Good Dynamics™

Technical Brief: Server Clustering and Affinities

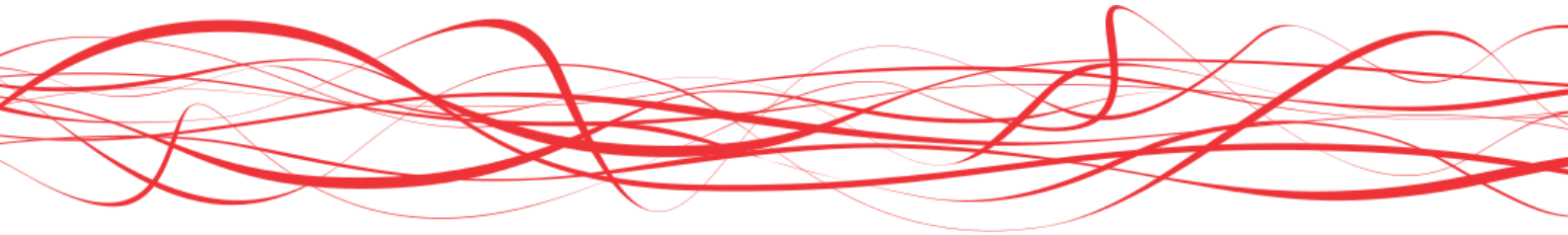


Table of Contents

1 Introduction

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

1.1 Server Clustering

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.

All 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:

Scale up the capacity for concurrent users.

Deliver high availability and disaster recovery (HA/DR).

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

1.2 Affinities

The 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, and as part of an enterprise HA/DR strategy. Enterprises may also have other motivations for allocating their GP servers between their GCs and application servers.

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, which should improve speed of data transfer.

An enterprise may assess that some of its data is especially sensitive. As a result, the enterprise may wish to segregate the servers that handle this data, and put additional protection 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 could also be aligned with the enterprise organisation'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.

1.3 Implementation

The server clustering and affinities features are enabler-type features. Configuration is required at the enterprise in order to realise the features' benefits.

The GC 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.

The remaining sections of this document describe what kinds of configuration are supported, and say how communication set-up is affected.

1.4 Background Reading

This document is a full technical brief. For a shorter overview of these features, see the Feature Overview: Server Clustering and Affinities document.

https://begood.good.com/docs/DOC-1976

For background reading on the roles of the GC, the GP, and the application servers in a GD deployment, see the Good Dynamics Administrator and Developer Overview. This is available on the Good Dynamics Network website here:

https://begood.good.com/docs/DOC-1061

2 Server Clustering

Server clustering is the deployment of a group of servers that works as a single node. The benefits of server clustering are widely known, and are outlined in the Introduction, above.

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

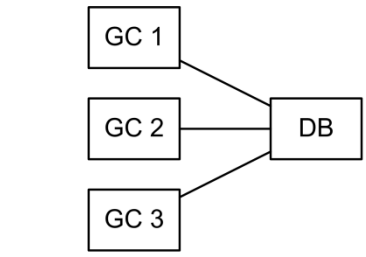
In GD, a server cluster does not consist of a mixture of servers that fulfil different roles.

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

The following sub-sections describe the configuration and communication set-up in a number of clustering scenarios, starting with the simplest.

2.1 Basic Good Control Clustering

To deploy a GC cluster, the enterprise installs multiple GC instances that communicate with the same GC database.



Brief: Basic GC Clustering
Brief Server Clustering and Affinities ad hoc.vsd v3.02

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.

2.1.1 Installation and System Administration

Installation of GCs in a cluster is similar to, but 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 Dynamics 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

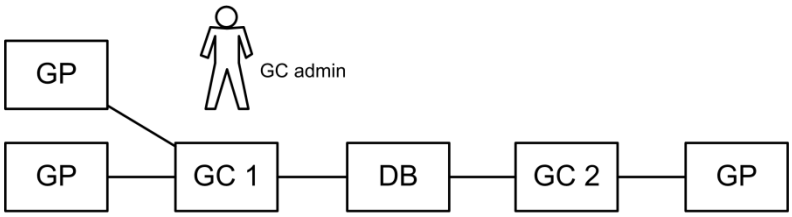
used to generate license keys for clustered installation. A copy of the key will be stored in the central GD Network Operation Center (NOC), so that the new GC will be able to connect to the GD infrastructure.

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.



Brief: Administration of GC Cluster
Brief Server Clustering and Affinities ad hoc.vsd v3.02

Warning 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 user's changes will be overwritten by the other's.

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 under Applications Programming Interface, below, 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.

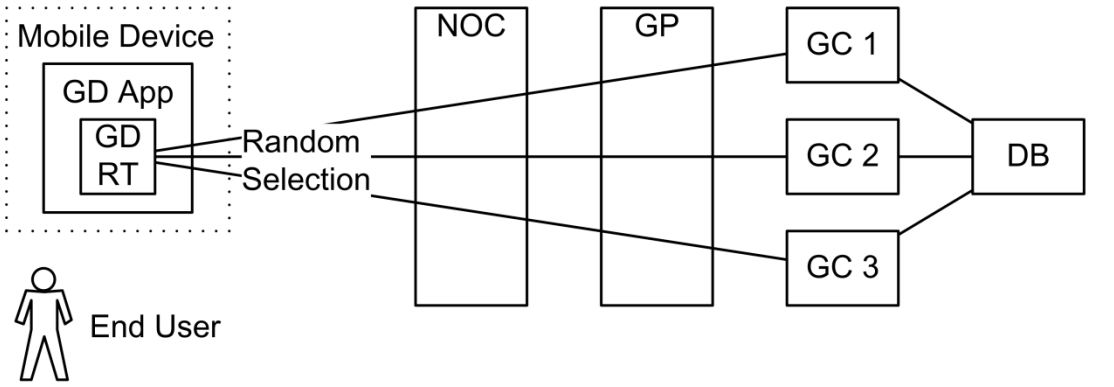
During installation of a GP, the address of a GC server must be entered. Where a GC cluster is in use, the address of any server in the cluster can be used for this purpose.

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.

2.1.2 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 diagrams) 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.



Brief: Application Communication in a Basic GC Clustering Deployment
Brief Server Clustering and Affinities ad hoc.vsd v3.02

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.

2.1.3 Resilience

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

Connection set-up 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.

(Side 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 above scenario, where a GC is off line when a connection attempt is being made, is one possible failure. Another possible failure is the scenario that a GC goes off line some time 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.

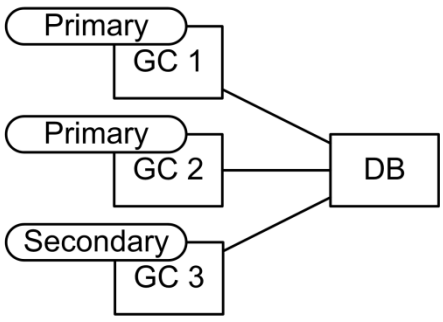
These resilience features are implemented by the GD runtime and infrastructure platform, and do not require special coding in the application layer.

2.2 Good Control Clustering with Server Priorities

The GC servers in a clustered deployment can be assigned server priorities. Server priorities dictate the order in which the GC servers will be tried when connection is required by a GD application.

Server priorities are typically stated as primary, secondary or tertiary, but could also be considered numeric.

Each GC server can be assigned a different priority, or multiple GC servers can share the same priority.



Brief: GC Clustering with Server Priorities
Brief Server Clustering and Affinities ad hoc.vsd v3.02

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.

2.2.1 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, see above.

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 tasks such as the following:

Assign the priority of a new GC server.

Change the priority of an existing GC server.

Note that 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.

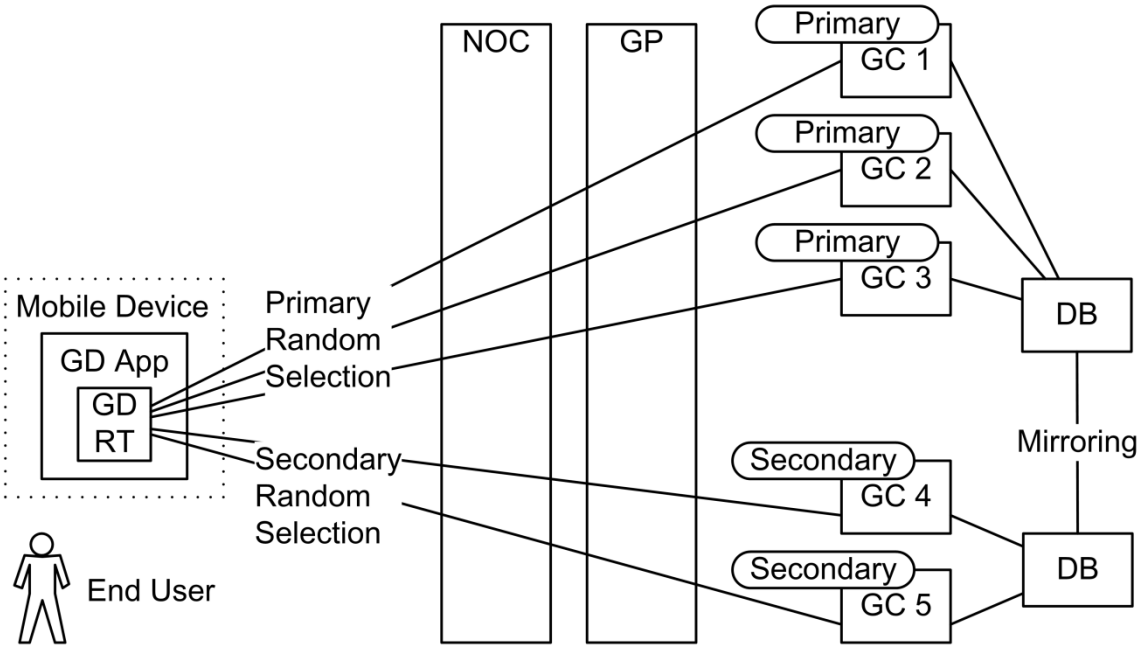
2.2.2 Application Communication

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.



Brief: Application Communication in a GC Deployment with Server Priorities
Brief Server Clustering and Affinities ad hoc.vsd v3.03

2.3 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 the Installation and System Administration sub-sections, above.

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 prioritised server selection algorithm as the GD runtime implements for communication with GC servers. See the Application Communication sub-sections, above.

If the application utilises Push Channel notifications, then the Ping Fail capability could be used for resilience. See the Resilience sub-section of the Basic Good Control Clustering section, above, for a description of how the GD runtime uses the capability with server clustering.

Note that the application could instead implement its own variant of the GC cluster communication set-up 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.

This is described further under Application Server Priorities and Affinities in the General Communication Set-Up Processing Flow section, under Combining Server Clustering with Affinities, below.

2.4 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 the Affinities section, below.

A GP cluster is different to a GC or application server cluster in a number of ways.

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 the Affinities feature section, below, 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.

2.4.1 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 under Applications Programming Interface, below, for details.

2.5 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.

Note that there is no limit on the number of proxy clusters.

3 Affinities

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

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 utilise affinities, the enterprise must organise 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 following sub-sections describe the configuration and communication set-up in a number of affinity scenarios, starting with the simplest.

3.1 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 will be made as follows:

Connections to a host server are made via a GP from the proxy cluster with which the host server has an affinity.

The GP server to use is selected at random from within the cluster.

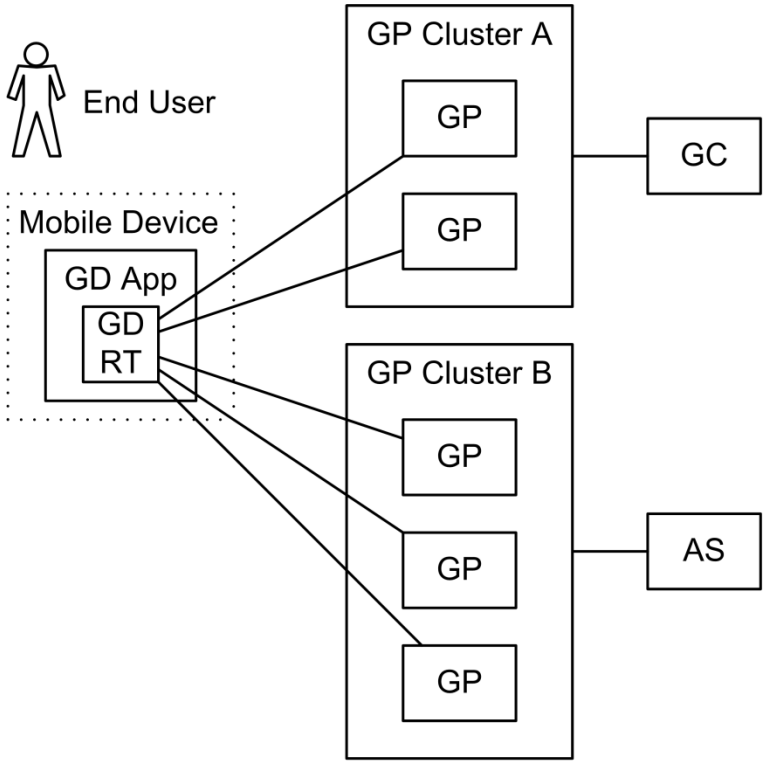
If the initially selected GP is over capacity, offline or otherwise inaccessible, then another GP in the same cluster is selected, again at random. This continues until an accessible GP is found, or until there are no more untried GP servers in the cluster.

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 set-up procedure is followed by the GD runtime for all host servers. This is different to server clustering, see above, where the GD runtime only implements the whole procedure for the GC, not for application servers.

Note that 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.

Larger clusters, those with more servers, are expected to be more resilient than smaller clusters. Hence, 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.



The above diagram illustrates:

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.

3.1.1 Configuration

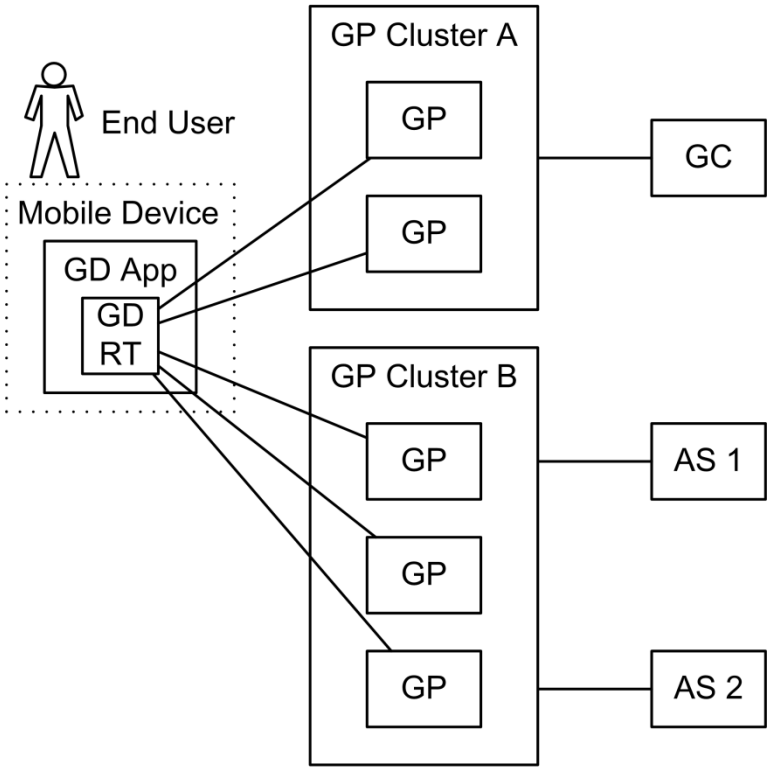
Affinity set-up is carried out in the GC console user interface, like any other configuration of the GD infrastructure within the enterprise.

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.

3.2 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.



Brief: Shared Affinities
Brief Server Clustering and Affinities ad hoc.vsd v3.02

The above diagram illustrates:

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.

3.3 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.

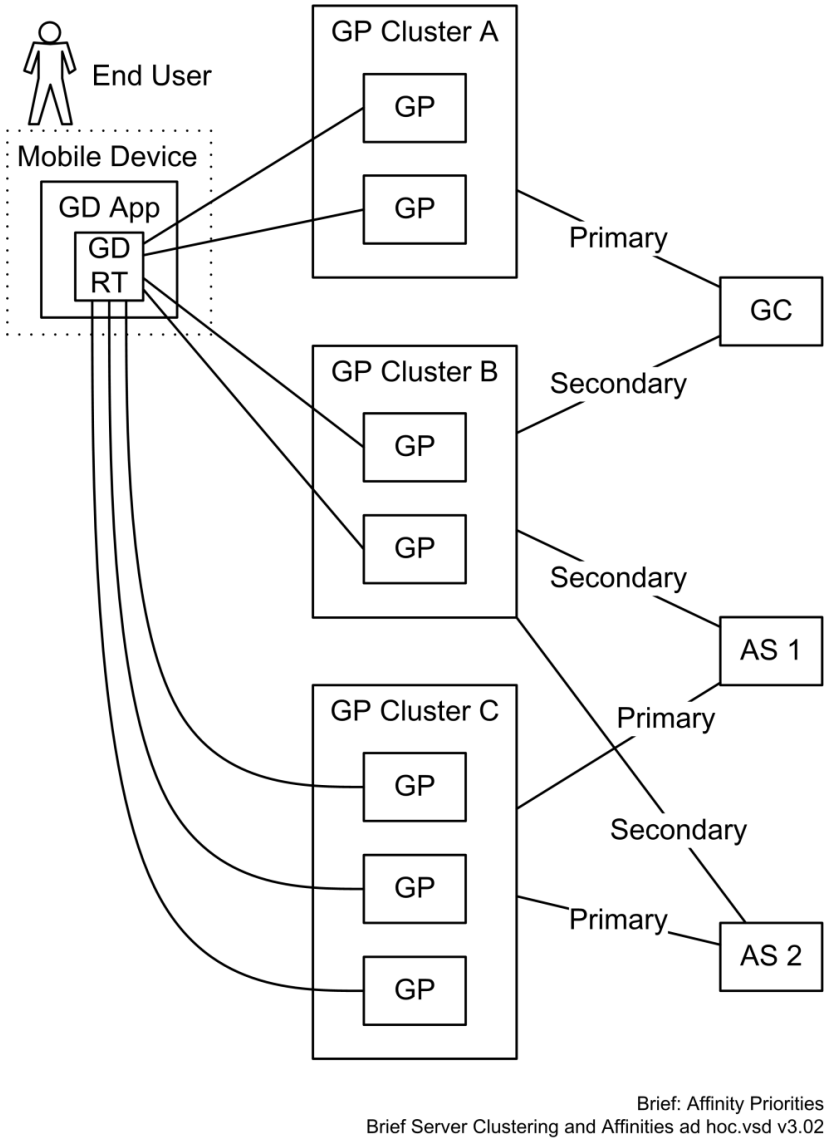
Compared to the simple affinities configuration, above, connection to a host server for which affinity priorities have been configured involves an additional selection step.

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 that 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.



The above diagram illustrates:

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.

3.4 Affinity Priority and Server Priority

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

3.5 Limitations

At time of writing, 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 utilise a GP when connecting to a host server that is located behind the enterprise firewall. It is unnecessary for a GD mobile application to utilise 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 utilised, 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.

4 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. These aspects are described in the preceding sections. The following sections describe other aspects, which are best understood when the features are considered in combination.

4.1 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 set-up in this type of deployment is also a combination of that for the features when separate. There are two levels to the set-up, 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.

4.1.1 Host Connection

Communication set-up 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 set-up.

If proxy connection set-up succeeds then host connection is attempted, using the proxy connection. If this also succeeds, then communication set-up overall has succeeded. Otherwise, if proxy communication set-up 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 set-up 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 set-up 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 set-up has failed.

Note that communication set-up could fail because no proxy connections can be set up, even if one or more host servers are actually running.

4.1.2 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

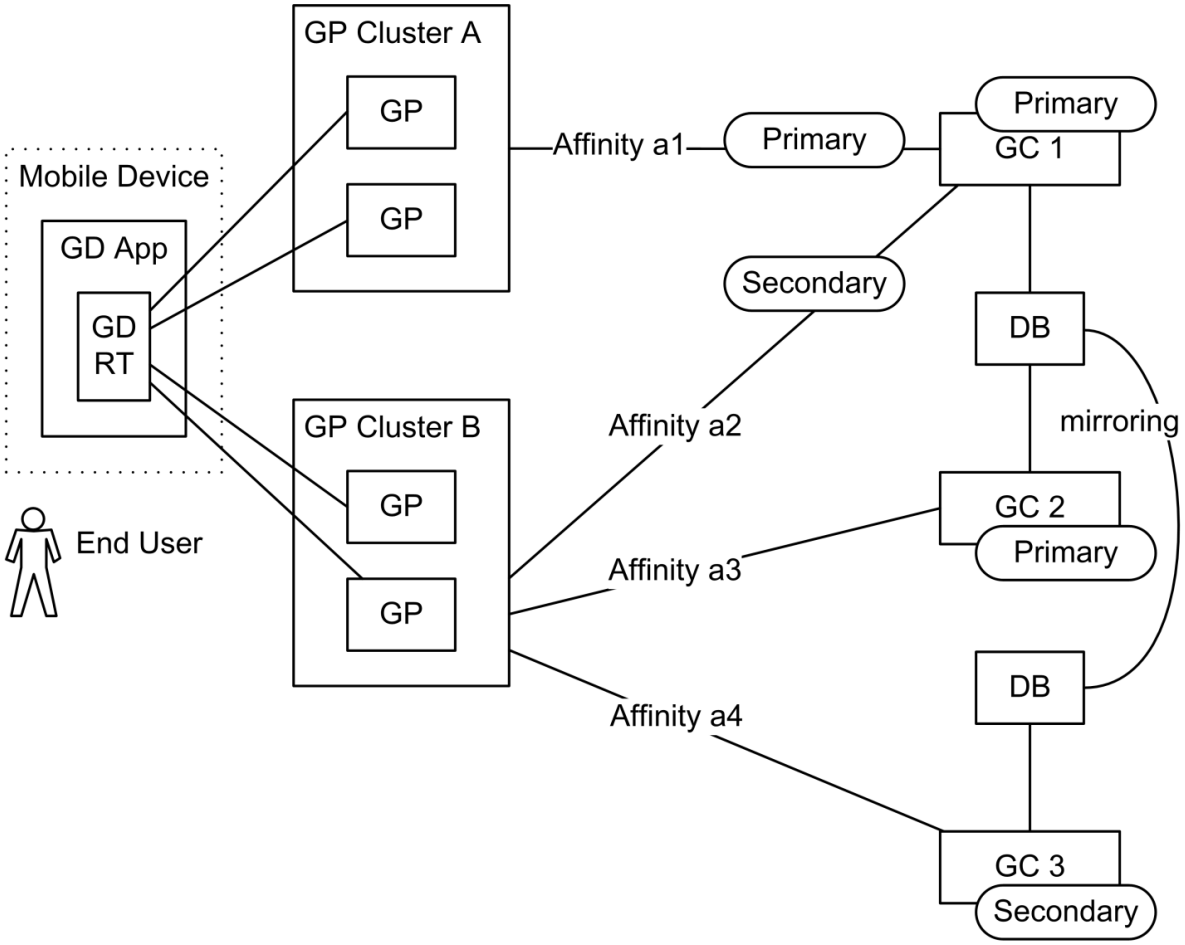
server, or an application server. In either case, the set-up of the proxy connection takes place in the GD runtime, not in the application code. Proxy connection requires connection to a GP server.

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 set-up 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 set-up fails.

4.1.3 Illustrative Deployment Diagram



Brief: Server Clustering and Prioritised Affinities
Brief Server Clustering and Affinities ad hoc.vsd v3.02

The above diagram shows an illustrative deployment where:

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. Note that 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 utilised instead.

The server clustering and affinity configuration illustrated in the above diagram could be represented as the following table.

Server	**Address** (not in diagram)	**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	

See also the Configuration Data Structure section, below, but note that the above is a denormalised representation.

4.1.4 Application Servers

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

As mentioned in some preceding sections, 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.

This is covered in more detail under the Applications Programming Interface heading, below.

4.1.5 Resilience Summary

Communication set-up 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.

4.2 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 would be as follows.

4.2.1 Configuration

The following configuration steps would be followed.

a) 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.

b) 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.

c) 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 that the configuration can be changed later. GP servers could be moved between clusters, new clusters could be created, new GC and GP servers could be installed, affinities could be re-assigned, GC server priorities could be changed, and so on.

4.2.2 Connection

When a connection from a GD application to Good Control is required, this proceeds as follows.

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:

5.1 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.

5.2 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:

5.2.1 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.

5.2.2 If there are no more untried proxy clusters with which the host server has an affinity:

5.2.2.1 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 paragraph, above.

5.2.2.2 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 paragraph, above.

5.2.2.3 If there are no more untried host servers, then communication set-up 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:

6.1 If host server connection succeeds, then communication set-up has **succeeded**.

6.2 If host server connection fails:

6.2.1 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 paragraph, above.

6.2.2 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 paragraph, above.

6.2.3 If there are no more untried host servers, then communication set-up has **failed**.

This is the complete connection set-up processing model.

4.2.3 Application Server Notes

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 the same as 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 remaining processing would have to be implemented by the application developer. In the terms used above, the application code has to implement:

Selection of host server priority.

Selection of host server.

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 under Applications Programming Interface, below, for a description of the API.

4.2.4 Configuration Changes

It is possible that an enterprise's server clustering and affinities configuration is changed whilst a connection attempt is in progress. The change will be handled by different components, depending on the type of change.

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 would include 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.

4.2.5 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 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 set-up attempt. At time of writing, the failure period is 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, above, could be implemented. A variant or alternative algorithm could also be implemented, as it could for selection of an application server within a cluster.

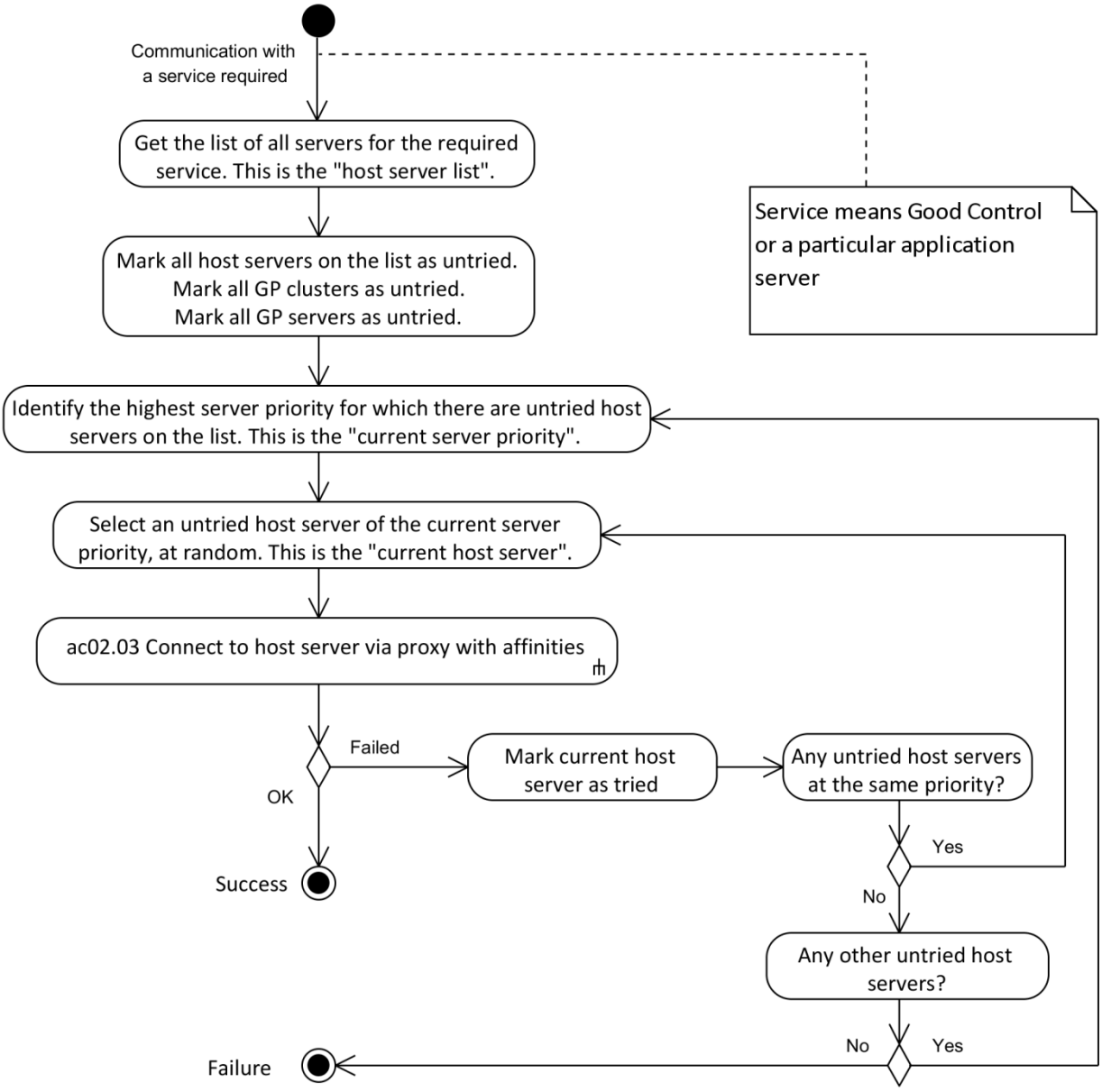
A GD application that makes only a short-lived connection to its application server implicitly implements retry primary, if the application makes a new server selection for every connection.

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, as discussed above.

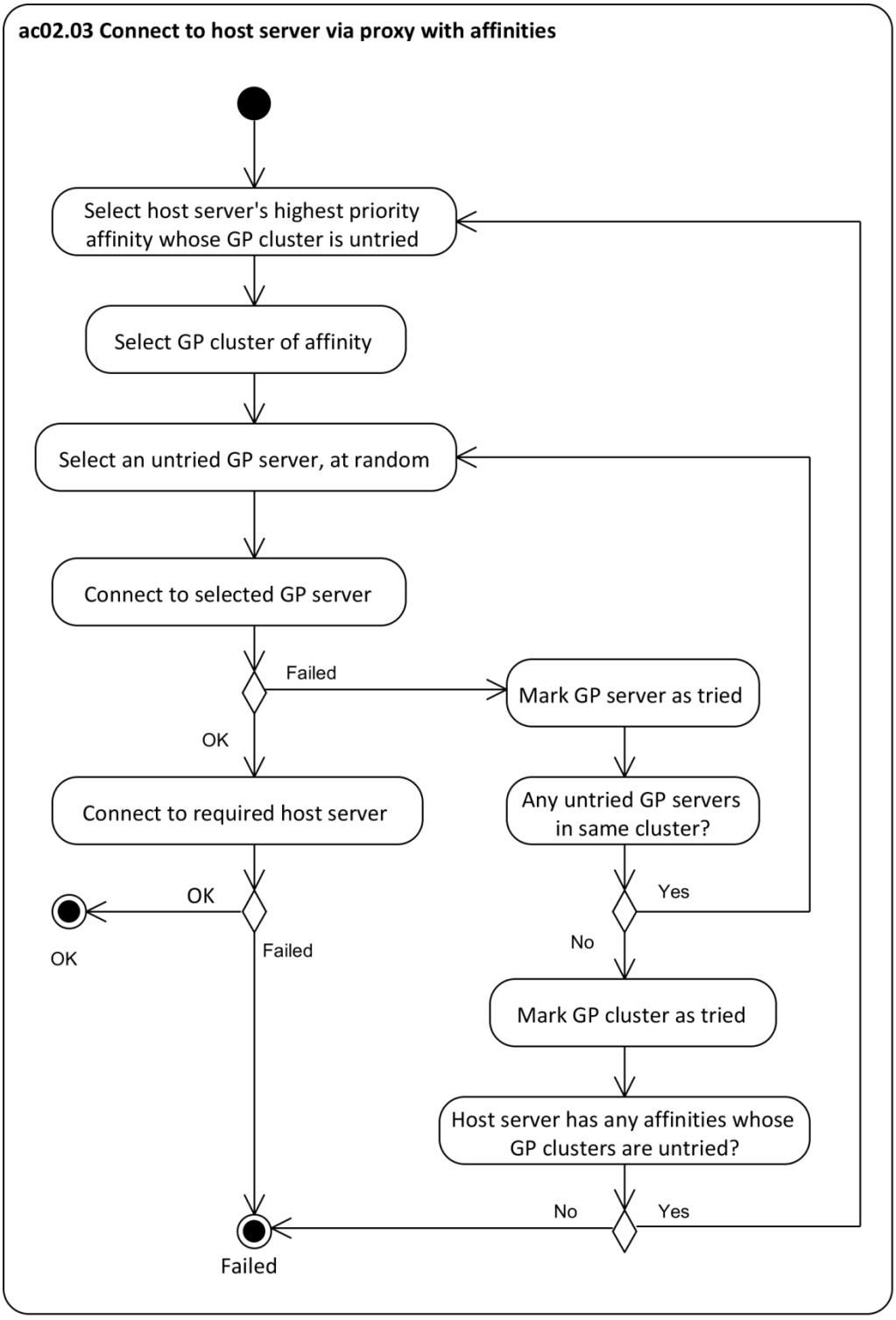
4.2.6 UML Activity Diagrams

The following diagrams illustrate the above processing 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.



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.



Name	ac02.03 Connect to host server via proxy with affinities
Documentation	UML Activity diagram showing the algorithm for selection of proxy cluster and GP server with affinities. For simplicity, historical failures of GPs are not shown, not are any optimisations.

4.3 Configuration Data Structure

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

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

4.3.1 Description

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

a) 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.

b) 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.

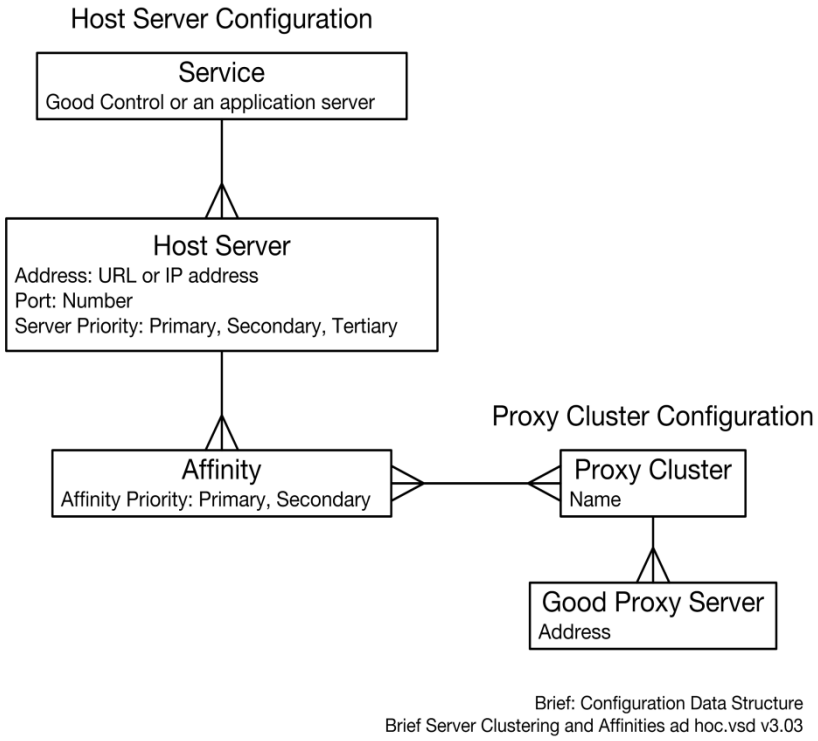
c) A Host Server has one or more *Affinity* relationships with one or more Proxy Cluster instances. An Affinity can have a priority order, expressed as primary or secondary.

d) A Proxy Cluster has zero or more Affinity relationships with Host Servers. Every *Proxy Cluster* has a name.

e) A Proxy Cluster consists of a number of *Good Proxy Server* instances. Any Good Proxy server is in only one proxy cluster.

4.3.2 Diagram

The following diagram summarises the above data structure.



4.4 Applications Programming Interface

The applications programming interface (API) of Good Dynamics provides support for server clustering and affinities in a number of places.

4.4.1 Good Dynamics 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 in the Complete Processing Model section, above.

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 under Further Reading, below, for details of the API.

Note that there is no GD API for any affinities capability, such as 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.

4.4.2 Good Proxy 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 under Affinity Priorities in the Affinities section, above. 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 under Further Reading, below, for details of the API.

4.4.3 Good Control Web Services

There is a GC web services API, 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, in order 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 the Application Communication sub-sections in the Server Clustering section, above.

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 that 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 forwards.

4.5 Backward Compatibility

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

4.5.1 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 utilise 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.

4.5.2 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.

The mobile application will behave as though the Default Configuration, see under Network Administrator Notes, below, is in effect. In this case, there will appear to be a single server with primary priority for the GC, and the same for any application server.

4.5.3 Good Dynamics 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 the references under Further Reading, below, for details.

4.6 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 to clustered deployments.

The principal differences are:

- 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 utilise server clustering, but the clustered resources are only available within that deployment.

4.7 Network Administrator Notes

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

4.7.1 Default Configuration

In principle, the use of server clustering and affinities is optional. However, in practise at least a default configuration of both of these features can be considered to be in place at the time of initial deployment.

The implicit default configuration of a GD deployment is as follows:

- 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.

4.7.2 Explicit Configuration

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

In practice, this would mean:

- 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.

This makes the whole configuration explicit and therefore comprehensible.

4.7.3 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 the Installation and System Administration sub-sections of the Server Clustering section, above, for background.

At time of writing, 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.

4.7.4 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 here. 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 arranged as follows:

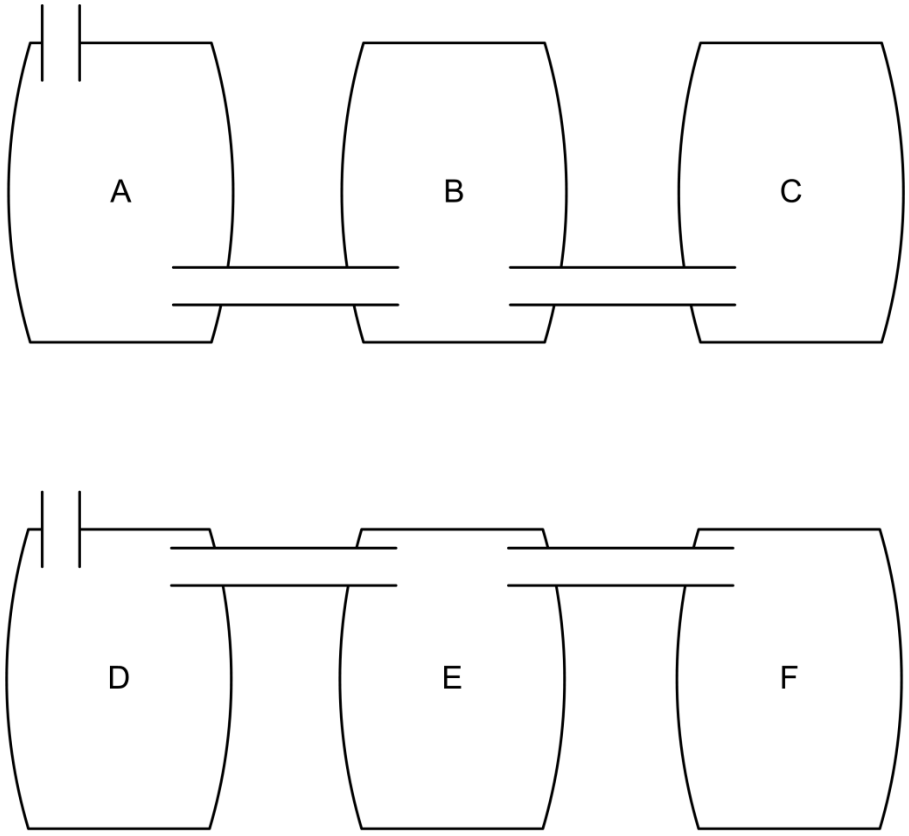
Rain water drains directly into one of the water butts.

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.

The arrangements are illustrated in the following diagram.



Brief: Water Butt Metaphor
Brief Server Clustering and Affinities ad hoc.vsd v3.02

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.

4.7.5 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 on-line 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.

5 Further 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 utilised with server clustering and affinities are in the technical documentation.

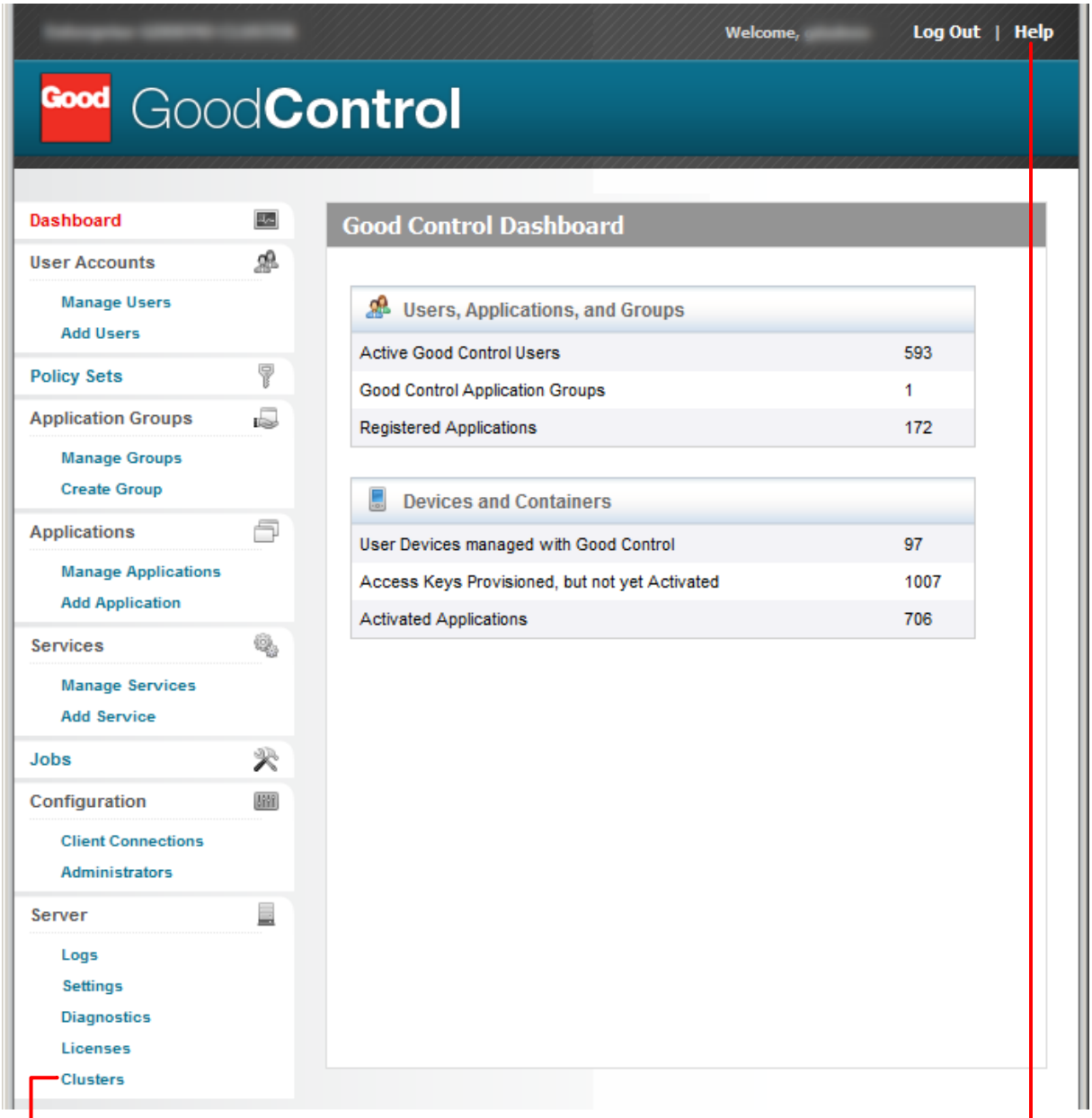
5.1 Configuration User Interfaces and Help

The user interface for configuring server clusters and affinities is part of the GC console. On-line help is available within the console, as it is for all GC functions.

The following screen captures shows the locations of these items.

5.1.1 Good Control Dashboard

Configuration of GD enterprise server clustering and affinities is accessed from the GC console dashboard. On-line help is accessible from the top-right hand corner of GC console screens.

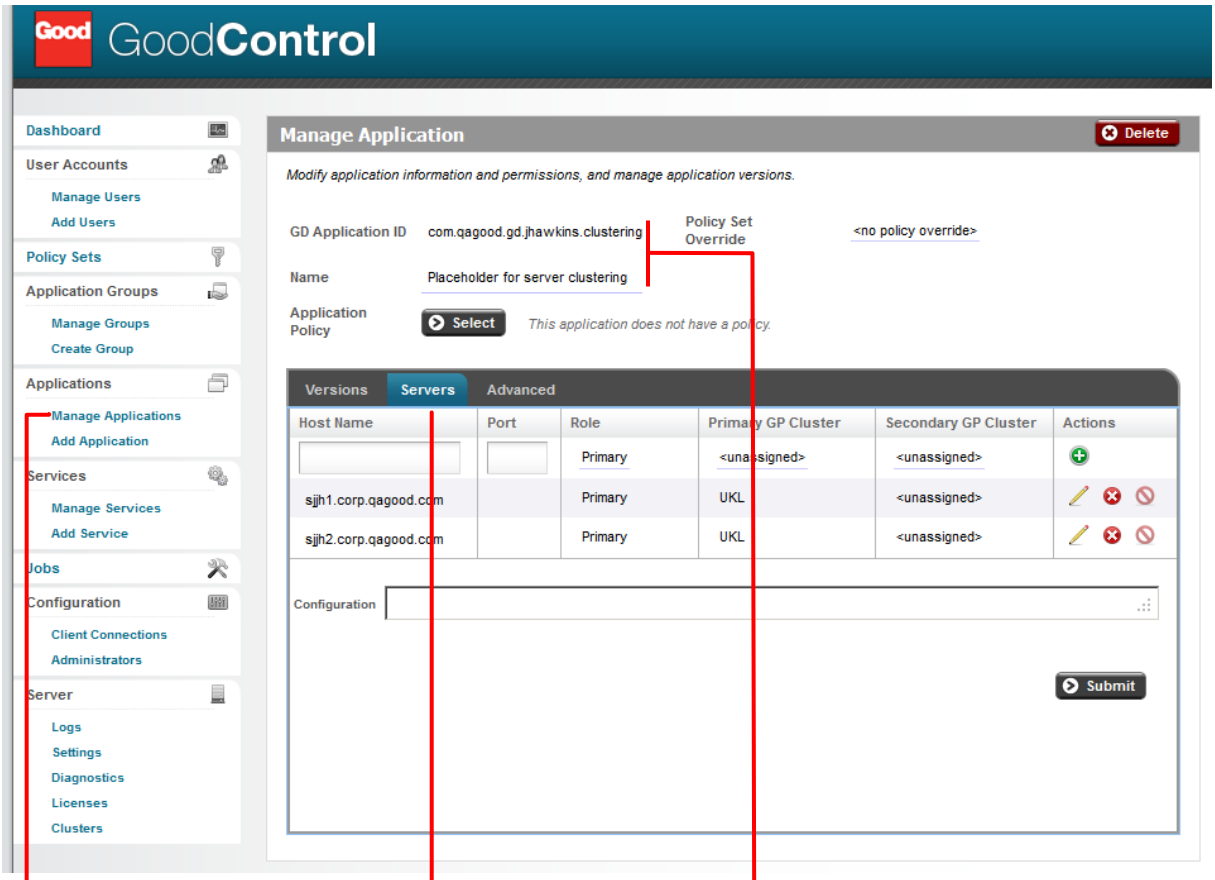


Configuration of:
GP clusters
GC clusters and affinities

On-line help
See under Server Settings & Configuration.

5.1.2 Good Control Application Management

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. The configuration user interface is on the Servers tab.



Manage Applications

Servers tab

Application name or ID would have been selected on the previous screen.

5.2 Technical Documentation

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

5.2.1 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.

> https://begood.good.com/view-doc.jspa?fileName=interface_g_di_o_s.html&docType=api#a77e4b4b68bb70a8072855c872918d2ec

Or, see the equivalent reference in the GDAndroid class documentation, if using the GD SDK for Android.

> https://begood.good.com/view-doc.jspa?fileName=classcom_1_1good_1_1gd_1_1_g_d_android.html&docType=android#aedeeab3604d3316fee1fda12cda56b8f

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

5.2.2 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.

> https://begood.good.com/view-doc.jspa?fileName=get_g_p_servers.html&docType=api

Or

> https://begood.good.com/view-doc.jspa?fileName=get_g_p_servers.html&docType=android

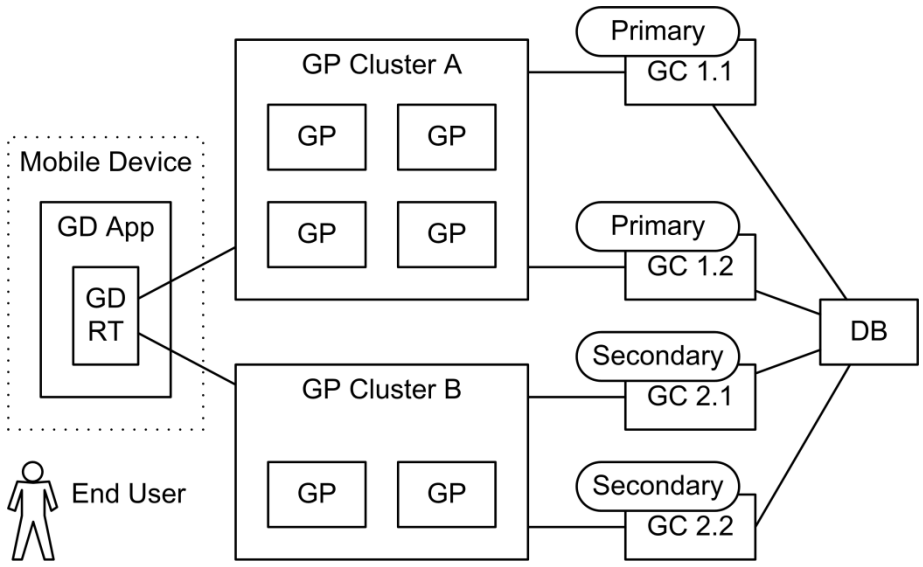
The same documentation is included in both places.

6 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.

6.1 Good Control Clustered Topology

The following diagram illustrates an example deployment for the purposes of discussion and explanation.



Appendix: Good Control Clustered Topology
Brief Server Clustering and Affinities ad hoc.vsd v3.02

The above 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.

The configuration can be represented as 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

6.1.1 Normal Operation

The normal operation of the GD runtime in the deployment shown above is 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.

6.1.2 Outage Operation

The operation of the application in the deployment shown above 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.

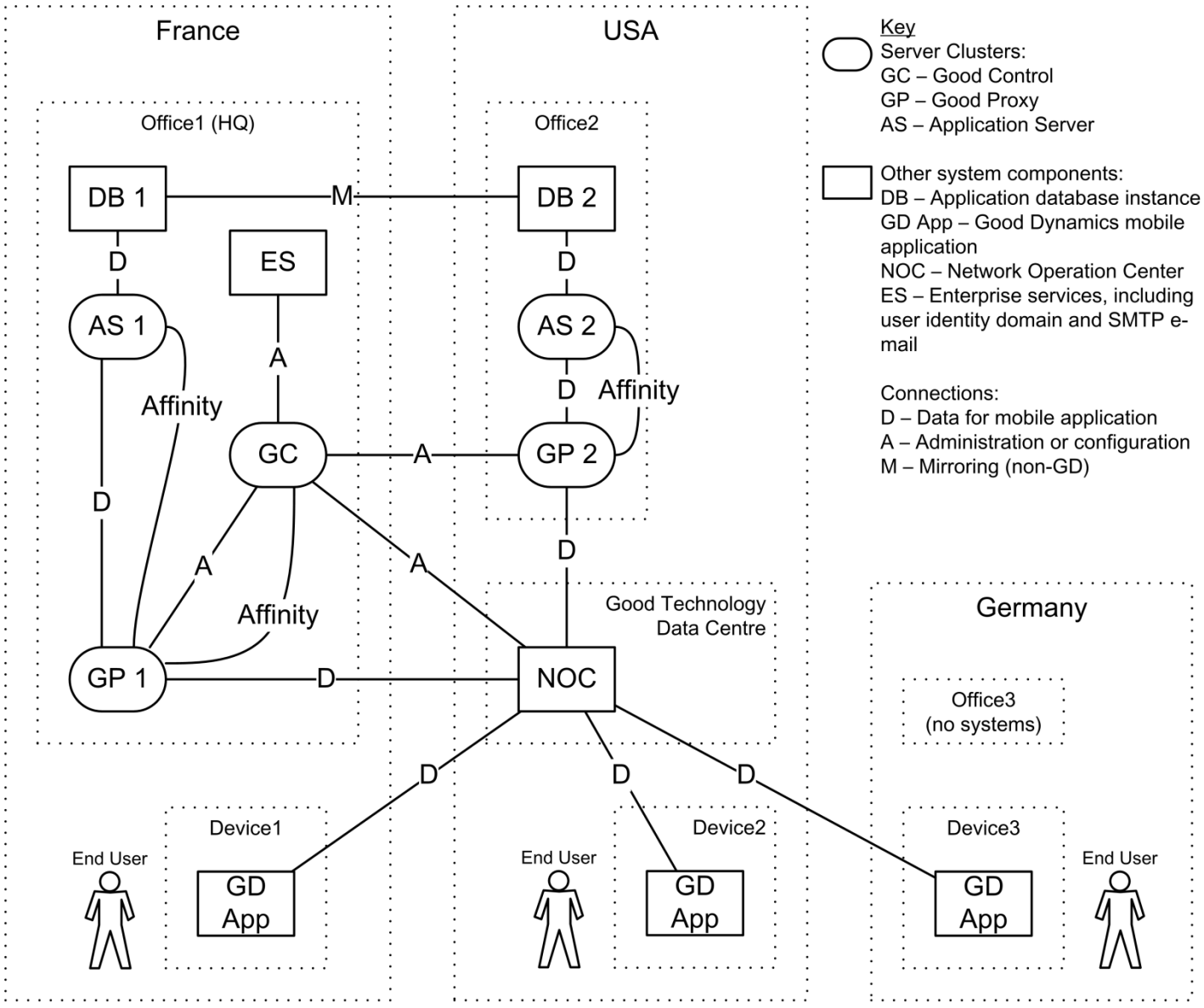
In the scenario that **all primary GC servers are down**, or appear to be, the GD runtime will connect to a secondary GC server, selected at random. The connection will go via one of the servers in GP Cluster B, also selected at random.

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 the Retry Primary sub-heading in the Server Clustering and Affinities section, above.

6.2 International Deployment with Sub-Clusters and Proximity

The following diagram illustrates an example deployment for the purposes of discussion and explanation.



Brief: International Deployment with Sub-Clusters and Proximity
Brief Server Clustering and Affinities ad hoc.vsd v3.03

The above diagram 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 that 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 the Resilience sub-sections, below.

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.

6.2.1 Configuration

The configuration in the above 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.

6.2.2 Normal Operation

The normal operation of the application in the deployment shown above 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.

6.2.3 Resilience to Cluster Failure

The operation of the application in the deployment shown above 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 the Retry Primary sub-heading in the Server Clustering and Affinities section, above.

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, below.

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.

6.2.4 Resilience to Site Failure

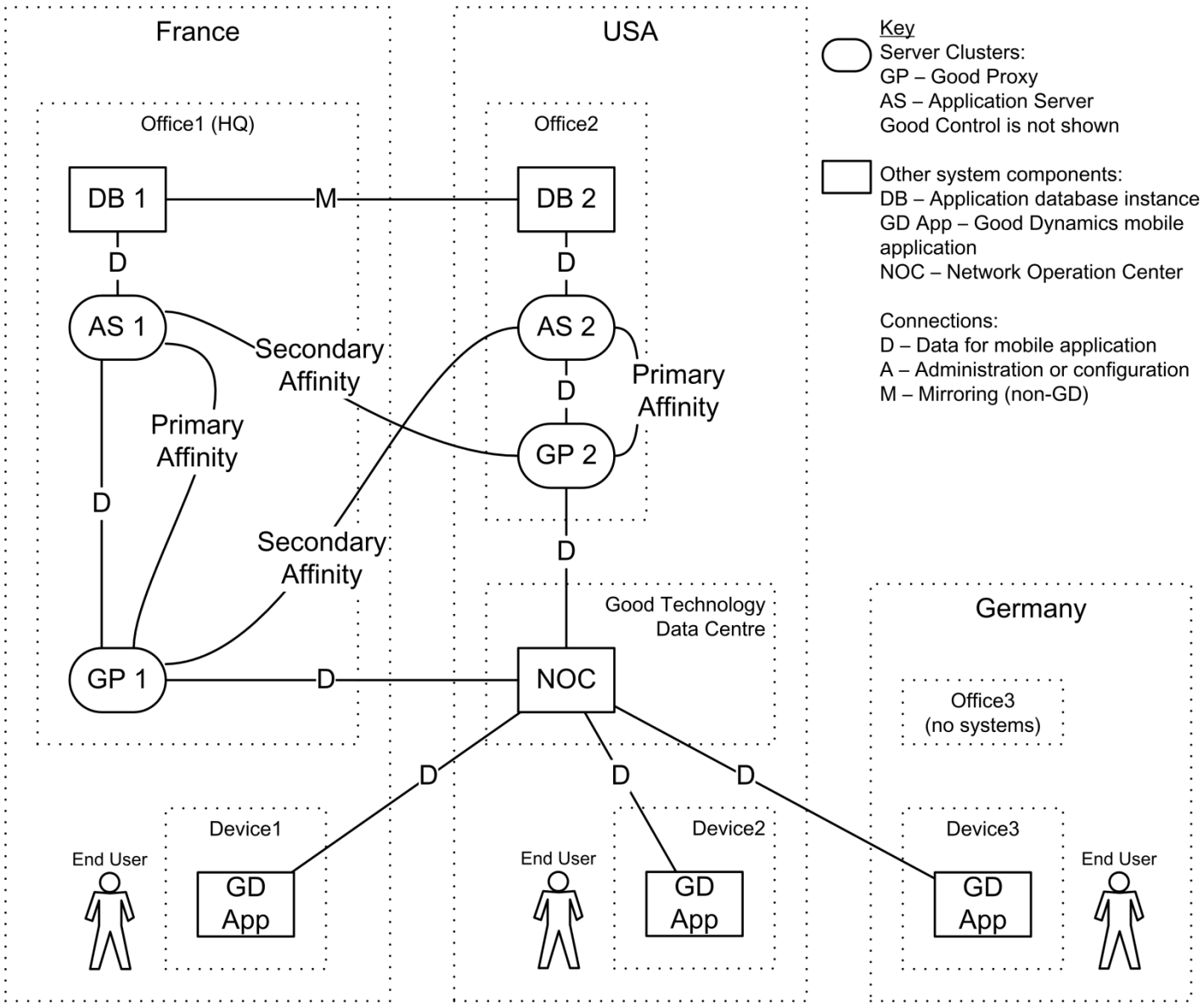
The operation of the application in the deployment shown above during a site outage is as follows:

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, see above. 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, see above. There would be no other immediate impact to the application, but see the GC cluster is down scenario, above, the results of which would apply.

6.3 International Deployment with Flexible Proximity

The following diagram illustrates an example deployment for the purposes of discussion and explanation.



Brief: International Deployment with Flexible Proximity
Brief Server Clustering and Affinities ad hoc.vsd v3.03

The above diagram shows some additional configuration of the deployment shown in the International Deployment with Sub-Clusters and Proximity example, above. See the previous section for description of its set-up. (The Good Control server and its connections are not shown in the above diagram, for simplicity.) 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 behaviour 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 that, 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.

6.3.1 Configuration

The configuration in the above deployment can be represented as 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.

Legal Notice

This document, as well as all accompanying documents for this product, is published by Visto Corporation dba Good Technology ("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.

Patents, Legal Information & Trademarks