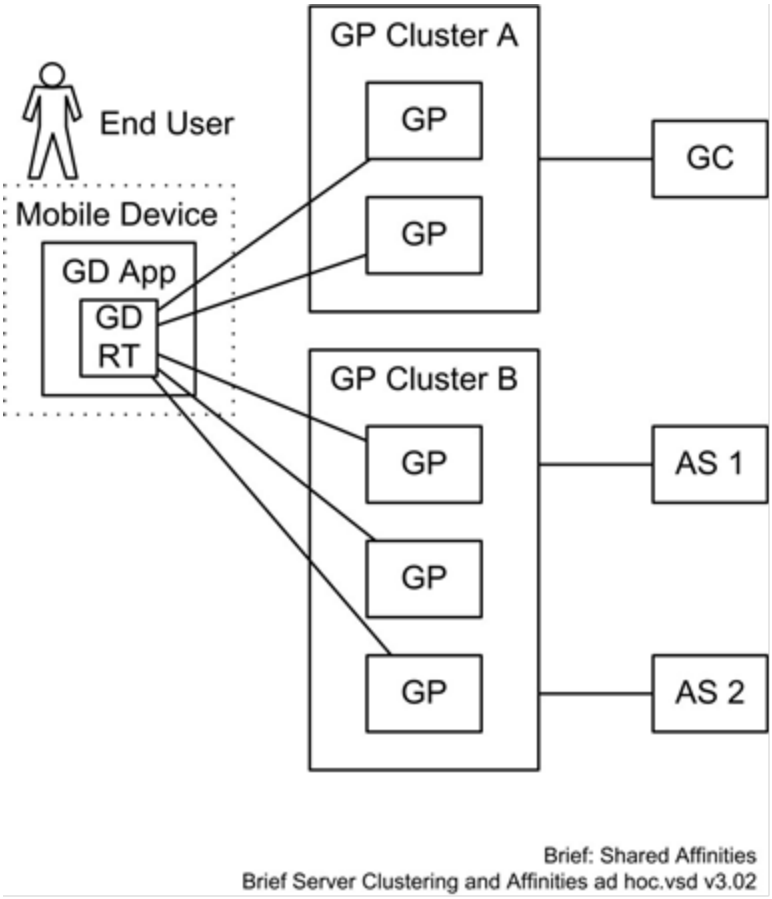


Figure 11: Shared Affinities

Affinity Priorities

A host server can have affinities with more than one proxy cluster. When a host server has multiple affinities, each can be assigned a priority order. The affinity priority order is followed when a connection to the host server is being made.

Compared to the simple affinities configuration, connection to a host server for which affinity priorities have been configured involves an additional selection step.

The GD runtime will first select the host server's highest priority affinity. The GD runtime then attempts to connect to one of the GPs in the proxy cluster with which the selected affinity is linked. As in the simple affinities configuration, the GD runtime attempts to connect to all the GPs in the proxy cluster, at random, until it succeeds or until there are no untried GPs in the cluster.

If the GD runtime cannot connect to any of the GPs in the first selected proxy cluster, the GD runtime selects the next highest priority affinity of the host server. The GD runtime then attempts to connect to one of the GPs in the proxy cluster with which that affinity is linked, as above for the first selected affinity.

If the GD runtime cannot connect to any of the GPs in any of the proxy clusters with which the host server has an affinity, then a connection cannot be made to the required host.

Note: Affinity priorities apply in the context of the host server, not the proxy cluster. The same proxy cluster could have primary affinity priority for one host server, but secondary for another. This is different to the priorities described in the server clustering section, which apply to the host servers themselves.

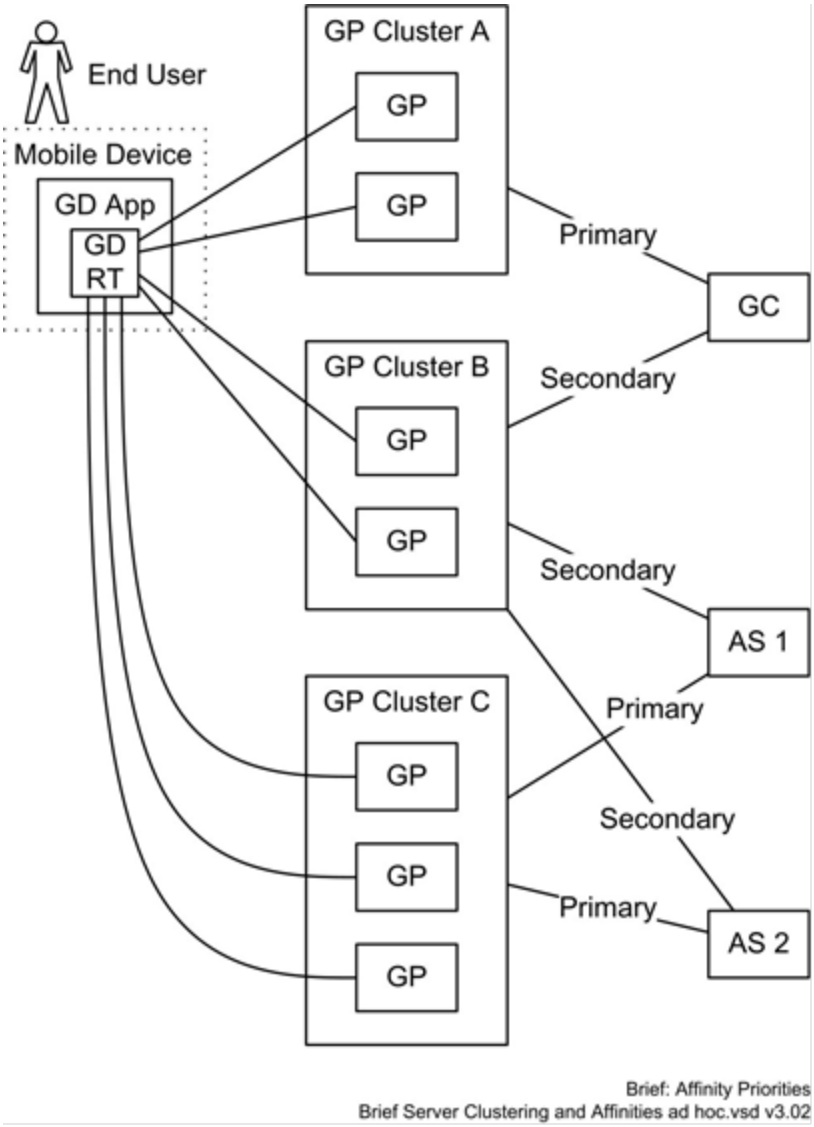


Figure 12: Affinity Priorities

The Affinity Priorities figure shows:

- GP Cluster A has primary affinity priority for the GC. GP Cluster B has secondary affinity priority for the GC. GP Cluster C has no affinity with the GC.
- AS 1 and AS 2 share GP Cluster C as their primary priority affinity, then share GP Cluster B as their secondary.
- During normal operation, when all GPs are running, the GD runtime will use:
  - GP Cluster A to connect to the GC.
  - GP Cluster C to connect to AS 1 and AS 2.
- Outside normal operation, when one or more servers are off line, the GD runtime could make connections to any of the GP servers in any cluster.

- The GC has some dedicated proxy server resources. Neither AS 1 nor AS 2 have individually dedicated proxy server resources.
- Some proxy server resources are set aside for secondary use.

Affinity Priority and Server Priority

Affinity priorities are compatible with server priorities, which are described in Server Clustering. How these capabilities work in combination is described in Server Clustering and Affinities.

Affinities Limitations

A maximum of two affinities can be assigned to a host server. If a host server has two affinities, one must be assigned primary priority and the other assigned secondary priority.

In principle, it is only necessary for a GD mobile application to use a GP when connecting to a host server that is located behind the enterprise firewall. It is unnecessary for a GD mobile application to use a GP when connecting to an application server that is accessible to the Internet. However, identification of Internet application servers as distinct from enterprise application servers depends on configuration in the GC console. If an application server's address or domain is configured as an enterprise address, then a GP connection will be used, and the affinities of the application server would come into play in GP selection. This applies regardless of whether the server's address is actually accessible to the Internet.

Server Clustering and Affinities

The server clustering and affinities features have been designed to be used together. Some aspects of these features can be understood in isolation. The following topics describe the aspects, which are best understood when the features are considered in combination:

- Host Server Clustering with Server Priorities and Affinities
- Complete Processing Model
- Configuring Data Structure
- Applications Programming Interface
- Backward Compatibility
- Separate Good Dynamics Deployments
- Network Administrator Notes

Host Server Clustering with Server Priorities and Affinities

GC and application servers that have been assigned server priorities can be assigned affinities to GP clusters, and the affinities can be assigned affinity priorities. The configuration in this type of deployment is a combination of that for server priorities and affinity priorities. Each host server is configured with its own set of affinity relationships to GP clusters. Host servers with the same server priority can have different affinities, and different affinity priorities.

The communication setup in this type of deployment is also a combination of that for the features when separate. There are two levels to the setup, which work similarly to the layers in a protocol stack. The upper level is the connection to a host, either the GC or an application server. The lower level is the proxy connection, to a GP. The upper level is the reason communication is being set up; the lower level is the enabler.

Host Connection

Communication setup starts with the selection of a host server.

If the host is the GC, then server selection is made by the GD runtime. If the host is an application server, then selection must be made in the application code.

For the GC, host selection is made in order of server priority. If there is more than one GC server with the same priority, then a random selection is made between them. For an application server, host selection could follow the same algorithm, or could follow some other algorithm, whatever is implemented by the application code.

After a host server is selected, an attempt is made to set up a proxy connection for the host server. See the next section for details of proxy connection setup.

If proxy connection setup succeeds then host connection is attempted, using the proxy connection. If this also succeeds, then communication setup overall has succeeded. Otherwise, if proxy communication setup fails, or if host connection fails, then the next host server is selected.

For the GC, the next host will be an untried host at the same priority, selected at random, if there are any. Otherwise the next host will be one at a lower priority, selected at random. In either case, proxy connection-setup is again attempted. For an application server, the algorithm could be the same, or could be different, as before.

The selection of host server and attempting of proxy communication setup continues until a working host connection is made, or until there are no more untried host servers. If all host servers have been tried and none have succeeded, then overall communication setup has failed.

Note: Communication setup could fail because no proxy connections can be set up, even if one or more host servers are actually running.

Proxy Connection

In order to make a host connection, a proxy connection must first be made. Proxy connection is always attempted in the context of a specific host, which could be a GC server, or an application server. In either case, the setup of the proxy connection takes place in the GD runtime, not in the application code. Proxy connection requires connection to a GP server.

The GP is selected according to the affinities of the selected host server. The GPs in clusters linked to affinities of higher priority are tried first, at random. If none are available, then the GPs in clusters linked to affinities of lower priority are tried next, at random.

Proxy communication setup succeeds as soon as connection to a GP succeeds.

If there is no GP to which connection succeeds in any of the clusters with which the specified host server has an affinity, then proxy communication setup fails.

Figure below shows an illustrative deployment in which:

- There are three GC servers: GC 1, GC 2, GC 3. There are two DB instances, one for GC 1 and GC 2 and a mirror for GC 3. The mirroring is non-GD.
- GC 1 and GC 2 have primary server priority. GC 3 has secondary server priority.
- GP Cluster A has primary affinity priority for GC 1. GP Cluster B has secondary affinity priority for GC 1. The GC 2 and GC 3 servers only have an affinity with GP Cluster B.
- The GD runtime on the mobile device could make connections to any of the GP servers in any cluster.
- A new connection attempt would select between GC 1 and GC 2, at random. If neither server was contactable, a connection attempt to GC 3 would be attempted.
- A connection attempt to GC 1 would first attempt to connect via a GP in GP Cluster A. If no GP in GP Cluster A was available, then the GPs in GP Cluster B would be used instead.

The server clustering and affinity configuration in Figure X could be also represented as the following table.

Server	Address	Server Priority	Primary Affinity	Secondary Affinity
GC 1	gc1.intranet.example.com	Primary	GP Cluster A	GP Cluster B
GC 2	gc2.intranet.example.com	Primary	GP Cluster B	
GC 3	gc3.intranet.example.com	Secondary	GP Cluster B	

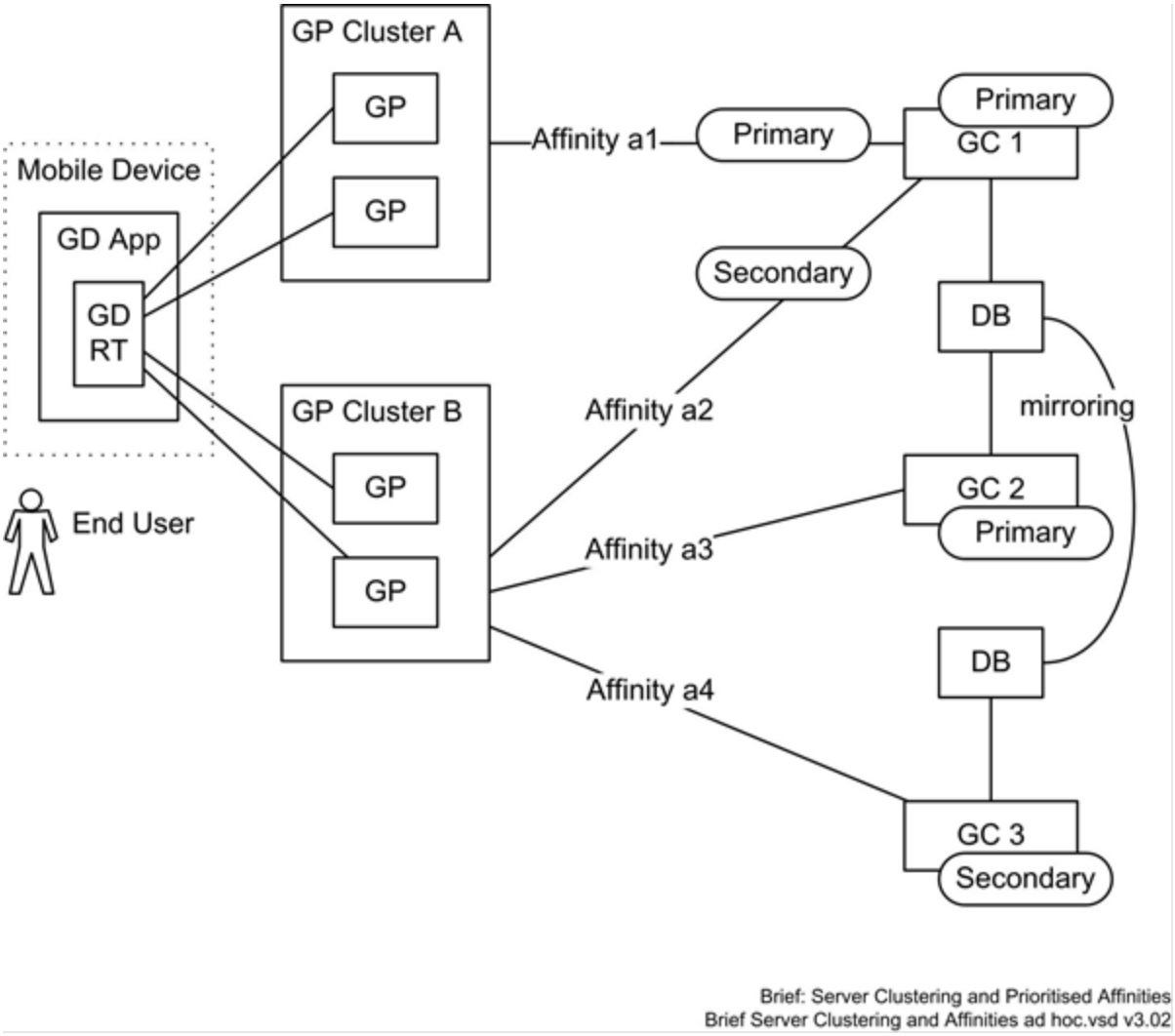


Figure 13: Server Clustering and Prioritized Affinities

See also Configuring Data Structure, noting that figure above is a denormalized representation.

Application Servers

The same capabilities that can be used for application servers can be used for GC servers as well. Application servers that have been assigned server priorities can be assigned affinities to GP clusters, and the affinities can be assigned priorities.

As previously mentioned, the GD runtime implements affinities completely for all host servers. The GD runtime also implements host server selection, but only for the GC. The application code must implement host server selection for application servers, and is provided with an API that enables this.

See Applications Programming Interface (API) for additional details.

Resilience Summary

Communication setup to a particular host server can fail because:

- The host server itself is over capacity, offline, or otherwise inaccessible.
- There are no working and reachable GP servers in any proxy cluster that has an affinity to the host server.

In any other circumstances, a working route can be established and communication set-up will succeed.

It is possible that connection to a particular host server fails because connection attempts to all GPs in all proxy clusters with an affinity to the host server have failed. In that case, it may still be possible to reach another host server that provides the same service, if the other server has an affinity with at least one different proxy cluster.

Complete Processing Model

The complete capabilities of the server clustering and affinity features are:

- Multiple GC servers
- Multiple application servers
- Server priorities
- Multiple named GP clusters
- Affinities between GC servers and GP clusters
- Affinities between application servers and GP clusters
- Affinity priorities

When all these capabilities are in use, the configuration and sequence of selections made in establishing communication will be as follows next.

Complete Processing Model: Configuration

Follow these three configuration steps.

1. Configuration of GC servers. A number of GC servers are installed, and configured with a server priority each. There are now a number of primary GC servers, and there may also be a number of secondary GC servers, and so on.
2. Configuration of GP clusters. A number of GP servers are installed, and configured into a number of proxy clusters. The proxy clusters are given names, but not assigned a priority order as such.
3. Configuration of affinities. Every GC server is assigned a primary priority affinity to a proxy cluster. Every GC server is also typically assigned a secondary priority affinity to a different proxy cluster. Note that any one proxy cluster could be the primary affinity of one GC server, and a secondary affinity of a different GC server.

This completes the configuration of the affinity and clustering topology.

Note: Later, the configuration can be changed. GP servers can be moved between clusters, new clusters can be created, new GC and GP servers can be installed, affinities can be reassigned, GC server priorities can be changed, and so forth.

Connection

A connection from a GD application to Good Control should proceed in the following sequence:

1. Selection of host server priority. The GD runtime selects the highest priority that has yet to be tried. For a new connection, this will be primary.
2. Selection of host server. The GD runtime selects an untried GC server with the selected server priority as host server. This selection is random. For a new connection, any server with primary priority could be selected. Note that this is only a logical selection and no attempt to connect is made yet.
3. Selection of proxy cluster. The GD runtime identifies the highest priority affinity whose proxy cluster has yet to be tried, and selects the proxy cluster linked to that affinity. For a new connection this will be the one with primary affinity priority.
4. Selection of GP server. The GD runtime selects an untried GP server within the selected proxy cluster. This selection is random. For a new connection, any server in the cluster could be chosen.
5. Attempt connection to proxy server. The GD runtime attempts to connect to the selected GP server:
   A. If proxy connection succeeds, then the GD runtime attempts to connect to the selected host server. See the Attempt connection to host server paragraph, below.
   B. If proxy connection fails, then the GD runtime selects a different, untried GP server in the same proxy cluster for the next attempt. If there are no more untried servers in the current selected proxy cluster:
      a. If there are any untried proxy clusters with which the selected host server has an affinity, the GD runtime selects a different cluster, returning to the Selection of proxy cluster paragraph, above.
      b. If there are no more untried proxy clusters with which the host server has an affinity, then:
         i. If there are any more untried host servers with the same server priority, the GD runtime selects a different host server, returning to the Selection of host server as described above.
         ii. If there are any untried host servers with a different server priority, the GD runtime selects a different host server priority, returning to the Selection of host server priority as described above.
         iii. If there are no more untried host servers, communication setup has failed.
6. Attempt connection to host server. Once a proxy connection has been established, the GD runtime attempts to connect to the selected host server. The host server was selected in the Selection of host server step, above. The connection attempt is made via the connected GP server as follows:
   A. If host server connection succeeds, communication setup has succeeded.
   B. If host server connection fails, then:
      a. If there are any more untried host servers with the same server priority, the GD runtime selects a different host server, returning to the Selection of host server described above.

b. If there are any untried host servers with a different server priority, the GD runtime selects a different host server priority, returning to the Selection of host server priority described above.
c. If there are no more untried host servers, then communication setup has failed.

The foregoing constitutes the complete connection setup processing model.

Notes Regarding Application Server

Application servers can be given priorities and affinities in the same way as GC servers and, in principle, the processing model for setting up a connection is identical the model conveyed above. However, the implementation of this is split between Good Dynamics itself and the application code.

Good Dynamics implements:

- Entry of application server priority and affinity configuration in the GC console.
- Making the configuration available to the application layer.
- Processing of proxy cluster selection.
- Processing of GP server selection.

The application code implements:

- Selection of host server priority.
- Selection of host server.

These final two processing steps would have to be implemented by the application developer.

The application code could implement the same algorithm as the GD runtime, i.e. selecting randomly between servers with the same server priority. The application code could also implement a variation of that algorithm, or a completely different algorithm of its own. An application that is to be migrated to the Good Dynamics platform might already have a server clustering capability and be required to remain compatible with it.

See Applications Programming Interface (API) for a description of the API.

Configuration Changes

It is possible that an enterprise's server clustering and affinities configuration is changed during a connection attempt. Such changes are handled by different components depending on the nature of the change. For instance:

- Changes to GP cluster configuration are handled within the GD runtime. This includes changes to affinity configurations, regardless of the type of host server to which the affinity applies.
- Changes to GC configuration are handled within the GD runtime.
- Changes to application server configuration, other than affinity changes, have to be handled by the application code. This includes adding or removing application server instances, or changing the priorities of existing application servers. Whatever the change, an event will be dispatched to the application layer so that an adjustment can be made.

Retry Primary

During an outage of a primary host server, a GD application may connect to a lower priority server. Unless some action is taken, the GD application could remain connected to the lower priority server even after the outage had finished.

The same is true of proxy connections. During a GP outage, a GD application may make a proxy connection to a host through an affinity with lower priority. After the outage, the GD application could remain connected through the lower priority affinity.

Ideally, neither situation should be allowed to persist and the GD application should retry a primary host server, or retry the GPs in the primary affinity, at some point.

For proxy connections, primary retry is provided by the GD runtime. When a GP connection fails, the GD runtime tracks the time of failure. After a period of time has elapsed, the failure is disregarded. The GP could then be tried again during a subsequent proxy communication setup attempt. The precise failure period can vary but is in general approximately 5 minutes.

For connections to the GC, primary retry is also provided by the GD runtime. Whenever a new connection is required, any previous individual server failures are disregarded. A new GC connection is required when a GD application starts:

- For the first time.
- After the device has been switched off.
- After having been terminated and unloaded from memory.

Mobile applications may generally be terminated and unloaded by explicit user action, or by the mobile device in order to conserve battery power or to accommodate other applications. Unloading might happen when the user takes a phone call, for example, or does not use the device for an extended period.

For connections to an application server, implementation of primary retry has to be in the application code. The simple algorithm employed for GC connection could be implemented. A variant or alternative algorithm could also be implemented, as it could for selection of an application server within a cluster.

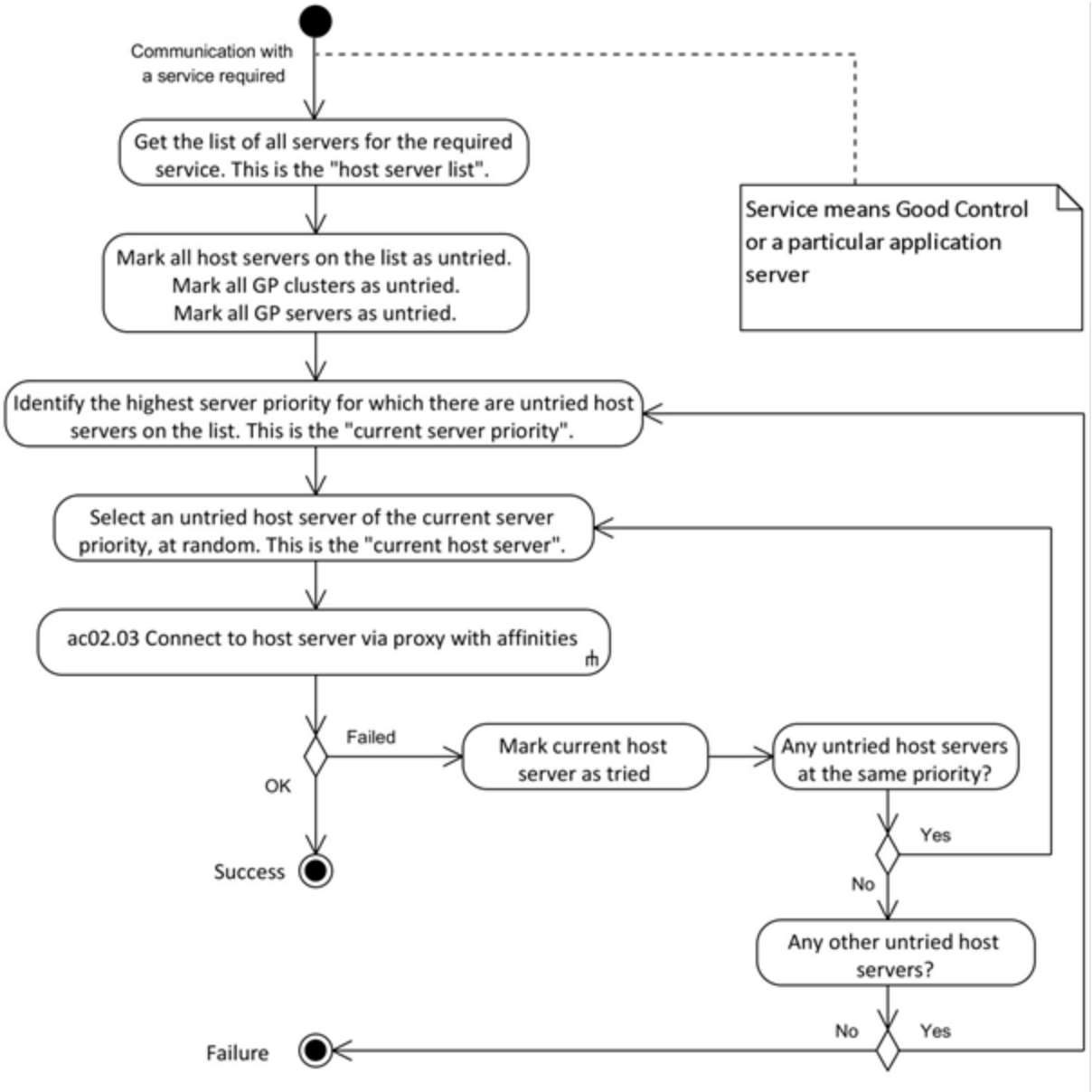
A GD application that makes only a short-lived connection to its application server implicitly implements retry primary, if the application makes a new server selection for every connection.

Note: It could be argued that a GD application should not wait for termination, and should periodically tear down and retry its connection to a host server in order to ensure that retry primary takes place. In practice this seems unnecessary because a mobile application is liable to be terminated frequently.

Unified Modeling Language (UML) Activity Diagrams

The following diagrams illustrate the processing described above as UML activity diagrams. The first diagram represents the whole activity. The second diagram is a sub-activity of the first.

For simplicity, this is assumed to be a new connection with no already-failed host servers or GP servers. Also for simplicity's sake, historical features of GPs are not shown, nor are any optimizations. The diagram likewise has no iteration zones, and shows explicit marking of tried status.



Name	ac02.02 Connect to service with clustering and affinities
Documentation	UML Activity diagram showing the algorithm for selection of host server and proxy server, with affinities. For simplicity, historical failures of GPs are not shown, not are any optimisations. This diagram has no iteration zones, and shows explicit marking of tried status.

Figure 14: Connect to Service with Clustering and Affinities

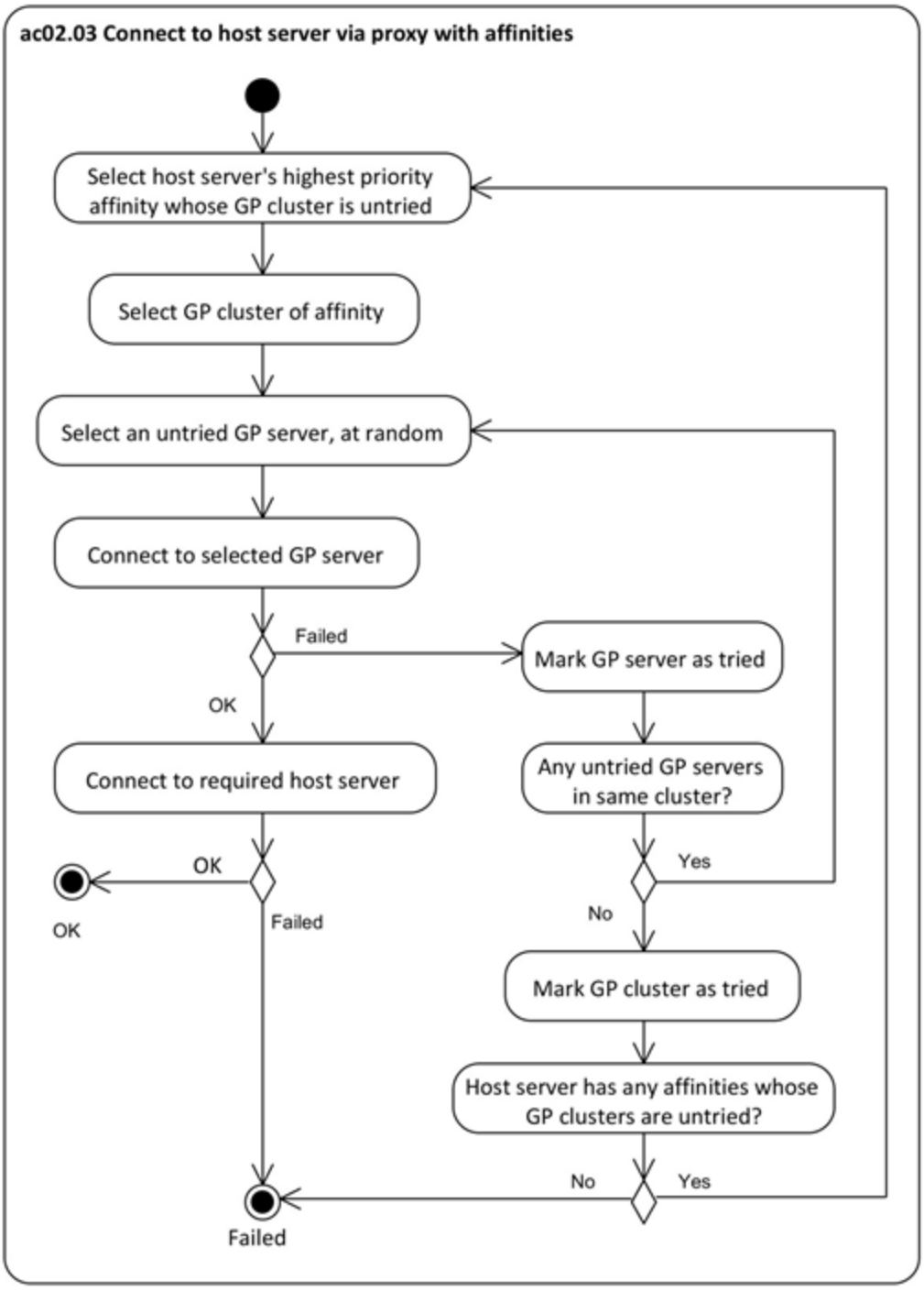


Figure 15: Connect to Host Server via Proxy with Affinities

Configuring Data Structure

The server clustering and affinities features require configuration by the deploying enterprise. Configuration is made using the Good Control console user interface and is stored in the GC server as structured data.

Note: This section describes the effective data structure for purposes of explanation. Bear in mind that the descriptions here do not necessarily show the internal representation, nor the structure as presented in the user interface.

Description

The data structure of the server clustering and affinity configuration is as follows.

1. At the top of the host server configuration is the Service, which may be Good Control, or a specific application server. The configuration for GC and application host servers is the same.
2. A Service is provided by one or more Host Server instances. Any Host Server provides exactly one Service. Every Hosted Server has an address and a port number, and a server priority. Server priority can be Primary, Secondary or Tertiary.
3. A Host Server has one or more Affinity relationships with one or more Proxy Cluster instances. An Affinity can have a priority order, expressed as primary or secondary.
4. A Proxy Cluster has zero or more Affinity relationships with Host Servers.
5. Every Proxy Cluster has a name.
6. A Proxy Cluster consists of a number of Good Proxy Server instances. Any Good Proxy server is in only one proxy cluster.

Diagram

Figure below illustrates the data structure description in Description.

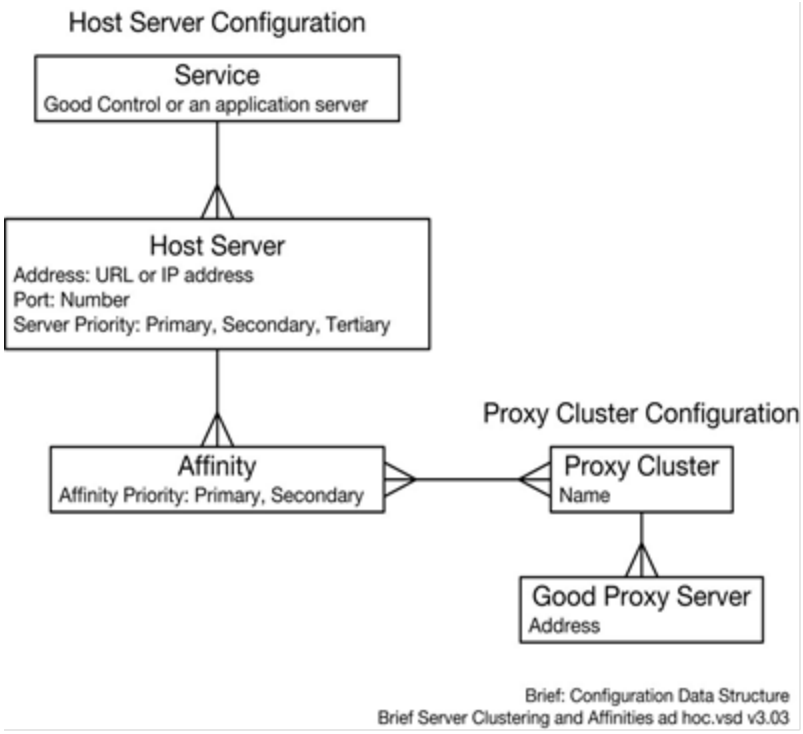


Figure 16: Configuration Data Structure

Applications Programming Interface (API)

Good Dynamics API furnishes support for server clustering and affinities in a number of ways in the following places:

- GD Runtime
- Good Proxy Services
- Good Control Web Services

GD Runtime

A GD application that communicates with a clustered application server should implement a host server selection algorithm that includes selecting in order of server priority. See the Application Server Notes for details.

The GD runtime offers an API to enable GD applications to implement selection of host server and host server priority. The API provides a representation of the server clustering configuration as entered in the GC console. The API is included in the **getApplicationConfig** function. See the references in Recommended Reading for details of the API.

Note: There is no GD API for any affinities capability—proxy cluster selection or any lower level processing. Attempting connection to a server address using GD secure communication implicitly initiates proxy cluster selection and GP server selection based on the configured affinities of the host server. These selections are made automatically by the GD runtime.

Good Proxy (GP) Services

The GPs at an enterprise act as access points for a number of services that can be used by application servers. For example, the Push Channel server-side API can be accessed in this way.

Every GP server at an enterprise provides access to all services, which means that consumers of these services can gain the benefits of GP clustering. Consumers should implement a server selection algorithm, in order to achieve this.

Ideally, a consumer of GP services would follow a similar server selection algorithm to that used by the GD runtime for GP selection (see Affinity Priorities). A service consumer should only communicate with GPs in proxy clusters with which it has an affinity.

The GD infrastructure offers a server-side API to enable application servers to implement selection of proxy cluster and GP server. The API provides a representation of the GP clustering and affinity configuration as entered in the GC console.

The API is provided as a specific HTTP service: **getGPServers**. The response to the service includes a list of GP server addresses and affinity priorities.

The API is accessible through the enterprise GP servers themselves, which means that a "bootstrap" approach should be taken. The address of a first GP server could be provided when the application server is installed. The application server can then access the API on the first GP server, which in turn provides a list of GP servers for use going forwards. The list should be saved by the application server, and updated periodically.

See the references in Recommended Reading for details of the API.

Good Control Web Services

A GC web services API is hosted by the Good Control server. Most actions that can be taken in the GC console user interface have an equivalent in the GC web services API. Clients of the GC web services API are typically system applications that are located behind the enterprise firewall.

When clustering is in use, every GC server functions as a host for GC web services, which means that clients of the web services API can gain the benefits of GC clustering. Clients should implement a server selection algorithm to achieve this.

Ideally, a client of GC web services would follow the same server selection algorithm that is used by the GD runtime for GC host server selection. (see Basic Clustering: Application Communication).

An API that enables host server selection within a cluster, according to server priorities, is made available as part of the **GetAppInfoRequest** service, which can be used to obtain a list of GCs and their addresses and server priorities.

Note: The API is provided as a GC web service, so a "bootstrap" approach should be taken. The address of a first GC server could be provided when the client application is installed. The client can then access the API on the first GC server, which in turn provides a list of GC servers for use going forward.

Backward Compatibility

GD software components that can use server clustering and affinities are backward compatible with GD software from earlier releases that did not have those features.

- Mobile Application
- Good Control and Good Proxy Servers
- GD Runtime

Mobile Application

Backward compatibility with the mobile application on the device is required.

When these features are in use, the whole infrastructure configuration is sent to, and stored by, the GD runtime on the device. Early versions of the GD runtime are not compatible with server clustering or affinities and would be unable to store or use the configuration.

When the GC detects that a connecting GD runtime is unable to store the whole configuration, it attempts to render a stripped-down version of the configuration that can be stored.

The stripped down configuration consists of only single server addresses for the GC and application server, and a single list of GP server addresses.

Good Control and Good Proxy Servers

A mobile GD application that supports server clustering is backward compatible with a deployment of earlier GC and GP servers that do not support it.

The mobile application will behave as though the default configuration is in effect (see Network Administrator Notes). In this case, there will appear to be a single server with primary priority for the GC, as well as the same for any application server.

GD Runtime

A GD application that does not support application server clustering, i.e. one that does not implement any application server selection algorithm, can be built with a version of the GD runtime that does support server clustering. In this case, backward compatibility is provided in the API.

The pre-server clustering API provides only a single address and port number for the application server. This API is still offered by GD runtime instances that support server clustering. See Recommended Reading for details.

Separate Good Dynamics Deployments

In all of the above descriptions, the enterprise will have chosen to install at least one additional GC in a cluster with an existing GC. It is also possible for an enterprise to install an additional GC as a separate deployment. Separate deployments are different than clustered deployments; the principal differences being:

- The GCs in separate deployments do not share the same GC DB.
- GP server resources cannot be shared. Any GP server can only be part of one deployment.
- GC server resources cannot be shared between separate deployments.
- There is a permanent association between any GD application and the deployment against which it was activated.
- Affinity relationships cannot be created between the host servers in one deployment and the proxy clusters in another.

Because resources are not shared between separate deployments, server outages in one deployment cannot be made up by servers in another deployment. A separate deployment can itself use server clustering, but the clustered resources are only available within that deployment.

Network Administrator Notes

The notes that follow are intended to assist with network administration of a GD deployment that includes server clustering and affinities:

- Default Configuration
- Explicit Configuration

- Communication Between Servers
- Tip: Water Butt Metaphor for Shared Server Priority
- Advice for Commissioning and Decommissioning Servers

Default Configuration

In principle, the use of server clustering and affinities is optional. In practice a default configuration of both of these features must be in place at the time of initial deployment. Implicitly, such a default configuration would include:

- All GC servers have primary server priority.
- All GP servers are in a single cluster.
- Every GC server has an affinity to the GP cluster.
- Every application server has an affinity to the GP cluster.

This means that any GP can be the proxy for a connection to any GC or application server.

Explicit Configuration

If there is any cluster or affinity configuration beyond the default, it is probably best to make all GPs clustered, as well as to give all host servers an explicit affinity to one or more proxy clusters.

In practice, this means:

- If any GP servers have been clustered and dedicated to particular host servers, put all other GP servers in a single undedicated cluster.
- For each host server that does not have an affinity to a proxy cluster, add an affinity to the undedicated GP cluster.

The result of which makes the whole configuration explicit and therefore comprehensible.

Communication Between Servers

Regardless of the clustering and affinity configuration that is in place, every GP in every proxy cluster needs to be able to receive infrastructure configuration data from a GC server. See Basic Clustering: Installation and System Administration under Server Clustering, for background.

Generally, GD enterprise infrastructure configuration works as follows:

1. Configuration takes place in the GC console user interface. The console of any of an enterprise's GC servers can be used for this.
2. Whichever GC was used updates the infrastructure configuration, which is stored in the GC DB. The GC DB is shared by all the GC servers at an enterprise.
3. Every GC server at the enterprise will attempt to send the changed configuration to every GP server.

This may have implications for network topology beyond what is specific to GD.

Tip: Water Butt Metaphor for Shared Server Priority

The difference between a cluster of servers and multiple clusters may be explained by means of the "water butt" metaphor given next. The same metaphor can be used to explain the difference between servers with the same priority, and servers with different priorities.

A cluster of servers, or servers with the same priority, can be compared to a group of water butts or rain barrels arranged so that:

1. Rain water drains directly into one of the water butts.
2. The other water butts are joined by pipes fitted at the bases of the butts.

By contrast, multiple clusters, or servers with different priorities can be compared to a group of water butts arranged similarly but with the joining pipes at the top of the butts.

The butts joined by pipes at their bases will fill at the same time. The butts joined by pipes at their tops will fill one after the other as shown in figure below.

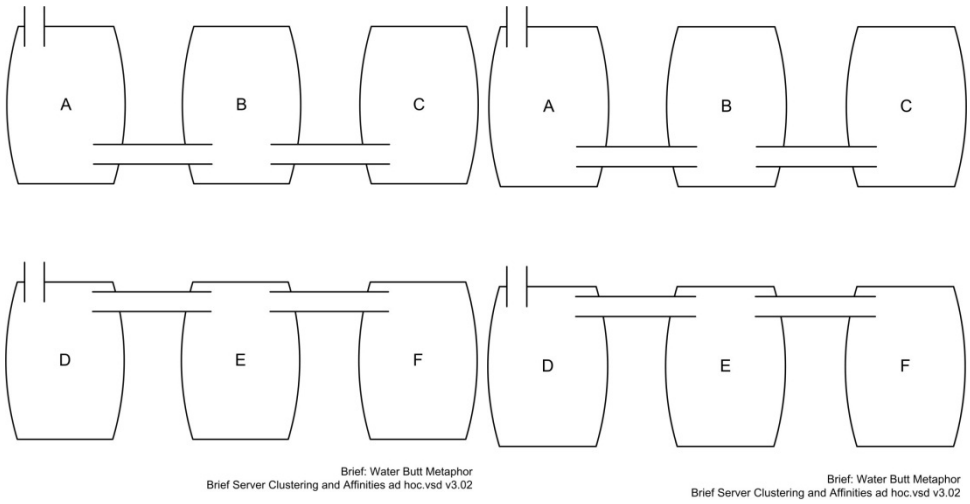


Figure 17: Clustering and Water Butts

Butts A, B and C are joined at the bottom and fill at the same time. Butts D, E and F are joined at the top, so butt D fills first, then butt E fills, then butt F fills.

Butts A, B and C are like servers in the same cluster, or servers with the same priority. Their processing loads increase at the same time. Butts D, E and F are more like multiple clusters, or servers with descending priority. One is loaded to maximum capacity before the next takes any load at all.

Advice for Commissioning and Decommissioning Servers

The following advice is given for commissioning servers to run GCs, GPs, and application servers deployed with server clustering and affinities.

All the servers for the GPs in a single proxy cluster should have the same processing capacity. This is because a server will be selected at random from the cluster. The same is true for the servers that are used for GCs or application servers at the same server priority.

The GP is required for connection between GD applications and their application servers. If all GPs are unavailable, no mobile users can connect to their enterprise data. The GC is not required for application connections. Existing end users can continue to access their data, for a time, even if all GCs are unavailable. This means:

- Typical deployments include more GPs than GCs.
- When upgrading GP hardware, it is advisable to add new GPs online before decommissioning old GPs.

The recommended procedure for decommissioning a GP server is to first change the configuration so that the GP is in a proxy cluster that has no affinities. The GP will then shed any traffic and can be switched off safely.

About Direct Connect

Direct Connect is a deployment option in which your client devices bypass the standard GD NOC and connect directly to the GC and GP servers, typically through a load balancer. Direct Connect is often used if network latency is a concern.

The clustering and affinities described here apply whether you are using Direct Connect or not. That is to say, you have Direct Connet is yet another networking option, which can be combined with clustering or affinities.

For more information about Direct Connect, see the *GD Direct Connect* guide.

Recommended Reading

Detailed assistance in setting up server clustering and affinities can be accessed from within the configuration user interface. Details of the APIs that should be used with server clustering and affinities are in the technical documentation found in the Resource Library on the GDN website.

Configuration UI and Online Help

The Good Control console is the UI for configuring server clusters and affinities. Online help is also available in the console.

Configuration of the GD enterprise server clustering and affinities is accessed from the GC console. Online Help is accessible in the top-right of the display.

Configuration of application server clustering and affinities is accessed on the application management screen. Select **Manage Applications** and then the application whose application server clusters are to be configured.

Technical Documentation

Technical documentation for developers is included in the API Reference on the Good Developers Network website, as follows:

- Mobile Applications
- Application Servers

Mobile Applications

GD mobile applications can obtain details of their application server's cluster and priority configuration. The details are included in the **getApplicationConfig** collection, in the **GDAppConfigKeyServers** value.

See the function reference in the GDiOS class documentation if using the GD SDK for iOS. If using the GD SDK for Android, see the GDAndroid class documentation.

See also the note on *Application Server Selection* after the table of **getApplicationConfig** keys.

Application Servers

The GP servers at an enterprise act as access points for a number of services that can be used by application servers. For example, the Push Channel server-side API can be accessed in this way. Every GP server at an enterprise provides access to every GP service.

Application servers can retrieve details of the GP cluster and priority configuration. This enables application servers to gain the benefits of GP clustering.

See the Good Proxy Server List API in the appendix of either API Reference:

- iOS
- Android

Appendix: Example Topologies

The following examples illustrate some types of network topology that include server clustering and affinities. The normal operation of each topology is discussed, as well as their operations in some outage scenarios.

Good Control Clustered Topology

Figure below illustrates an example deployment for the purposes of discussion and explanation.

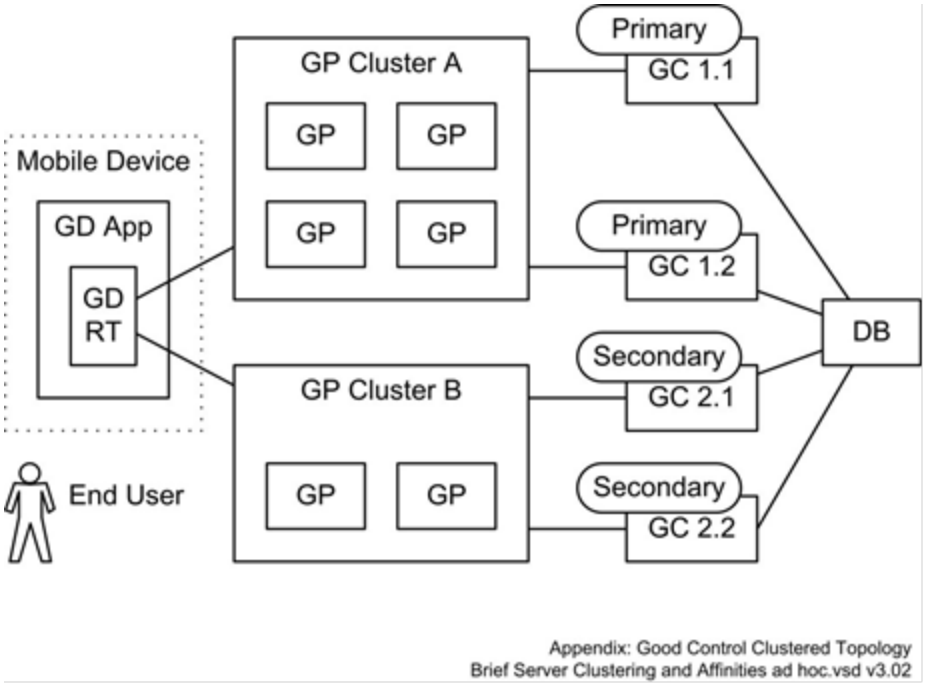


Figure 18: Good Control Clustered Topology

The diagram shows an example deployment of a GC with clustering and affinities.

Four servers are deployed for the GC service. Two servers are assigned primary server priority; the other two are assigned secondary server priority. All GC servers share the same database instance. There are six GP servers, deployed in two proxy clusters. One GP cluster has four servers; the other has two servers.

The primary GC servers both have an affinity with the larger proxy cluster, GP Cluster A. The secondary GC servers both have an affinity with the smaller proxy cluster, GP Cluster B.

This configuration is represented in the following table.

Server	Server Priority	Primary Affinity
GC 1.1	Primary	GP Cluster A
GC 1.2	Primary	GP Cluster A
GC 2.1	Secondary	GP Cluster B
GC 2.2	Secondary	GP Cluster B

Normal Operation

The normal operation of the GD runtime in the deployment shown in figure Good Control Clustered Topology can be stated as follows.

The GD runtime connects to either primary GC server at random. Connections are made through one of the four servers in GP Cluster A, also selected at random.

The GC servers with secondary priority, and the servers in GP Cluster B, are not used in normal operation.

Outage Operation

The operation of the application in the deployment during various outages is as follows.

In the scenario that one primary GC server is down, all GD runtime instances will connect to the other primary GC server. Note that a GC can refuse connections when overloaded, and so the other primary GC server could start to appear offline to new connections.

In the scenario that all primary GC servers are down, or appear to be, the GD runtime will connect to a secondary GC server, selected at random. The connection will go via one of the servers in GP Cluster B, also selected at random.

In the scenario that every GP in GP Cluster A is down, the GD runtime will connect to a secondary GC server, selected at random, as it would if all primary GC servers were down. Although the servers in the primary cluster are running OK, there is no route to them.

In the scenario that one server in GP Cluster A is down, connections that were using that GP as proxy will fail, one by one, and reconnect through another GP in the same cluster. New connections will also be made through other GPs in the same cluster. Any GD App instance that has attempted to connect through the GP that is down will mark it as failed. After a time has passed, the failed mark will expire. The GP could then be tried again, if it is randomly selected for a new connection. New connections will be attempted by GD App instances as necessary. See Retry Primary for more details.

International Deployment with Sub-Clusters and Proximity

Figure below illustrates an example deployment for the purposes of discussion and explanation.

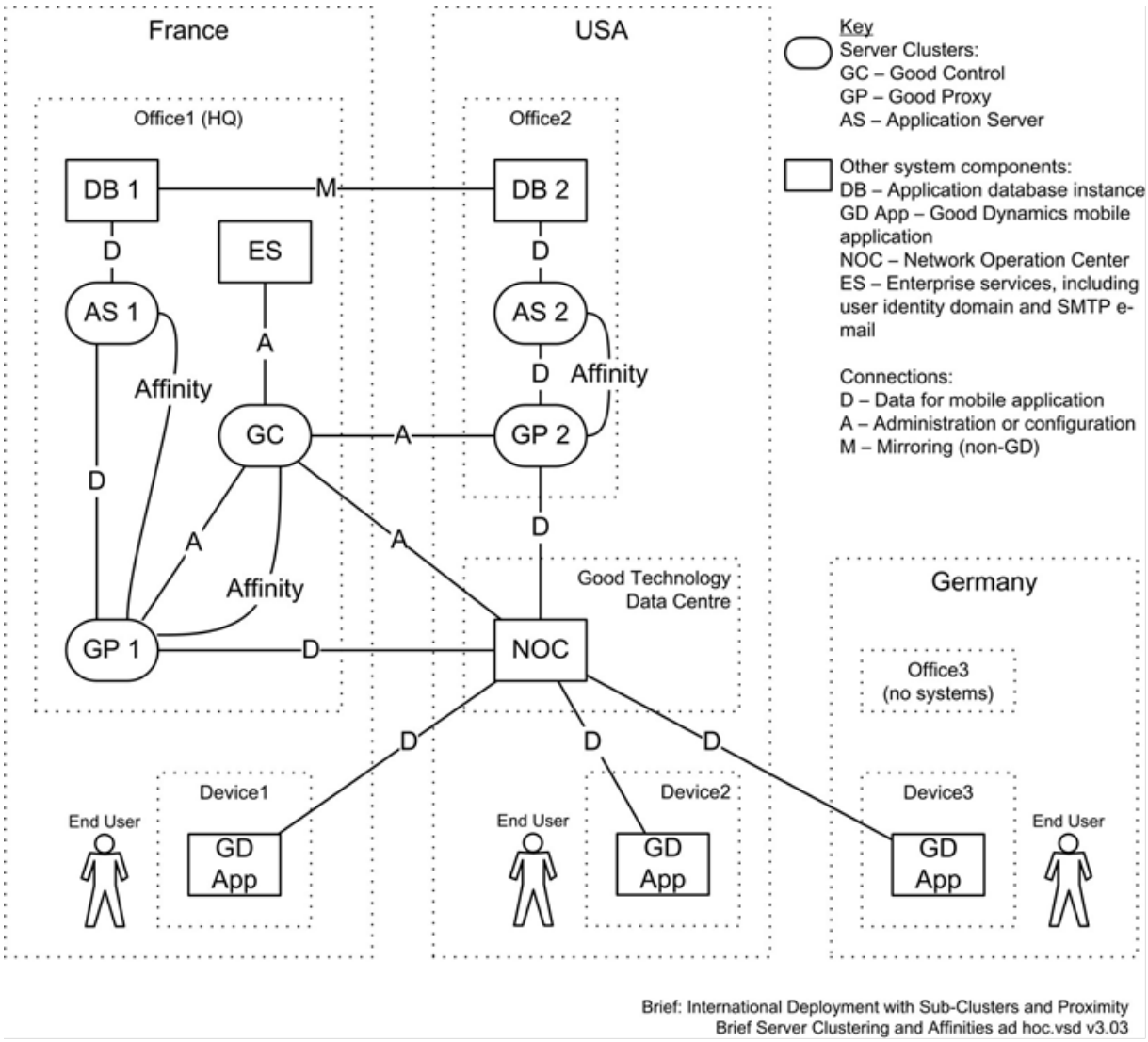


Figure 19: International Deployment with Sub-Clusters and Proximity

The diagram above shows an example deployment of a single GD application.

The application is deployed at an international company with offices in three countries: France, USA and Germany. Two of the offices have server rooms. In each server room there is a GP cluster, an application server (AS) cluster, and an instance of the application's database (DB). The database instance in one office is a mirror of the other. In one server room is a GC cluster, connected to a number of enterprise services (ES) including the Microsoft Active Directory (AD) server. There are end users in all three countries, each of whom has a mobile device running a Good Dynamics App (GD App).

All application servers have the same server priority. Every GC and application server has an affinity with the proxy cluster that is in the same location. All GC servers have an affinity with the GP 1 proxy cluster. The servers in the AS 1 cluster also have an affinity with the GP 1 proxy cluster. The servers in the AS 2 cluster have an affinity with the GP 2 cluster.

Note: The clusters of host servers in the above are not as such configured in GD. Each individual host server is configured with priority and affinities. The clusters are for convenience of description.

This configuration delivers the following benefits:

- Faster data communication between host servers and proxy servers, based on proximity.
- Resilience to failure of single servers, by using clusters.
- Resilience to failure of clusters, or even sites, by using multiple clusters at different sites. See Resilience to Cluster Failure.

Note that there would be no special benefit from associating end users in a particular country with application servers in the same country. Mobile application data traffic with servers that are behind the enterprise firewall is all routed through the Network Operation Center, which is in the USA.

For additional information on using GD Direct Connect to securely bypass the NOC, see the Direct Connect Feature Summary and Configuration Guide available from GDN.

Configuration

The configuration in the preceding deployment can be represented as the following table.

Server	Server Priority	Primary Affinity
In the GC cluster	Primary	GP 1 cluster
In the AS 1 cluster	Primary	GP 1 cluster
In the AS 2 cluster	Primary	GP 2 cluster

The clusters are not as such configured. A number of instances of the same server that have the same priority and affinities forms a de facto cluster.

Normal Operation

The normal operation of the application in the deployment is as follows.

End user devices in every country connect to the NOC in the USA. From there, each device connects to an application server selected at random by the application. The processing load is split equally between all servers internationally.

Connection to an application server is always proxied by a GP in the cluster that is in the same server room.

User and other administration takes place in Office1, where the GC is located. Any network configuration is provisioned from there to the GPs in all locations.

Resilience to Cluster Failure

The operation of the application in the deployment during a cluster outage is as follows.

In the scenario that every server in the AS 1 cluster is down or off line, the following takes place:

1. Instances of the GD App being run by end users detect that the AS 1 servers with which they have current connections are not responding.
2. One by one, the GD App instances attempt to connect to other servers in the AS 1 cluster, or in the AS 2 cluster at random. (All clusters have equal priority.)
3. Connection attempts to AS 1 fail, but connection attempts to AS 2 succeed.
4. Eventually, all instances connect to an AS 2 server.

Depending on the details of the AS 1 outage, some data may have been lost and never committed to the DB 1 instance, and hence not mirrored to the DB 2 instance.

When the AS 1 cluster comes back up, assuming the GD App implements retrying of primary servers, instances of the GD App will move their connections to servers in the AS 1 cluster, one by one. See Retry Primary for details.

In the scenario that every server in the GP 1 cluster is down or off line, the same steps take place. In effect, an outage of the proxy cluster has all the impacts of an outage of the application cluster. Compare with the International Deployment with Flexible Proximity example topology.

When the GP 1 cluster comes back up, assuming the GD App implements retrying of primary servers, instances of the GD App will move their connections to servers in the AS 1 cluster, one by one. The moved connections will use servers in the GP 1 cluster as proxies. Note that there is no specific way to reconnect only the proxy level of the connection. The GD App has to disconnect and reconnect at the host server level.

In the scenario that all servers in both the AS 1 and GP 1 clusters are down or off line, the same steps take place. In this scenario, switching to AS 2 may be quicker. In the previous scenario, connections would be being made to GP servers, only to fail when attempting to connect onward to an AS 1 server. In the scenario that the GP cluster is also down, these proxy connections would fail first. It may be possible for the system administrator to take the GP 1 cluster off line when they discover that the AS 1 cluster is off line, or to configure their network to do this.

In the scenario that every server in the GC cluster is down or off-line, there is no immediate impact on the application because GC is not in the data path. Current connections remain in place and new connections can be established. However, no new end users could be added or activated, nor could existing users be blocked. After a configurable period, GP servers in all clusters will stop making new proxy connections. This is a security feature of Good Dynamics.

Resilience to Site Failure

The operation of the application in the deployment shown International Deployment with Sub-Clusters and Proximity during a site outage is as follows:

In the scenario that every server in Office2 is down or off line, a similar sequence of events takes place to the scenario that the AS 1 and GP 1 clusters are down. The difference is that the GD App instances will gradually shift to the AS 1 cluster, not the AS 2 cluster, which is down.

In the scenario that every server in Office1 is down or off-line, GD App instances will shift to the AS 2 cluster as they would in the scenario that the AS 1 and GP 1 clusters are down. There would be no other immediate impact to the application, but see the GC cluster is down scenario the results of which would apply.

International Deployment with Flexible Proximity

Figure below illustrates an example deployment for the purposes of discussion and explanation.

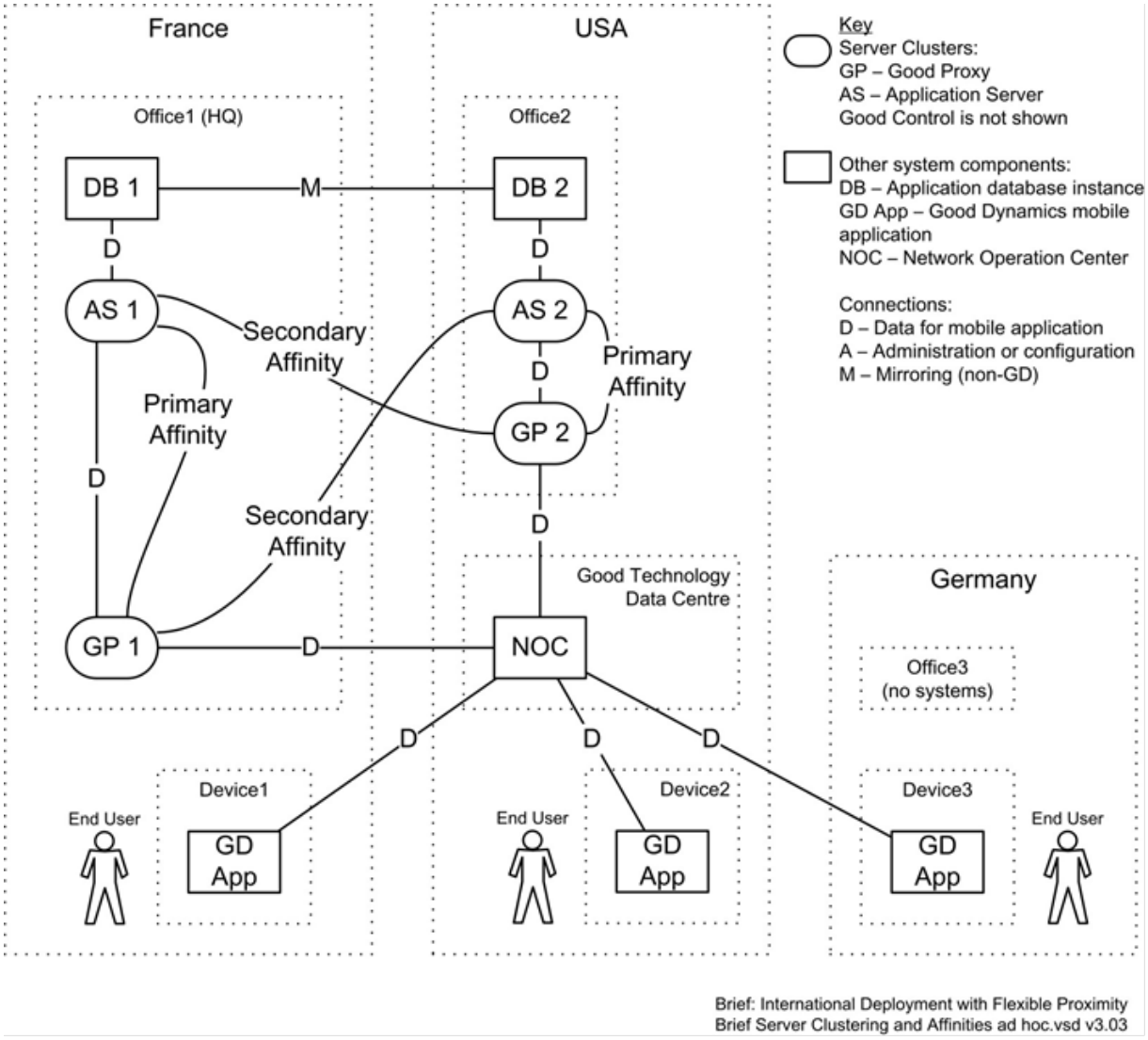


Figure 20: International Deployment with Flexible Proximity

The diagram shows some additional configuration of the deployment shown in the International Deployment with Sub-Clusters and Proximity example. The additions are as follows.

Every application server has a primary affinity with the proxy cluster in the same location. Every application server also has a secondary affinity with the proxy cluster in the other location.

This additional configuration leads to different behavior in a particular outage.

In the scenario that the GP 1 cluster is down, the GP 2 cluster continues to service connections to servers in the AS 1 cluster. It would be expected that these connections are slower, since the host servers are in a different location, but this may be preferable to the host servers being effectively off-line when their nearest proxy cluster is off-line.

Note: In the scenario that the AS 1 cluster is down, the GP 1 cluster does not transfer connections to the AS 2 cluster. Transference effectively takes place in the GD runtime on the device, not in the infrastructure. The configuration in the deployment above is represented in the following table.

Server	Server Priority	Primary Affinity	Secondary Affinity
In the GC cluster (not shown)	Primary	GP 1 cluster	GP 2 cluster
In the AS 1 cluster	Primary	GP 1 cluster	GP 2 cluster
In the AS 2 cluster	Primary	GP 2 cluster	GP 1 cluster

The clusters are not as such configured. A number of instances of the same server that have the same priority and affinities forms a de facto cluster.

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

Category	Title	Description
	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 • 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*	Working with the GD SDK for Android and the essential

Category	Title	Description
	• *API Reference for Android*	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*

Revision History

Clustering and Affinities

Date	Description
2015-07-15	Version numbers updated for latest release; no content changes.
2015-05-18	Version numbers updated for latest release; no content changes.
2015-04-30	Version numbers updated for latest release; no content changes.
2015-03-23	Version numbers updated for latest release; no content changes.
2015-01-15	Updated for latest release
2014-12-16	Styling and page layout updated
2014-12-15	Added statement of support for multi-realm Kerberos Constrained Delegation
2014-10-07	Removed superfluous screenshot of GC console
2014-09-25	• Updated for latest versions of GC and GP • Version stamp on cover page

Date	Description
2014-08-21	Added standard list of all Good Dynamics documentation
2014-08-11	Start of revision history

Good Dynamics™

Kerberos Constrained Delegation

Kerberos Constrained Delegation

Last updated: July 22, 2015

Versions GP 2.0.*xx.yy*, GC 2.0.*xx.yy*

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

© Copyright 2015. All rights reserved. All use is subject to license terms posted at www.good.com/legal. GOOD, GOOD TECHNOLOGY, the GOOD logo, GOOD FOR ENTERPRISE, GOOD FOR GOVERNMENT, GOOD FOR YOU, GOOD APPCENTRAL, GOOD DYNAMICS, SECURED BY GOOD, GOOD MOBILE MANAGER, GOOD CONNECT, GOOD SHARE, GOOD TRUST, GOOD VAULT, and GOOD DYNAMICS APPKINETICS are trademarks of Good Technology Corporation and its related entities. All third-party technology products are protected by issued and pending U.S. and foreign patents.

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

Table of Contents

Revision History

Kerberos Constrained Delegation

Date	Description
2015-08-19	Version number updated for latest release; no content changes.
2015-07-22	Corrected lack of clarity in Configuring Kerberos-related Properties in Good Control: the property `gc.krb5.enabled` must always be set true for KCD.
2015-07-09	Clarified necessities for SQL Database Account for Multi-Realm KCD Configuration: not db_ owner rights, but the actual owner of the database.
2015-05-19	Added a discussion of DNS setup needed when the GP is installed in a domain separate from the GC. See Domains, Realms, Forests, Trusts, and GC and GP: Topologies.
2015-05-18	Version numbers updated for latest release; no content changes.
2015-05-11	Removed erroneous statement that the GC and GP must be installed in the same domain. For normal Kerberos this is a requirement, but for KCD it is not needed.
2015-03-23	Version number updated for latest release; no content changes.
2015-03-03	In Concepts and Steps for a Typical KCD Single Realm Installation , for the setspn command, clarified that some versions of Windows Server might need the -s option instead of the -a option.
2015-01-23	Clarified the need for Good Control servers in resource realms. See Domains, Realms, Forests, Trusts, and GC and GP: Topologies.
2015-01-15	Updated for latest release: support for multiple Kerberos realms and forests.
2014-12-16	Styling and page layout updated
2014-11-04	Reorganized for clarity/logic
2014-10-13	Clarified the meaning of *target*: the host name or user name to be protected by KCD.
2014-10-07	Removed unnecessary, outdated screenshot of GC properties screen.
2014-09-25	• Updated for latest versions of GC and GP • Version stamp on cover page
2014-08-21	Added standard list of all Good Dynamics documentation
2014-08-13	Interaction diagram redrawn. "Foo" as name for protected server eliminated; this is the target host, the host to be protected by Kerberos.

Date	Description
2014-08-12	Added note: If you have clustered your GCs, you must must run the setspn command for and copy the keytab file to every machine in the cluster.
2014-08-11	Start of revision history

How Kerberos Constrained Delegation Works

With Kerberos Constrained Delegation (KCD) end users can access enterprise resources without having to enter their network credentials. KCD uses *service tickets* that are encrypted and decrypted by keys that do not contain the user's credentials.

Part of KCD is a mechanism called *delegation*. When this mechanism is configured, the application delegates authentication to Good Control (GC) to act on its behalf to request access to an enterprise resource.

Another mechanism is the ability to *constrain* the accessed resources. With this mechanism administrators can limit the network resources that are accessible. This is accomplished by configuring the account under which the delegate (the GC) run as trusted only for specific services.

Intended Audience and Skills

This guide assumes that you are familiar and comfortable with Good Control, its database, Microsoft Active Directory, networking, and Kerberos.

Related Kerberos Tools

Your Kerberos environment setup must include the following components:

- **Active Directory (AD) Server** – the directory service that authenticates and authorizes all users and computers associated with your Windows network.
- **Kerberos Key Distribution Center (KDC)** – the authentication service on the AD server that supplies session tickets and keys to users and computers in the Active Directory domain.
- **AD Support Tools** – additional tools that are used to configure, manage, and debug AD.

How To Use This Guide

This document details:

1. The concepts and steps for configuring KCD with a single Kerberos realm: Concepts and Steps for a Typical KCD Single Realm Installation .
2. The concepts and steps for configuring GC and KCD for multiple realms: Concepts and Steps for KCD Multi-Realm Configuration.

Always start with part 1. A basic understanding of configuring KCD wit a single realm is essential before configuring multiple realms. The steps in Part 2 rely on your understanding the steps in part 1.

Video Tutorials

There are helpful video tutorials about configuring single-ream and multi-realm KCD:

- Single-realm: https://community.good.com/videos/1443
- Multi-realm: https://community.good.com/videos/1474

The videos also include other helpful, supporting materials.

About GD Clustering

Clustering of GD components fully supports KCD. However, be sure you configure all the servers in your cluster for KCD. This is specifically called for in the commands documented in Concepts and Steps for a Typical KCD Single Realm Installation and Concepts and Steps for KCD Multi-Realm Configuration.

Terminology and Equivalences

Terminology in Kerberos is notoriously obscure and difficult to understand. Here is some of the Kerberos terminology is used in this guide.

- *User impersonation* is the Kerberos term for the configurations discussed in this document. Because the user never presents any credentials in this configuration, but instead relies on the Active Directory and Kerberos systems for authentication, the user is technically "impersonated." (User impersonation is technically distinct from Kerberos resource delegation.)
- *REALM* is the Kerberos term for a collection of "entities," either user realms or resource realms, which are any realm other than a user realm. When entered on Kerberos command lines, the REALM name must always be in uppercase.
- *Domain* in this context means "directory service domain", most frequently from Active Directory.

The terms *realm* and *domain* are conceptually equivalent in KCD. The relationship is as follows:

1 Kerberos realm : *n* Good Control server : *n* Good Proxy

There must be a minimum of one Good Control server for each Kerberos realm. The GC must still reside in the same Kerberos realm as the resource because cross-realm resource delegation is not supported. In addition, you must ensure that all Good Proxy servers and clusters can communicate with all other GC and GP servers that you intend to configuration for multi-realm KCD. Other requirements for multi-realm and forest topologies are detailed below.

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.

Relationship to Cloud GC: Feature Not Applicable

The feature, service, server type, or software described in this guide is not available on Good Control Cloud because it is not applicable in a hosted environment.

KCD and the GD SDK

Except as noted in this section, no additional work with the GD SDK is required by developers to operate with KCD. The GD Runtime itself handles the necessary interactions between a GD-based application and KCD.

iOS Apps with KCD: Use NSURL System, Not GDHTTPRequest

If you intend to develop iOS applications for use with a KCD-enabled GD, keep in mind that the GDHTTPRequest method does not support KCD. Instead, use the Apple-supplied NSURL system to make your requests.

Domains, Realms, Forests, Trusts, and GC and GP: Topologies

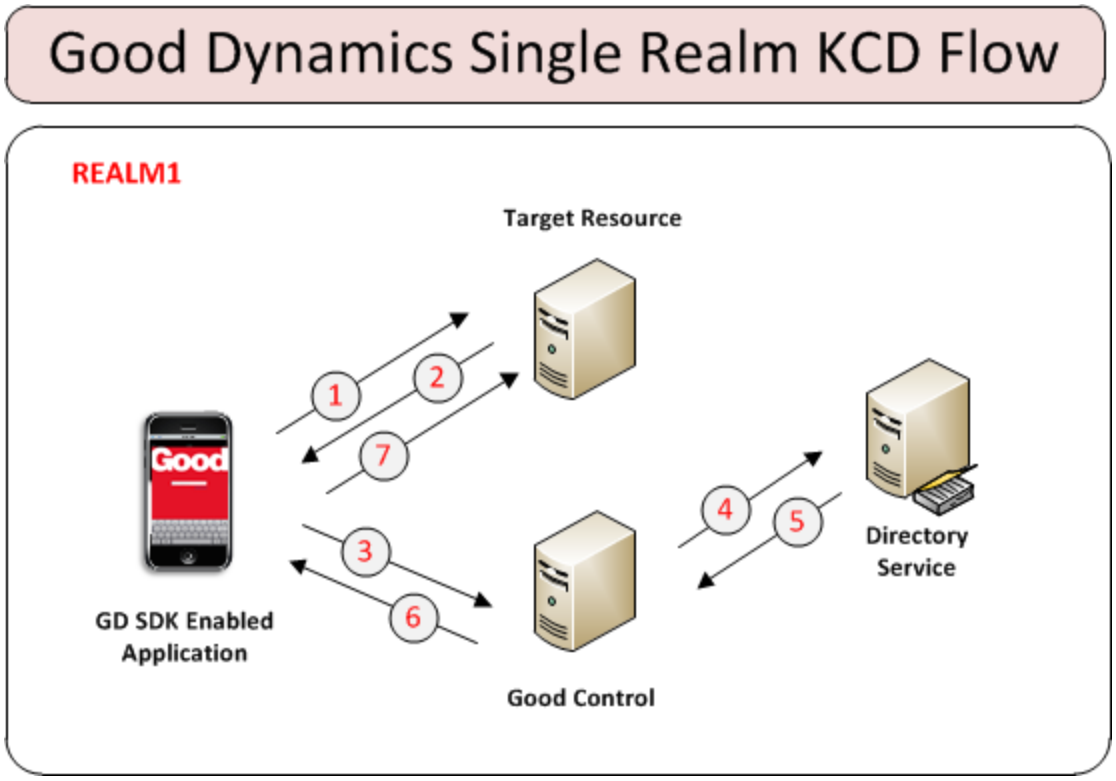
The terms *realm* and *domain* are conceptually equivalent in KCD. The relationship is as follows:

1 Kerberos realm : *n* Good Control server : *n* Good Proxy

There must be a minimum of one Good Control server for each Kerberos realm. The GC must still reside in the same Kerberos realm as the resource because cross-realm resource delegation is not supported. In addition, you must ensure that all Good Proxy servers and clusters can communicate with all other GC and GP servers that you intend to configuration for multi-realm KCD. Other requirements for multi-realm and forest topologies are detailed below.

Example Diagram: Single Kerberos Realm

A typical single realm KCD transaction works as follows.



1. An application makes a request an internal server or service. We call this the *target*.

   The *target* can be either a host name (server name) or an account that is to be protected by Kerberos/GD. For instance, if you are running IIS on a server as the Network service, then the *target* is the computer running IIS as Network. On the other hand, if you run IIS as an actual user (for instance, IISSrvUser), then *target* is that user name, IISSrvUser.

   Thus, if you have 1,000 IIS servers running as Network service, *target* is the names of those 1,000 machines. But if you run IIS as a user on those 1,000 machines, then *target* is the name of that user.

   Another example is WebSphere, in which case *target* is the Kerberos user name of the WebSphere service.
2. The target replies with an authentication challenge that the GD Runtime intercepts.
3. The GD Runtime sends a request to GC for a service ticket to access the target.

4. GC authenticates the user/container (via internal GD protocols) and asks for a service ticket on behalf of the user (this is *delegation*) for the service on the target host.
5. Active Directory (AD) checks its local policy. Then (a) if the user has permission to access the resource on the target host and (b) if the resource on the target host is allowed (this is *constrained*), AD returns to Good Control a service ticket for the resource.
6. Good Control sends the necessary information from the returned service ticket to the GD Runtime.
7. The GD Runtime uses the information from GC to complete the authentication to the target host.

DNS for GC and GP in Separate Domains, Kerberos vs. KCD

The Good Control server and the Good Proxy server are often installed in the same Kerberos domain but they do not have to be. You might want to install the GP in your DMZ or "sacrificial" workgroup. If you choose this latter configuration, you need to set-up some required network configuration, as detailed below.

There is a distinction in how Good Dynamics operates between normal Kerberos and Kerberos Constrained Delegation (KCD) that affects your network configuration.

- In KCD, on behalf of the client applications, the GC service itself requests authentication tickets from the ticketing server (the domain controller).
- In normal Kerberos (without constrained delegation), the client applications themselves make the ticketing requests, not the GC, and the request is passed through (egress)the GP. This means that the GP needs to be able to discover the name of Kerberos domain controller (server). In your Domain Name System (DNS), you need to add an `SRV` record specifying the Kerberos service that enables this discovery. This `SRV` record must be associated with an `A` or `AAAA` record, not a `CNAME` record. The syntax below is for a Kerberos domain controller in an Internet domain named example.com:

  ```
  _kerberos._tcp.example.com. 86400 IN SRV 0 5 5060 kerberos.example.com.
  ```

  This points to a server named kerberos.example.com listening on TCP port 5060 for Kerbeors requests. The priority given here is 0, and the weight is 5.

  Consult your networking documentation for the exact steps and definitions in creating a `SRV` record for your Kerberos services when the GP is in its own domain.

Requirements and Recommendations for Multiple Kerberos Realms and Forests

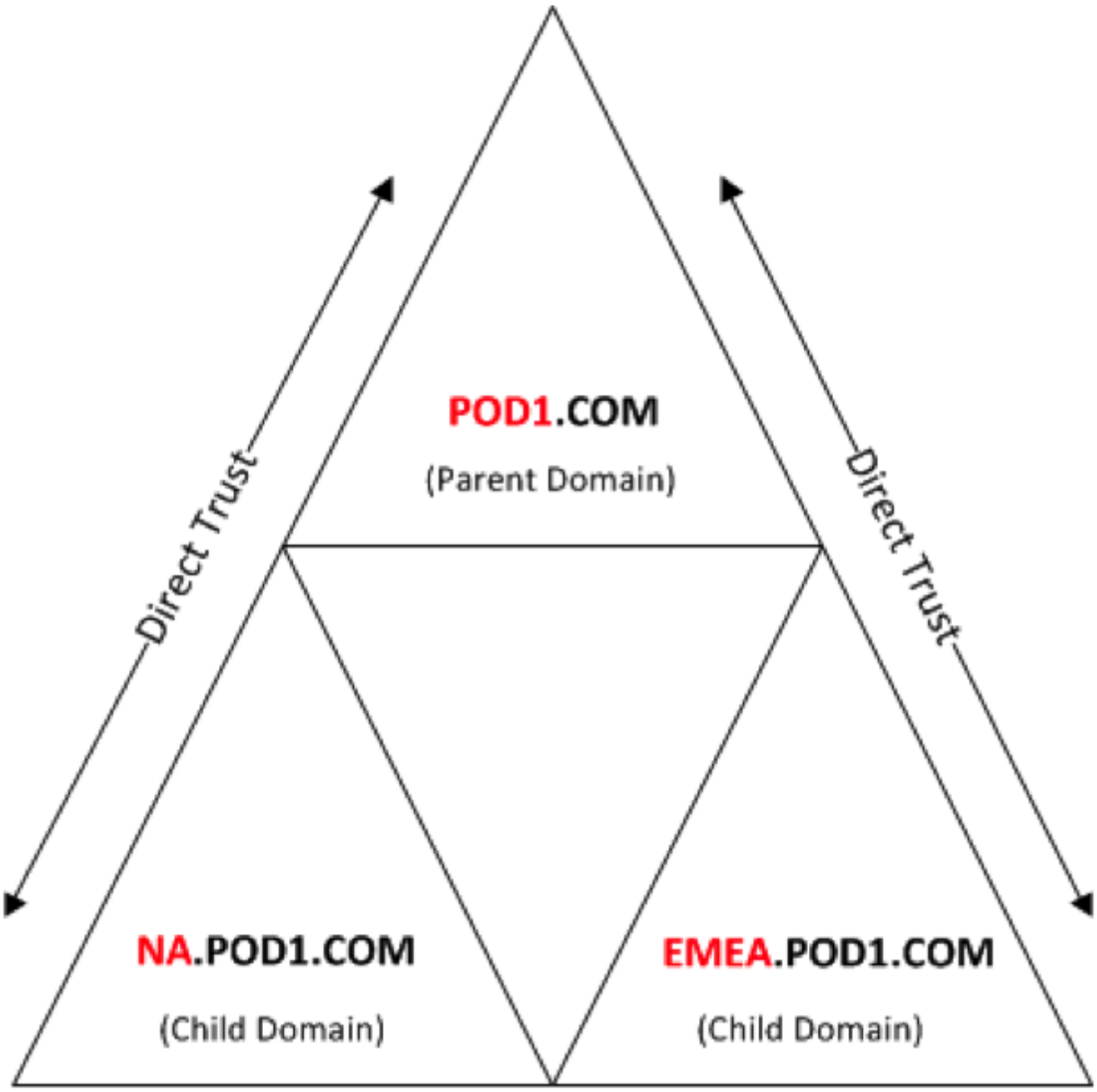
Note: Much of this material is available as a video tutorial from Good's Solutions Architecture group at https://community.good.com/videos/1474.

Multi-realm KCD is supported across any combination of Active Directory 2003, Active Directory 2008, and Active Directory 2012. For multiple Kerberos realms and forests, ensure the following prerequisites before implementation.

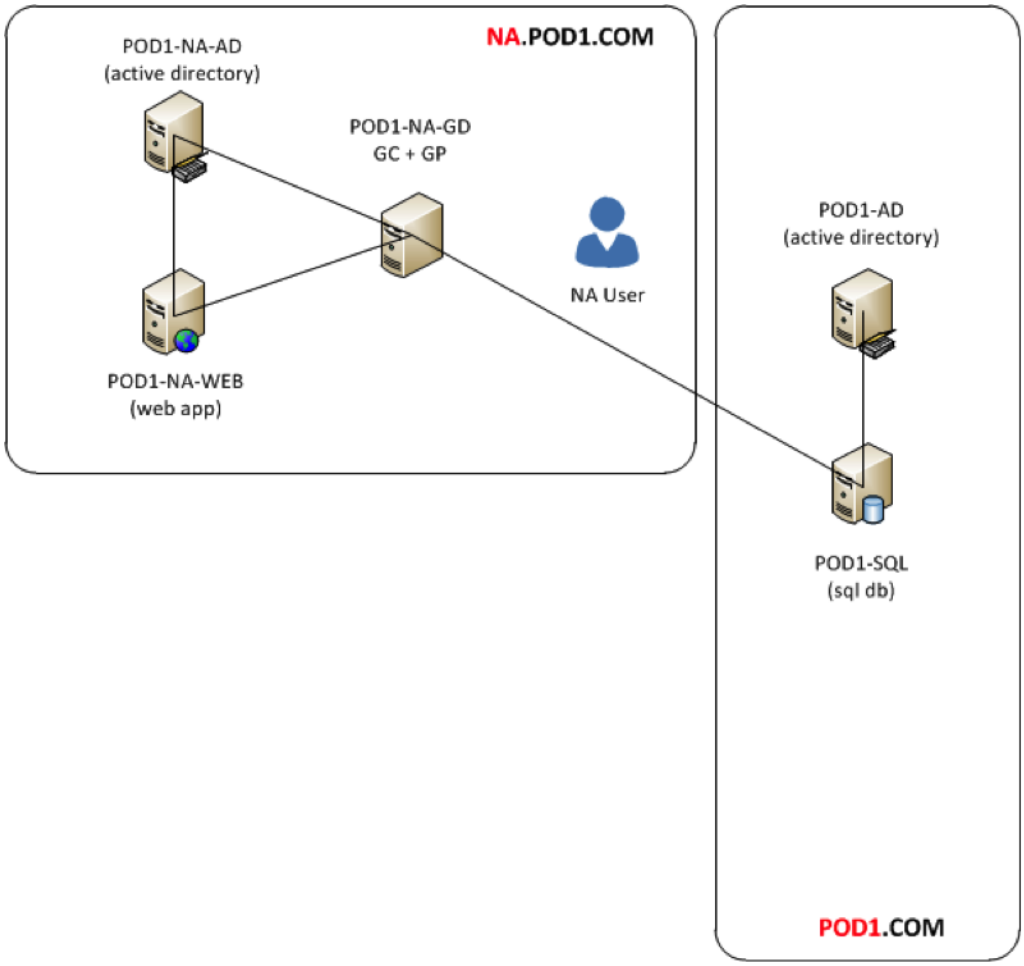
Good Control	SQL Database	KCD	Forests
• At least one Good Control server must be installed in every resource realm that has resources needed by other realms. • All Good Control servers must be in the same cluster. Therefore, they all must share the same database.	If you are using Microsoft SQL Server in the multi-realm configuration, make sure of the following: • Use an SQL account, not the Windows account. • That SQL account must be the owner of the database. • All GCs in the multi-realm KCD configuration must use that same SQL account.	• Ensure that single realm KCD is working before configuring multi-realm KCD. • Ensure that target resources are properly configured for KCD .	• All trust must be bidirectional, transitive forest trust. • If a Kerberos domain is between any two other domains, the trust must be transitive.

Example KCD Multi-Realm within Single Forest

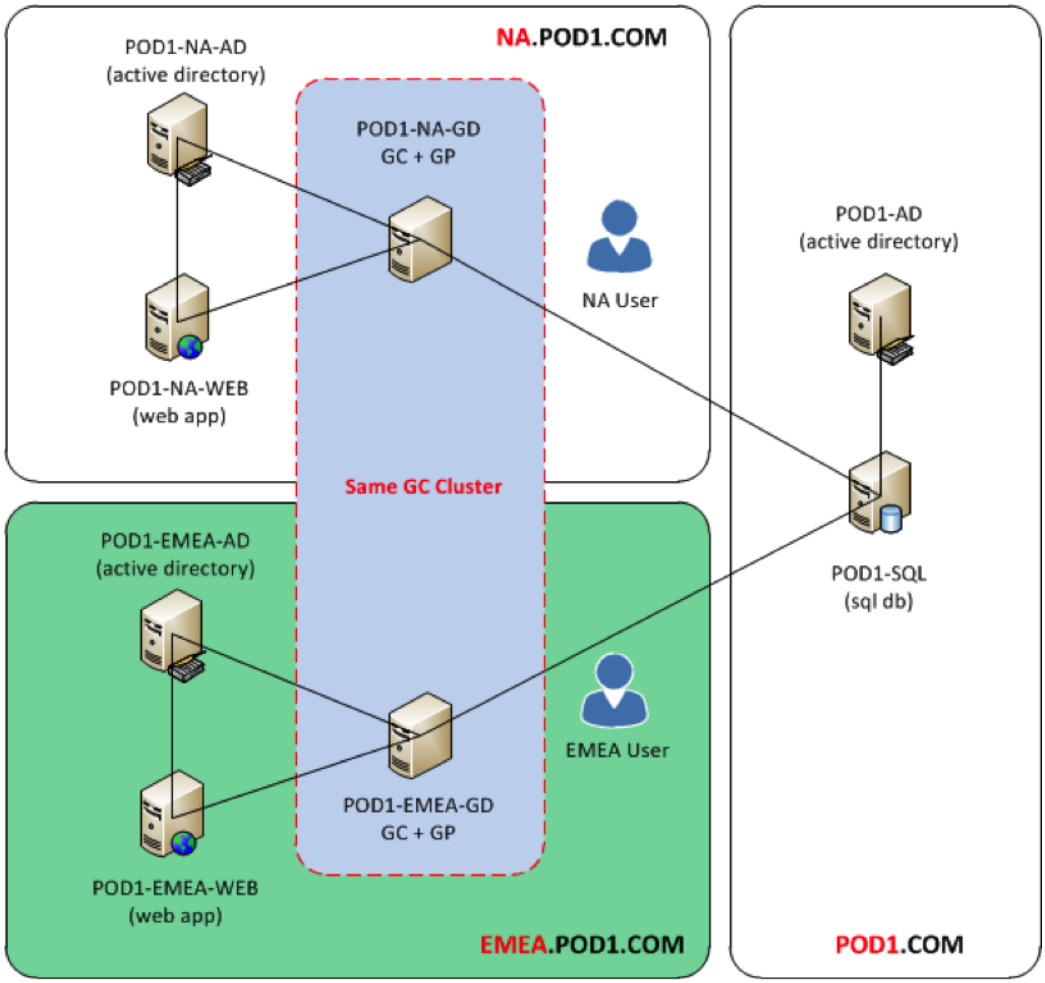
Consider the following diagrams. These are different views of the same deployment: 1. Trust relationships between domains in a single forest (without regard for Good servers)



2. Verification of the Good Control and Good Proxy connections to the parent domain, POD1.com



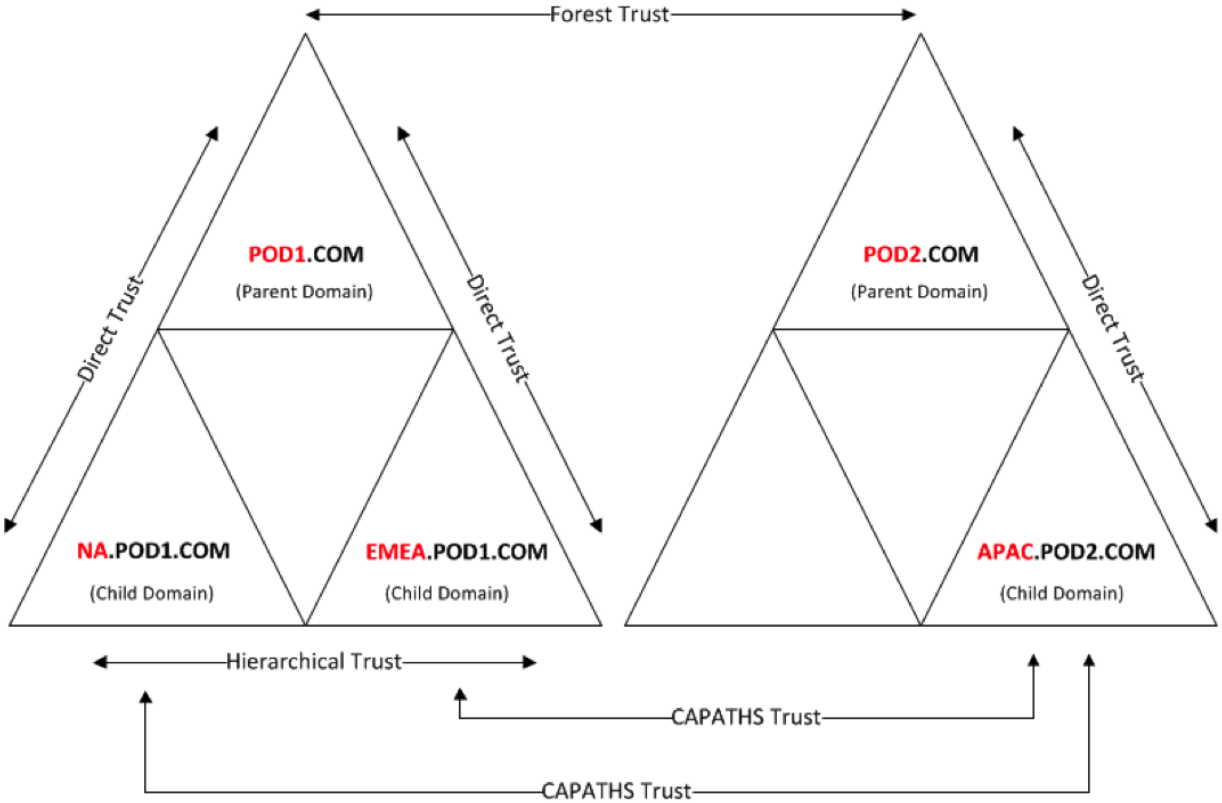
3. The logical layout of the multi-realm, single-forest configuration, with GC/GP servers



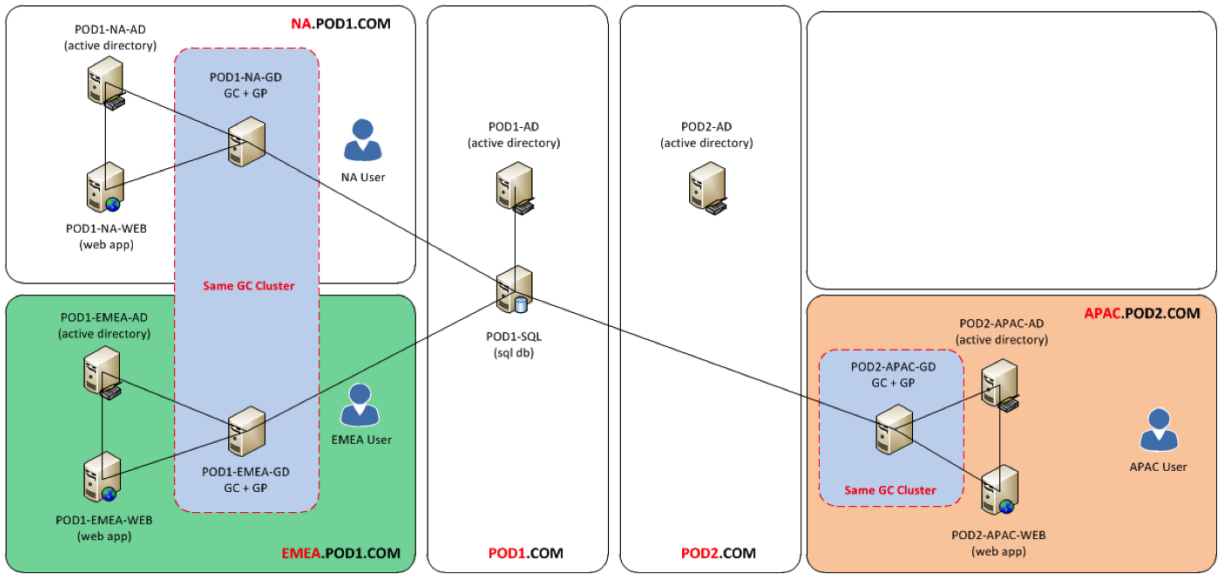
Example KCD Multi-Realm between Two Forests

Here are similar diagrams when two forests are involved.

1. Trust relationships between domains in two forests



2. Logical view of two forests and domains with GC/GP cluster

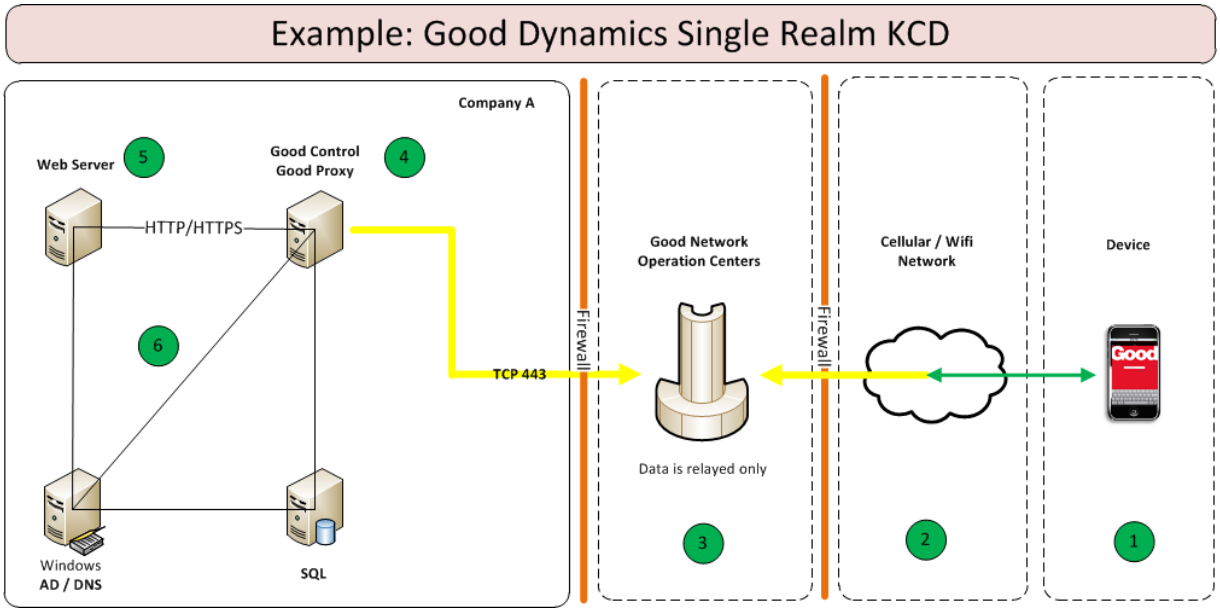


Concepts and Steps for a Typical KCD Single Realm Installation

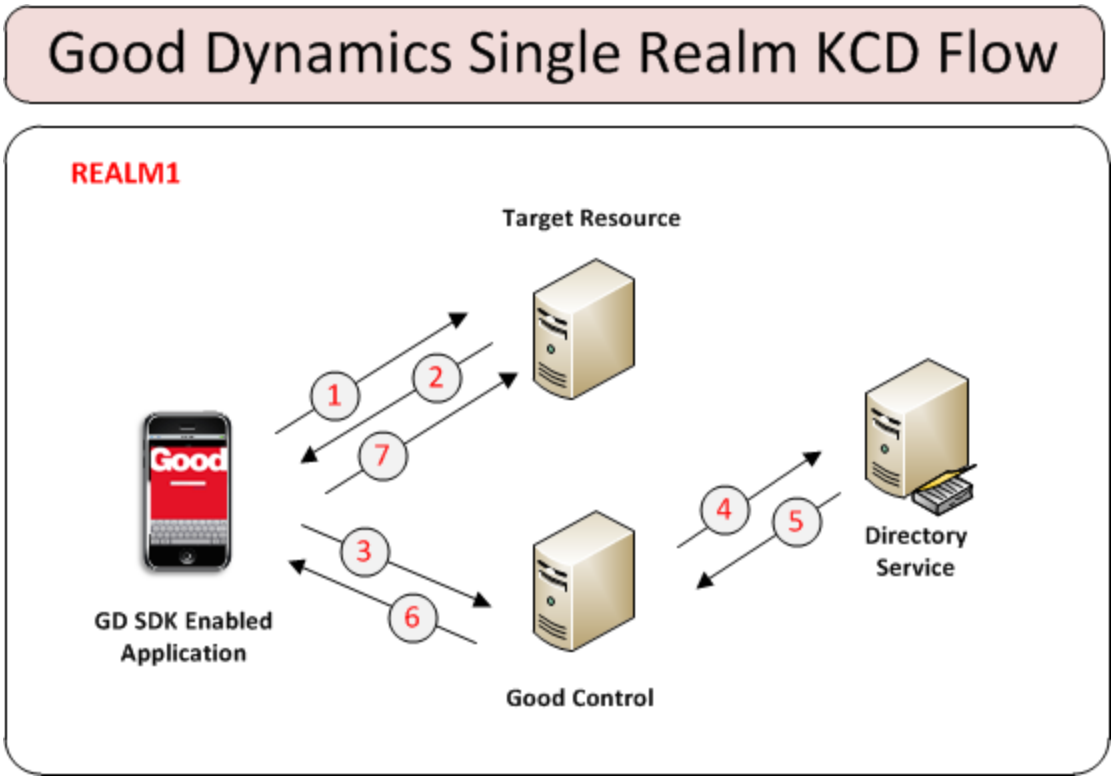
The steps that follow show how to set up a typical installation for a single Kerberos realm with Good Dynamics.

Example Diagrams: Single Kerberos Realm

Consider the following single-realm KCD topology:



A typical single realm KCD transaction works as follows.



1. An application makes a request an internal server or service. We call this the *target*.

   The *target* can be either a host name (server name) or an account that is to be protected by Kerberos/GD. For instance, if you are running IIS on a server as the Network service, then the *target* is the computer running IIS as Network. On the other hand, if you run IIS as an actual user (for instance, IISSrvUser), then *target* is that user name, IISSrvUser.

   Thus, if you have 1,000 IIS servers running as Network service, *target* is the names of those 1,000 machines. But if you run IIS as a user on those 1,000 machines, then *target* is the name of that user.

   Another example is WebSphere, in which case *target* is the Kerberos user name of the WebSphere service.
2. The target replies with an authentication challenge that the GD Runtime intercepts.
3. The GD Runtime sends a request to GC for a service ticket to access the target.
4. GC authenticates the user/container (via internal GD protocols) and asks for a service ticket on behalf of the user (this is *delegation*) for the service on the target host.
5. Active Directory (AD) checks its local policy. Then (a) if the user has permission to access the resource on the target host and (b) if the resource on the target host is allowed (this is *constrained*), AD returns to Good Control a service ticket for the resource.
6. Good Control sends the necessary information from the returned service ticket to the GD Runtime.
7. The GD Runtime uses the information from GC to complete the authentication to the target host.

Step 1: Map the Good Control (GC) Service Account to a Service Principal Name (SPN)

To use the command line, open an administrator command prompt on the AD machine and enter:

`setspn -a GCSvc/GC_host_machine DOMAIN\GC_service_account`

Replace the host machine name, domain, and service account variables with values appropriate to your environment. For example:

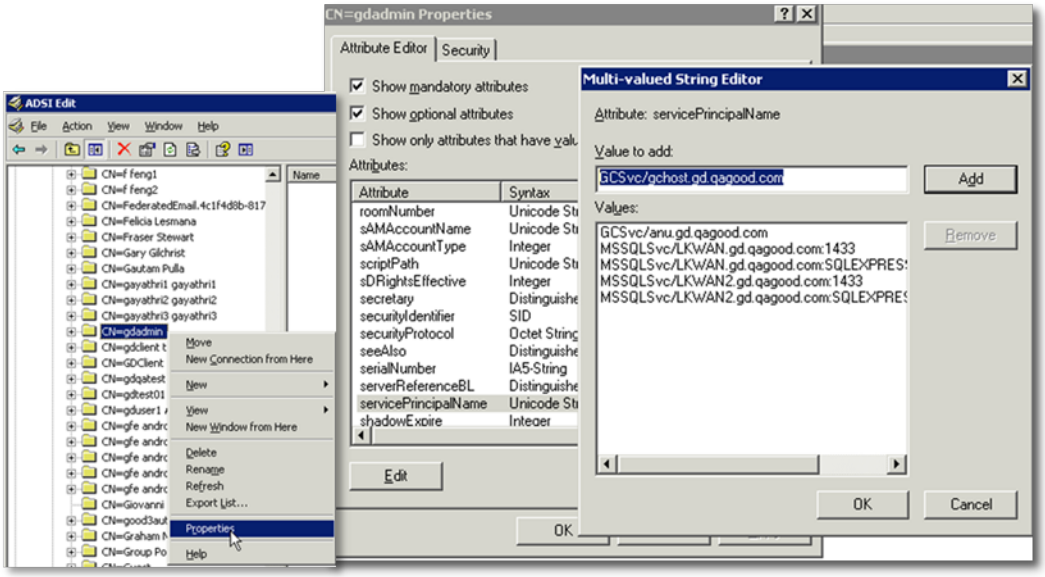
`setspn -a GCSvc/gchost.gd.qagood.com gd.qagood.com\gdadmin`

This should produce:

Or, alternatively, you can use the mmc console:

1. Open the **ADSI Edit** mmc console.
2. Right-click the GC service account and select **Properties**.

3. Select **servicePrincipalName** from the attributes list.



4. In the **Value to Add** popup, enter `GCSvc//`*`fqdn_gc_host_machine`* as a new SPN, using a host machine name appropriate to your environment.

Important: If you have clustered your GC servers, you must run these commands for every GC server in the cluster.

Step 2: Create a keytab file for the GC service account

1. Open a command prompt window on the KDC server
2. Use the `ktpass` command to set the Kerberos account password:

   `ktpass /out` *`outfilename`*`.keytab /mapuser` *`kerberos_account@REALM_IN_ALL_CAPS`* `/princ` *`kerberos_account@REALM_IN_ALL_CAPS`* `/pass` *`kerberos_account_password`* `/ptype KRB5_NT_PRINCIPAL`

   where:

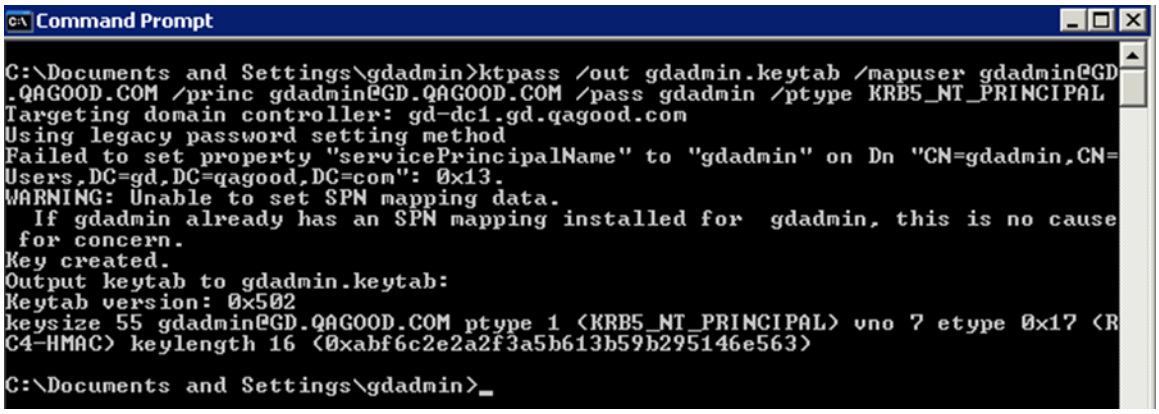
outfilename	is the name of the output file.
kerberos_account	is the Kerberos account name.
REALM_IN_ALL_CAPS	is the Kerberos realm in all capital letters.
kerberos_account_password	is the desired password. **Caution:** Make sure the *kerberos_account_password* is the same as the existing password for the Kerberos account. Enclose the value of *kerberos_account_password* in double quotation marks, especially if it contains special characters, such as ^.

Note: A new keytab file must be generated and copied to the GC machine whenever the *kerberos_account_password* is changed.

Note: In the following example, note that the Kerberos realm `GD.QAGOOD.COM` is in all capital letters:

```
ktpass /out gdadmin.keytab /mapuser gdadmin@GD.QAGOOD.COM /princ gdadmin@GD.QAGOOD.COM
/pass gdadmin /ptype KRB5_NT_PRINCIPAL
```

This results in the following example:



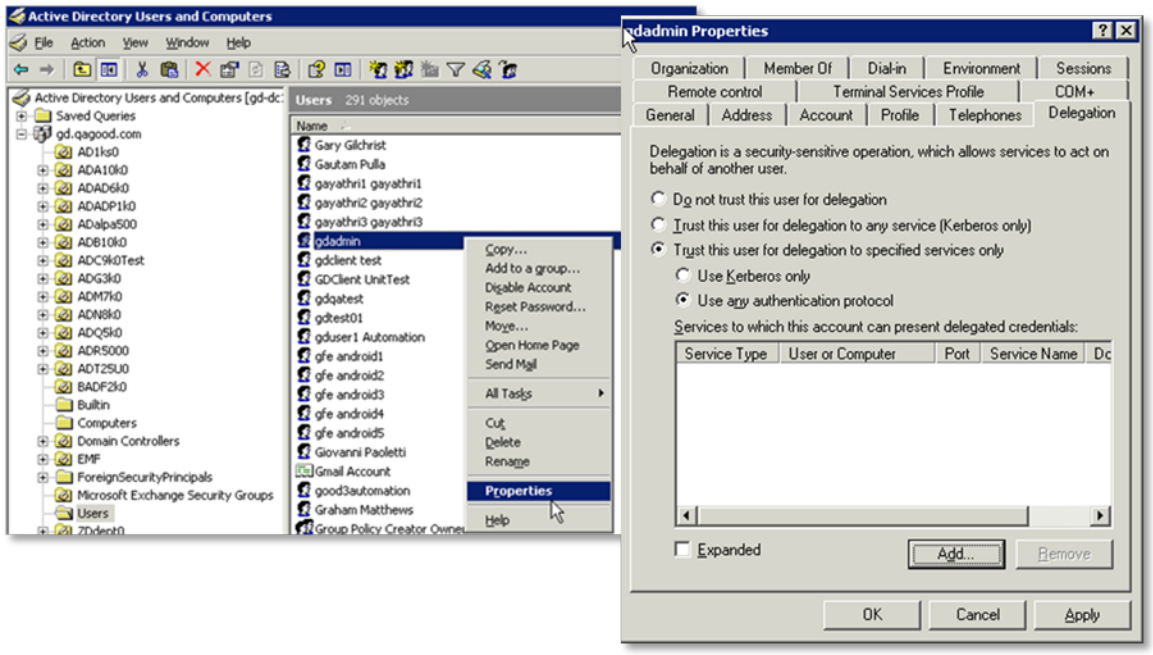
3. Copy the new keytab file (**gdadmin.keytab** in the example) saved in this directory to the GC server.

   Important: If you have clustered your GC servers, you must copy the keytab file to every GC server in the cluster.

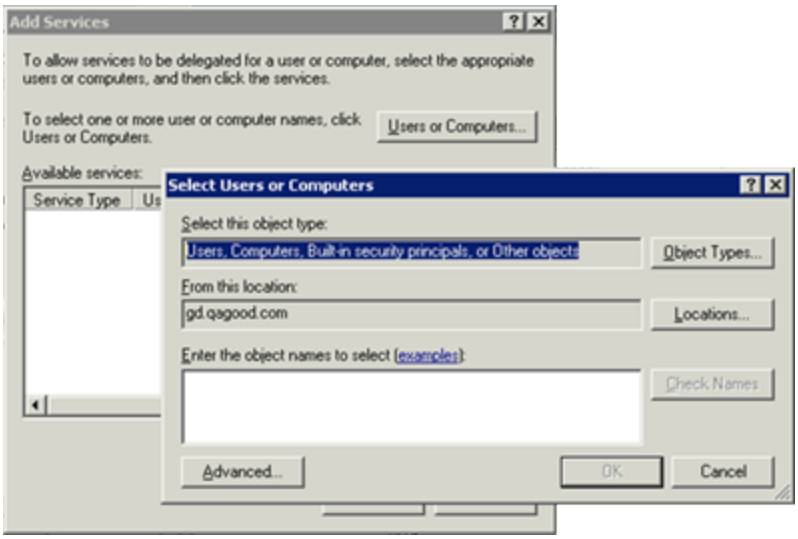
Step 3: Configure constrained delegation for the GC service account

1. In your Active Directory Users and Computers mmc console, right-click the GC service account in the list of users and select **Properties**.

2. In the **Properties** popup, click on the **Delegation** tab.



3. Enable Trust this user for delegation to specified services only by activating its radio button.
4. Also enable Use any authentication protocol, then click Add...

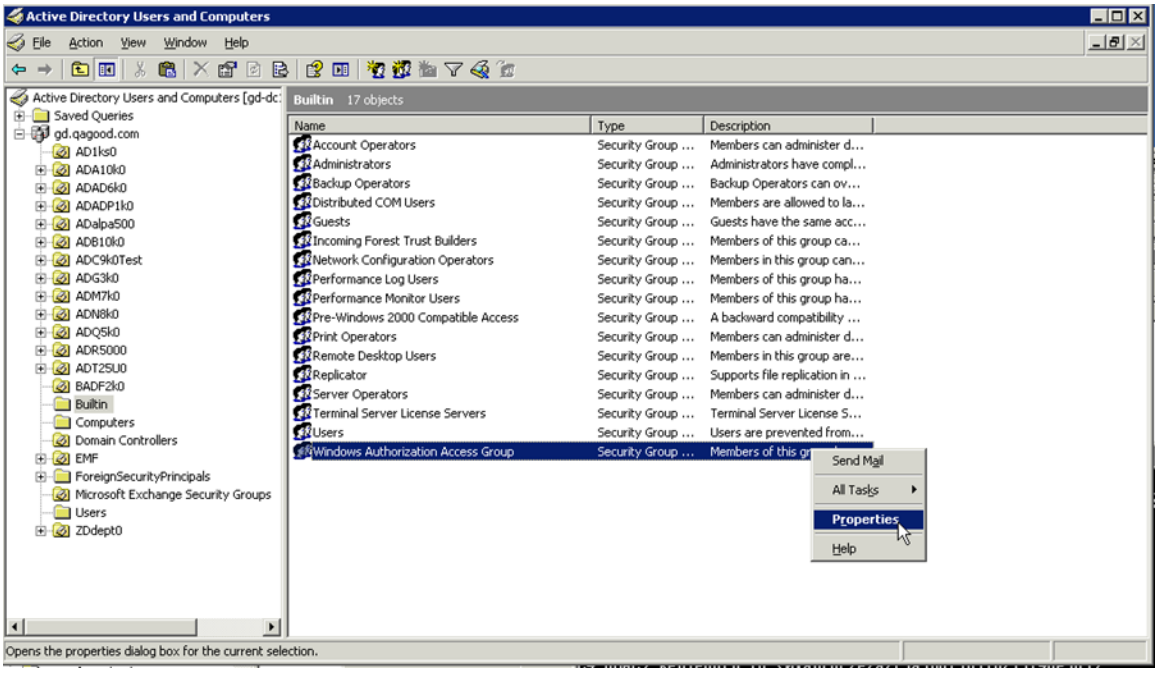


5. In the **Add Services** popup, click the **Users** or **Computers** button.
6. In the **Select Users or Computers** popup, enter the name of the *target* (the server name or user account to be protected by Kerberos), then click **OK**. For a discussion of *target*, see How Kerberos Constrained Delegation Works.

7. Click **OK** in the **Add Services** popup, then click **OK** again in the **Properties** popup.

Step 4: Enable enumeration of AD user objects group membership

1. In your **Active Directory Users and Computers** mmc console, select **Builtin** from the list on the left, then right-click **Windows Authorization Access Group** and select **Properties**.

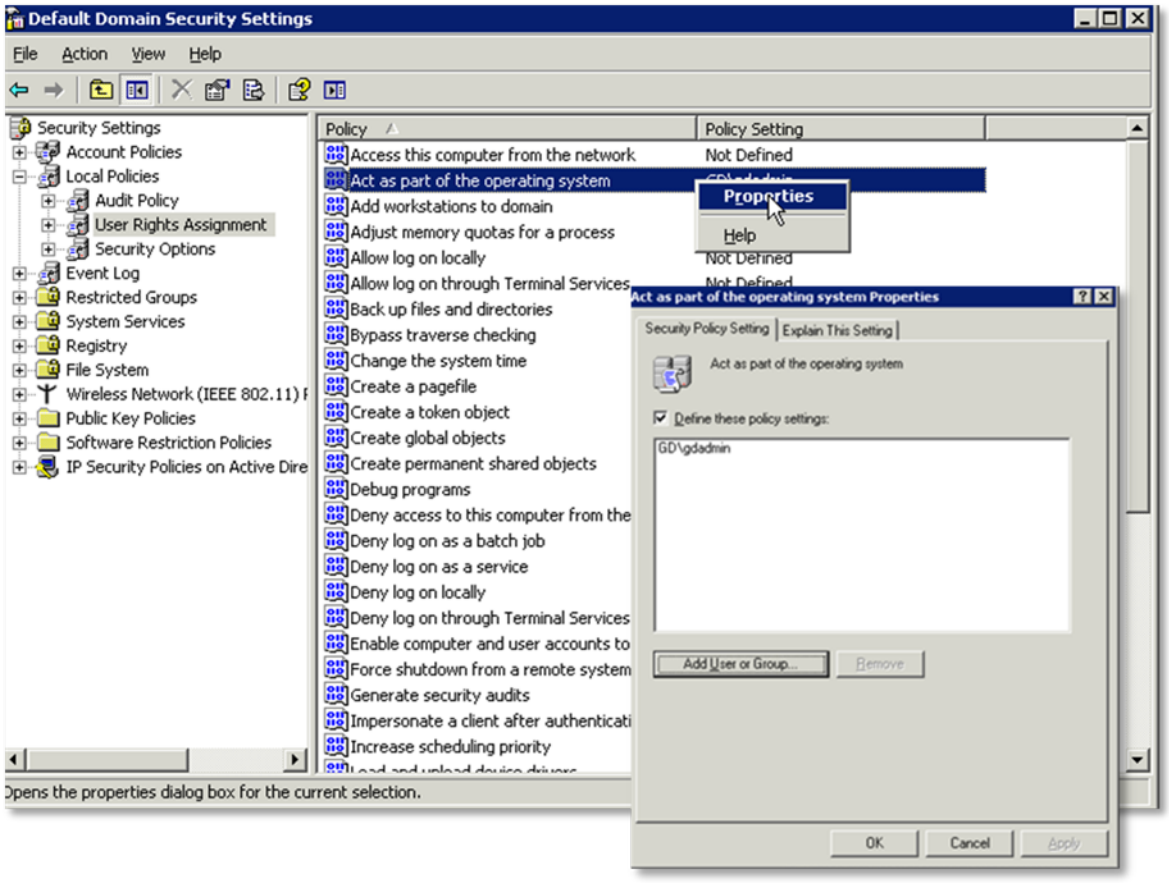


2. In the **Windows Authorization Access Group Properties** popup, click the **Members** tab, then click **Add...**
3. In the **Select Users**, **Contact**, **Computers** or **Group** popup, enter the name of the GC service account, and click **OK**.

Step 5: Enable the GC service account to act as part of the operating system

1. Open the **Default Domain Security Settings** mmc console.
2. Under **Local Policies**, select **User Rights Assignments**, then right-click **Act as part of the operating system**

in the right panel and select **Properties**.



3. In the **Properties** popup, click on **Add User or Group...**, then enter the name of the GC service account and click **OK**.

Concepts and Steps for KCD Multi-Realm Configuration

Be sure to be familiar with the concepts and steps in Concepts and Steps for a Typical KCD Single Realm Installation . The steps here assume your familiarity with the commands and steps for a single realm.

With multi-realm configuration, always start by configuring and testing a single realm first. Then proceed to adding the other realms or forests.

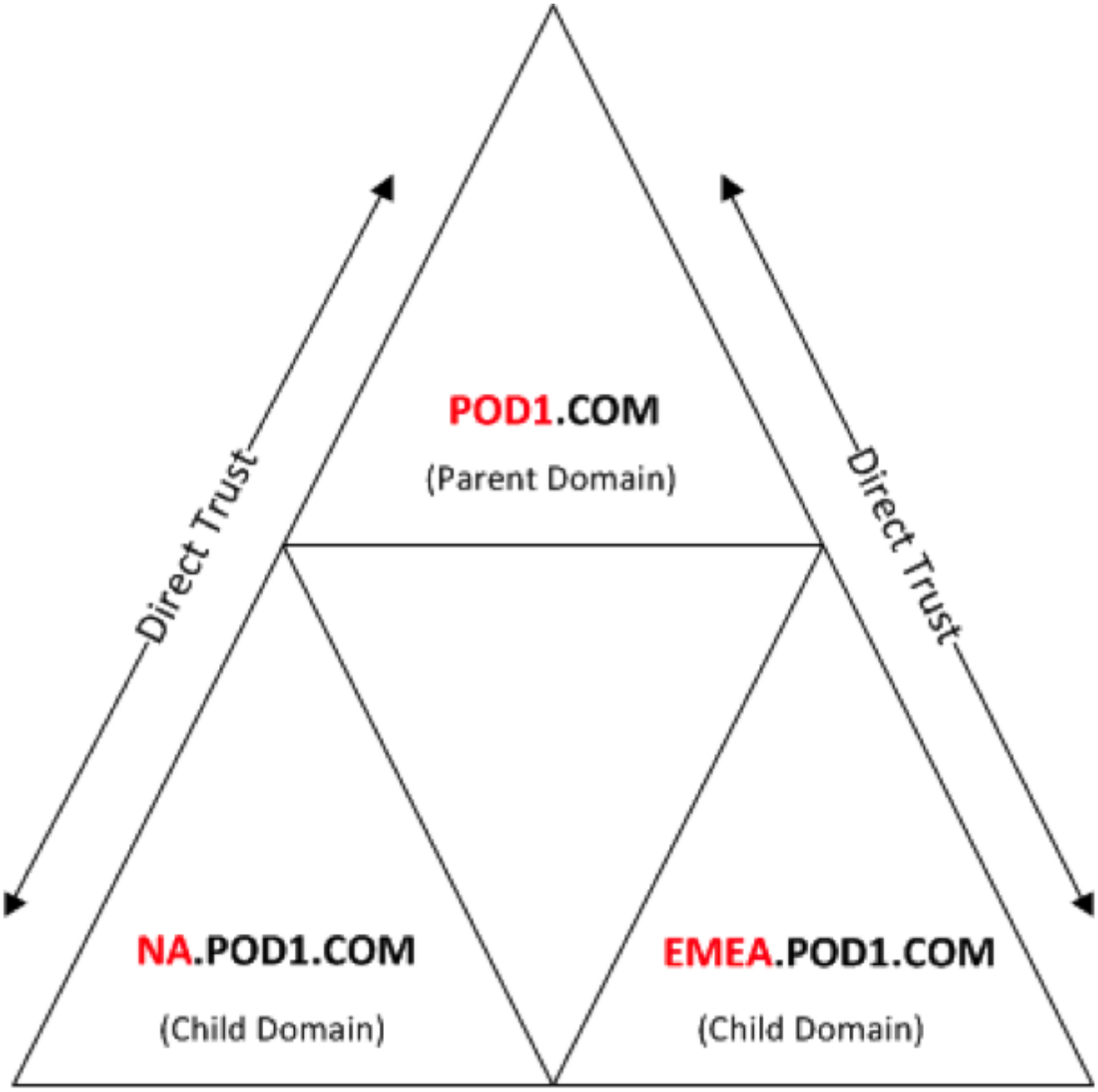
SQL Database Account for Multi-Realm KCD Configuration

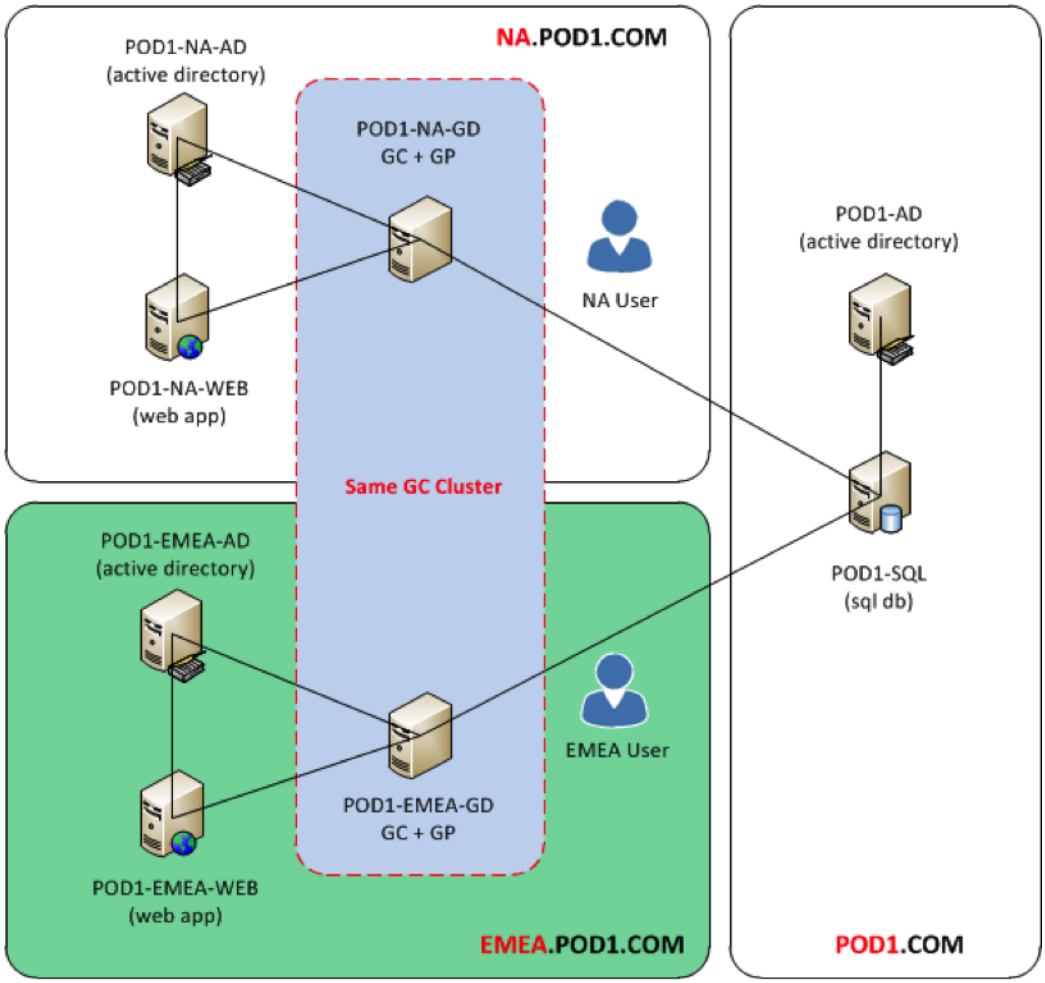
If you are using Microsoft SQL Server in the multi-realm configuration, make sure of the following:

- Use an SQL account, not a Microsoft Windows service account.
- That SQL account must be the owner of the database.
- All GCs in the multi-realm KCD configuration must use that same SQL account.

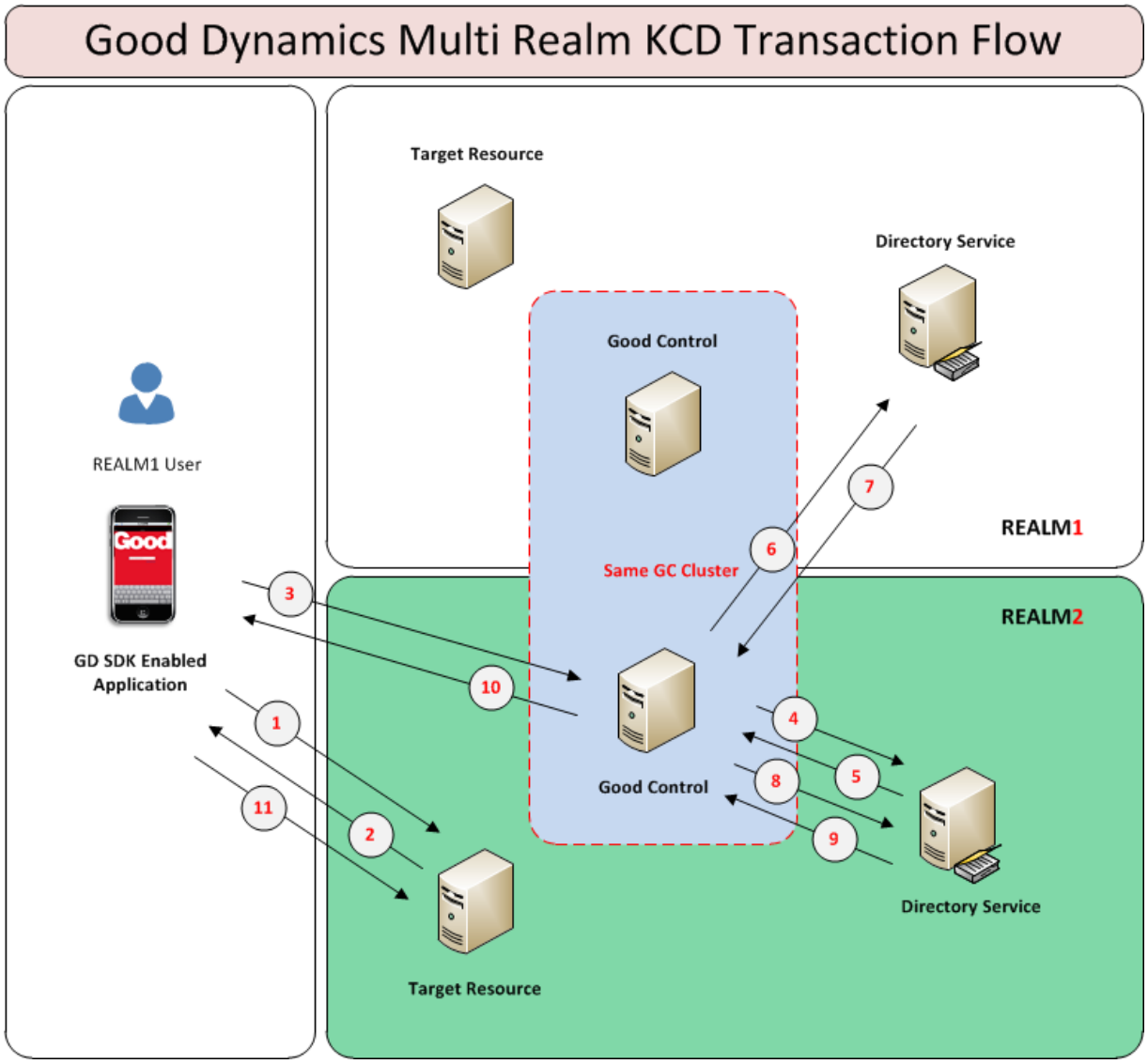
Concepts and High-Level Steps for Single Forest

Consider the trust relationships among the domains shown below:





The flow with KCD mutli-realm topology is like this:



1. A user in realm 1 with a GD SDK enabled application makes a request to a resource in realm 2
2. The target resource replies with an authentication challenge that the GD runtime intercepts
3. Base on the challenge information, the GD runtime sends a request to a GC that resides in the same realm as the target resource. The request is for a service ticket to access the target resource.
4. GC authenticates the user/container (via internal GD protocols) and asks for a service ticket on behalf of the user (this is delegation) for the service on the target host.
5. Active Directory (AD) checks its local policy and determines that the user is in a remote realm. AD returns a referral to the GC to contact the remote AD for authorization.
6. GC follows the referral to the new AD server for authorization.

7. The realm 1 AD server receives and authorization request. Base on itsí policy, the AD server will return the appropriate authorization response.
8. The GC sends the authorization response to the original AD in realm 2.
9. Base on the authorization response, (a) if the user has permission to access the resource on the target host and (b) if the resource on the target hose is allowed then AD returns to Good Control a service ticket for the resource
10. Good Control sends the necessary information from the returned service ticket to the GD runtime.
11. The GD runtime uses the information from GC to complete the authentication to the target host.

Configuration Steps

The steps here are high-level. They rely on information from Concepts and Steps for a Typical KCD Single Realm Installation and on other sources, as indicated below.

Step	See Also...
Determine the name of your service account to use in the other steps.	
Add Service Account as a administrator in Good Control.	Good Control help
Add Service Account that is actual Owner of the GC database.	
Install GC/GP to the same cluster.	GD Server Installation guide
Create SPN for Service Account.	Step 1: Map the Good Control (GC) Service Account to a Service Principal Name (SPN)
Create Keytab file for Service Account.	Step 2: Create a keytab file for the GC service account
Configure constrained delegation for Service Account.	Step 3: Configure constrained delegation for the GC service account
Add Service Account to AD Group: “Windows Authorization Access Group”.	Step 4: Enable enumeration of AD user objects group membership
Add Service Account to Local Security Group: “Act as part of Operating System”.	Step 5: Enable the GC service account to act as part of the operating system
Create krb5.conf file. Only needed if there is a CAPATH trust	Example krb5.conf Files for CAPATH Trust
Update GC Configuration.	
Configure Test KCD website for testing (optional)	

Configuring Kerberos-related Properties in Good Control

In GC's **Server** > **Server Properties** tab, the following settings are required to enable KCD in Good Control.

Property	Optional/Required	Description
`gc.krb5.enabled`	Required	Check this to enable KCD.
The first four properties listed below must be set when gc.krb5.enabled = true.		
	`gc.krb5.kdc=`*`kdc_host_name`* Required	*`kdc_host_name`* is the fully qualified name for the KDC. Usually corresponds to the Active Directory address.
	`gc.krb5.principal.name=`*`gc_service_account`* Required	*`gc_service_account`* is the service account name under which the KCD service is running.
	`gc.krb5.realm=`*`REALM`* Required	*`REALM`* is the name of the Active Directory realm. Example: `GD.QAGOOD.COM`. **Note:** The value must be in all capital letters.
	`gc.krb5.keytab.file=`*`keytab_file_location`* Required	*`keytab_file_location`* is the location of the keytab file. Example: `C:/good/gdadmin.keytab`. **Note:** Do not use backslashes in this path name. Use normal slashes.

Troubleshooting and Diagnostics

GD diagnostic aids are in place to help you troubleshoot real or suspected KCD problems by detecting and/or reporting issues that your system administrator can either fix or else refer the known details to Good technical support for timely investigation and resolution.

Kerberos and KCD Log File Error Codes

Information captured in your GC server logs can often help to explain Kerberos authentication and KCD issues/errors. Here's an example of a Kerberos error log.

```
com.good.gmc.security.kerberos.KerberosException: Failed to impersonate userPrincipal tanu100@GD.QAGOOD.COM;
     krbErrCode: 6;
     krbErrText: Client not found in Kerberos database
     at com.good.gmc.security.kerberos.impl.KerberosServiceImpl.impersonateUser(KerberosServiceImpl.java:211)
     at com.good.kcd.TicketProxy2$ProxyHandler.fetchServiceTicket(TicketProxy2.java:168)
     at com.good.kcd.TicketProxy2$ProxyHandler.messageReceived(TicketProxy2.java:145)
```

```
        at org.apache.mina.core.filterchain.DefaultIoFilterChain$TailFilter.messageReceived(DefaultIoFil-
terChain.java:716)
        .
        .
        .
```

The two most important parameters in the error messagesare **krbErrCode** and **krbErrText**, which furnish a description of possible error conditions detected.

Complete documentation for Microsoft's Kerberos error messages is available at http://technet.microsoft.com/en-us/library/bb463166.aspx .

Useful KCD Links

Kerberos Constrained Delegation Overview

http://technet.microsoft.com/en-us/library/jj553400.aspx

About Kerberos constrained delegation

http://technet.microsoft.com/en-us/library/cc995228.aspx

Troubleshooting Kerberos Errors

http://www.microsoft.com/en-us/download/details.aspx?id=21820 outlines basic troubleshooting strategies. Summarizes issues that typically cause problems with Kerberos authentication. Lists Kerberos error messages, possible causes, and possible resolutions. Describes tools commonly used to troubleshoot Kerberos authentication problems.

Logon and Authentication

See http://technet.microsoft.com/en-us/library/cc786325(v=ws.10).aspx.

Authentication – Name Resolution Issues

See http://blogs.technet.com/b/askds/archive/2008/05/14/troubleshooting-kerberos-authentication-problems-name-resolution-issues.aspx

Authentication – Service Principal Name (SPN) Issues

Refer to the following (consists of three parts):

- http://blogs.technet.com/b/askds/archive/2008/05/29/kerberos-authentication-problems-service-principal-name-spn-issues-part-1.aspx
- http://blogs.technet.com/b/askds/archive/2008/06/09/kerberos-authentication-problems-service-principal-name-spn-issues-part-2.aspx
- http://blogs.technet.com/b/askds/archive/2008/06/11/kerberos-authentication-problems-service-principal-name-spn-issues-part-3.aspx

How to Enable Kerberos Event Logging

See http://support.microsoft.com/default.aspx?scid=kb;EN-US;262177

Using Command Line Tools to Troubleshoot Kerberos (YouTube)

Watch http://www.youtube.com/watch?v=wNSfFBhLywk.

Troubleshooting Kerberos Constrained Delegation (YouTube)

Watch http://www.youtube.com/watch?v=Tf3bxY0eSSY.

Troubleshooting Kerberos Setup and Secure Searches

See https://developers.google.com/search-appliance/kb/secure/kerberos-troubleshooting.

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

Category	Title	Description
		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 • 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*

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*