When using the DGP Replica or Hybrid schema types, 1) each database is duplicated in multiple geographically distributed locations, and 2) each table in the database is treated as an append-only collection of immutable records. The asynchronous replication process runs constantly in the background to quickly merge the records created at each location into the corresponding tables of the other locations, creating an eventually consistent union of the records from all the locations (similar in concept to CRDT's). Replication from each source to each destination is configured and managed independently.

Each location functions autonomously, and does not attempt to verify the consistency of the data across other locations prior to writing new immutable records into its own local database. Instead, merge conflicts are corrected as they are detected during the replication and verification processes, and the state value of each record is adjusted accordingly.

Configuring Merge Replication

The ReplicaWork UI creates replication configuration records stored in the ReplicaWork table, which acts as a queue to control the iterative execution of the Replicate and Verify processes.

The replication and verify processes use exactly the same logic, but are scheduled differently. The steps of the process are:
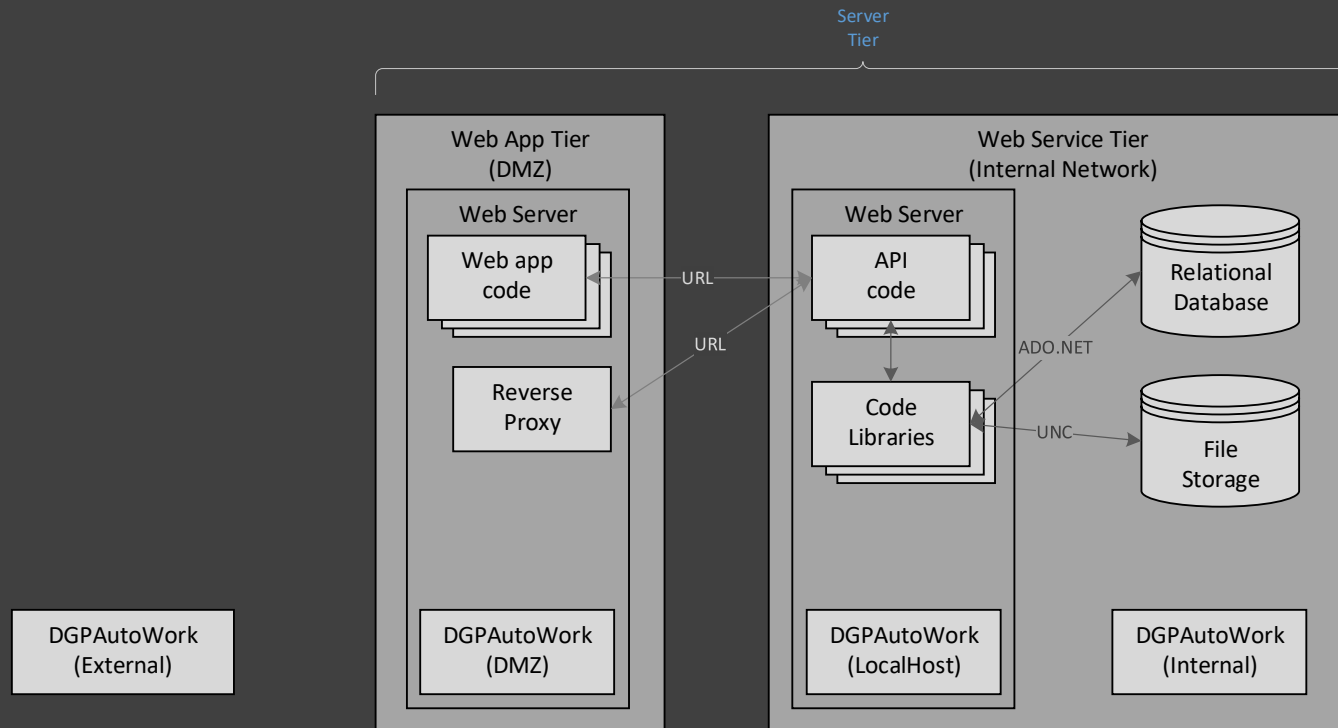
1. Call an API method at the source URL to query a source table for any records that are newer than the placeholder value.
2. If any source records are found, call an API method at the destination URL to merge the records into the destination table.
3. Update the queue record to schedule its next run and release it. If records were merged into the destination, also update the new placeholder value.

To create configuration records for replication, some information from various sources needs to be collected.

- The schemaname.tablename combination for the tables to be replicated (this is just a description label for the record)
- The name of the source database (a database name that is unique within the DGP system)
- The URL of the web service of the source table location
- The name of the method used to poll for records in the source table
- The URL of the web service of the destination table location
- The name of the method used to merge source records into the destination table

When creating new configuration records, the schemaname.tablename combinations and the names of the API methods for replication, count checks, etc. are documented in the API section, AutoWork API, Schema Tables. The URL's of the source and destination web services are determined by the network topology of the various locations of an environment for a given system, whether locations are public or private, and whether the replication process will be a push or a pull.

The difference between a push (local source, remote destination) and a pull (remote source, local destination) is mainly the determination of which location will have the configuration records. Push processes will be configured at the source, while pull processes will be configured at the destination.

Server Tier

Web App Tier (DMZ)

Web Server

Web app code

Reverse Proxy

DGPAutoWork (DMZ)

Web Service Tier (Internal Network)

Web Server

API code

URL

URL

Code Libraries

ADO.NET

UNC

Relational Database

File Storage

DGPAutoWork (LocalHost)

DGPAutoWork (Internal)

DGPAutoWork (External)

There are 4 different network areas within a DGP location.

- LocalHost runs on the same web server as the web service API's
- Internal runs on other computers in the internal network
- DMZ runs on the web servers the DMZ network (or dedicated computers)
- External runs on computers outside of the DGP location

*Important Note: the URL's used in the configuration records must be able to be resolved from wherever the DGPAutoWork app will be run. This will frequently include URL's that are only able to be called from the DMZ of a location, which means that testing those records remotely using the AutoWork test harness will not work correctly.

Field Name	Field Values	Description
Network	localhost, Internal, DMZ, External	View the diagram above to see the different network areas within a DGP location
SchemaType	REPLICA, HYBFILE, HYBREC	• REPLICA is the basic DGP schema that supports merge replication, and is the default value. • HYBFILE is a hybrid of a database table and a collection of shard file servers to store unstructured content (Lattice Filestore). • HYBREC is a hybrid of a database table and a collection of shard database servers to store large structured content (not used).
SchemaTable	SchemaName.TableName	The identifier label for the database name and table name

SrcDBName		The database name that is unique within an environment
WorkType	REPLICATE, VERIFY used in ReplicaWork queue	• REPLICATE is the standard merge replication process • VERIFY is merge replication rolled forward from the StartID to fix gaps of missing records (stops itself when it catches up to the main replication process)
Source URL		The URL used to call the API method to query for source records
Source Method		The name of the API method to call to query for source records
Dest URL		The URL used to call the API method to merge a batch of source records into the destination table
Dest Method		The name of the API method to call to merge source records into the destination table
StartID	64-bit integer	This value is used as the starting value of a process placeholder. It will generally be set to zero for most processes. However, large sets of records can be segmented using the StartID and FinalID fields.
FinalID	64-bit integer	This value sets the upper limit of a process placeholder for some processes. It will generally be set to zero, and is ignored in most cases, but when paired with the StartID, it can be used to break up very large numbers of records into smaller segments.
BatchSize	Int	The number of records to be used for iterative processes such as merge replication, but not for the detection processes.
IntervalMS	Int	The IntervalMS sets the delay between iterations of the process when it has "caught up" with the workload – whenever there is a backlog of work to be done, the interval is ignored and the process is scheduled to run immediately.
NextRun	64-bit integer	The Next Run value is the Unix Time in milliseconds that the process is scheduled to be run – initially it can be set to zero.
RunState	READY,	Setting the Run State value to READY is equivalent to turning a process on to begin execution.

Logging	ON, OFF	Logging set to "ON" will write details of each iteration of the automated process to the event viewer, and should only be turned on when debugging a process.
MaxDurMs	Int	The maximum allowed time in MS that an iteration of a process should take to complete. If an iteration processing time is longer than the maximum it will log an error.

Replicate vs. Verify Mode

Verify processes fix gaps in destination data by rerunning the replication process in parallel with the main REPLICATE process. There is no need to create Verify configuration records until they are needed. They are created with the "Clone" option of the context menu in the data grid, and then switching the work type to "Verify". The FinalID placeholder value should by default be set to zero, which means that the verify scan will run until it catches up with the most recent data, and then stop. The StartID can also be set to zero, which means it would roll replication forward from the first source record. Otherwise, a higher StartID value can be used to jump ahead to more recent data. The IntervalMS value is not used, so setting it to a default value of 1000 or 2000 is fine. Verify replication processes will automatically stop themselves when the placeholder value exceeds the FinalID value or the batch of source records is less than the maximum (which occurs when the VERIFY scan has reached the most recent records in the table).

Chain Replication

Hierarchical chain replication makes use of the fact that each replica table is a union of all the records from all the locations to use source records replicated into a destination table as a source for those records when the actual source is not accessible due to the network topology, security, etc. It is therefore polling a destination table for records that have been replicated to it from an original source (which itself is not accessible).

The configuration of chain replication is the same as regular replication, but will generally be a "pull" from a remote destination table acting as a source table. The only difference is the URL of the web service used to poll for source records, and the fact that those records will not be available until the replication from their original source has been completed. Chaining destination tables together as source tables can be as many layers deep as necessary to work within a network topology.
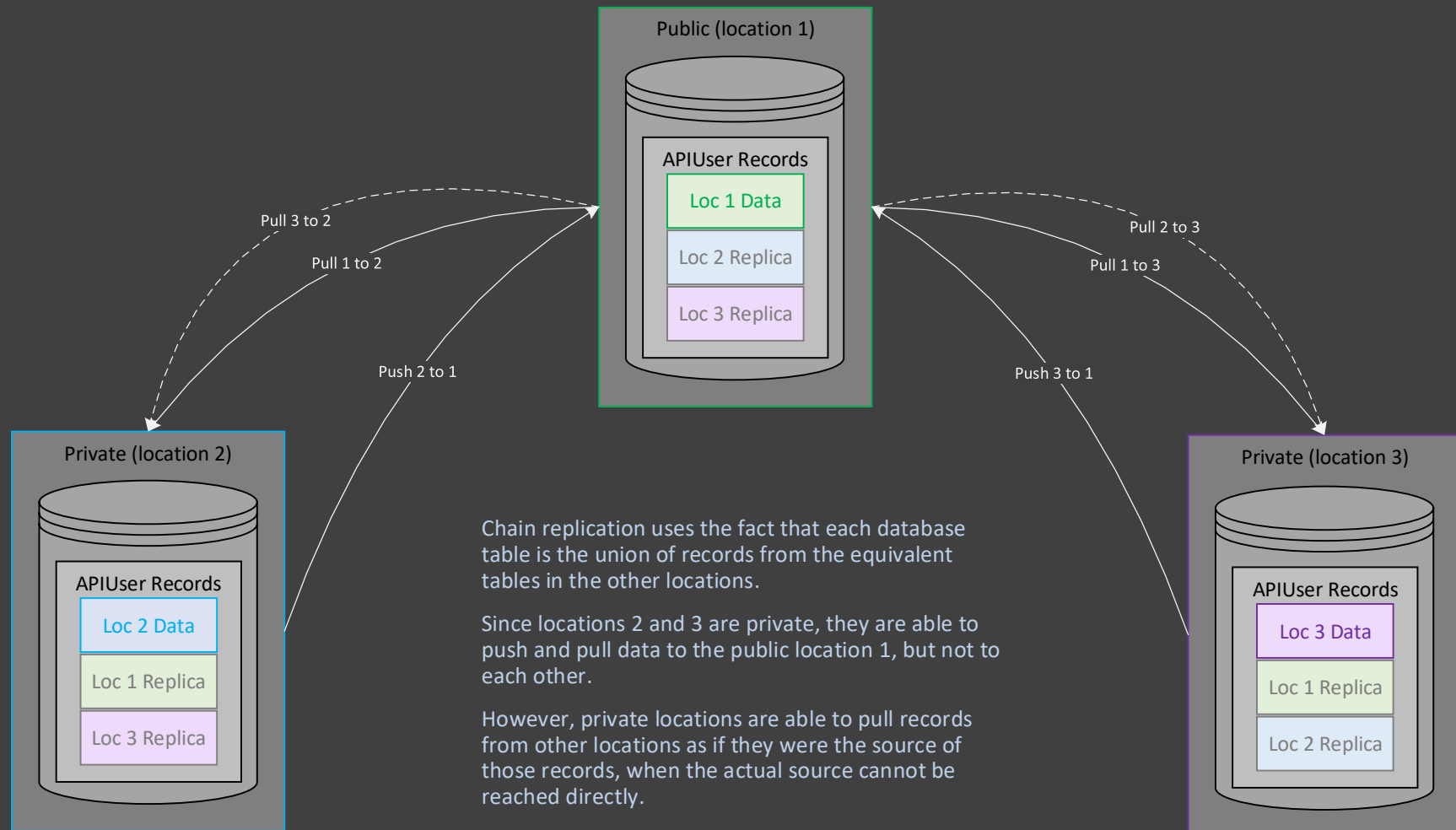
DGP Lattice Beta — □ ×

Connect Filestore User Security Configuration Testing Utility

ReplicaWork Queue *SchemaTable* [____] Search Clear [25 ▾] New |< < Page 1 of 2 > >|

SchemaType	SchemaTable	SrcDBName	WorkType	SrcURL	SrcMethod	DestURL	DestMethod	Sta
REPLICA	DGPFileStore.Favorites	DGPFileStore_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	Favorite.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	Favorite.Replicate.base	0
REPLICA	DGPFileStore.Favorites	DGPFileStore_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	Favorite.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	Favorite.Replicate.base	0
REPLICA	DGPFileStore.Files	DGPFileStore_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	File.GetSource.upload	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	File.Replicate.upload	0
REPLICA	DGPFileStore.Files	DGPFileStore_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	File.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	File.Replicate.base	0
REPLICA	DGPFileStore.Files	DGPFileStore_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	File.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	File.Replicate.base	0
REPLICA	DGPFileStore.Files	DGPFileStore_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	File.GetSource.upload	http://localhost/DGPWebSvc/DGPCntrl.aspx	File.Replicate.upload	0
REPLICA	DGPFileStore.FileTag	DGPFileStore_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	FileTag.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	FileTag.Replicate.base	0
REPLICA	DGPFileStore.FileTag	DGPFileStore_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	FileTag.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	FileTag.Replicate.base	0
REPLICA	DGPFileStore.Folders	DGPFileStore_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	Folder.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	Folder.Replicate.base	0
REPLICA	DGPFileStore.Folders	DGPFileStore_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	Folder.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	Folder.Replicate.base	0
REPLICA	DGPFileStore.Tags	DGPFileStore_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	Tag.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	Tag.Replicate.base	0
REPLICA	DGPFileStore.Tags	DGPFileStore_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	Tag.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	Tag.Replicate.base	0
REPLICA	DGPSysInfo.APIMethods	DGPSysInfo_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	APIMethod.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	APIMethod.Replicate.base	0
REPLICA	DGPSysInfo.APIMethods	DGPSysInfo_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	APIMethod.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	APIMethod.Replicate.base	0
REPLICA	DGPSysInfo.APIRoles	DGPSysInfo_WWin10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	APIRole.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	APIRole.Replicate.base	0
REPLICA	DGPSysInfo.APIRoles	DGPSysInfo_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	APIRole.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	APIRole.Replicate.base	0
REPLICA	DGPSysInfo.APIUser	DGPSysInfo_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	APIUser.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	APIUser.Replicate.base	0
REPLICA	DGPSysInfo.APIUser	DGPSysInfo_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	APIUser.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	APIUser.Replicate.base	0
REPLICA	DGPSysInfo.DataGroup	DGPSysInfo_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	DataGroup.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	DataGroup.Replicate.base	0
REPLICA	DGPSysInfo.DataGroup	DGPSysInfo_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	DataGroup.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	DataGroup.Replicate.base	0
REPLICA	DGPSysInfo.GroupUser	DGPSysInfo_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	GroupUser.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	GroupUser.Replicate.base	0
REPLICA	DGPSysInfo.GroupUser	DGPSysInfo_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	GroupUser.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	GroupUser.Replicate.base	0
REPLICA	DGPSysInfo.RoleMethod	DGPSysInfo_Win10Test	REPLICATE	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	RoleMethod.GetSource.base	http://localhost/DGPWebSvc/DGPCntrl.aspx	RoleMethod.Replicate.base	0
REPLICA	DGPSysInfo.RoleMethod	DGPSysInfo_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	RoleMethod.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	RoleMethod.Replicate.base	0
REPLICA	DGPSysInfo.RoleUser	DGPSysInfo_Win10Dev	REPLICATE	http://localhost/DGPWebSvc/DGPCntrl.aspx	RoleUser.GetSource.base	http://Win10Test/DGPWebSvc/DGPCntrl.aspx	RoleUser.Replicate.base	0

Methods: 2 | ClientMS: 4.54 | ServerMS: 3.27

In this diagram, the DGPAutoWork Windows services have been installed on the web app servers in the DMZ. Replication has been set up as a "push", from each location to the other locations, with a local query to the internal network to get source records, and a push of those records to be merged into the remote destination tables of the other locations.
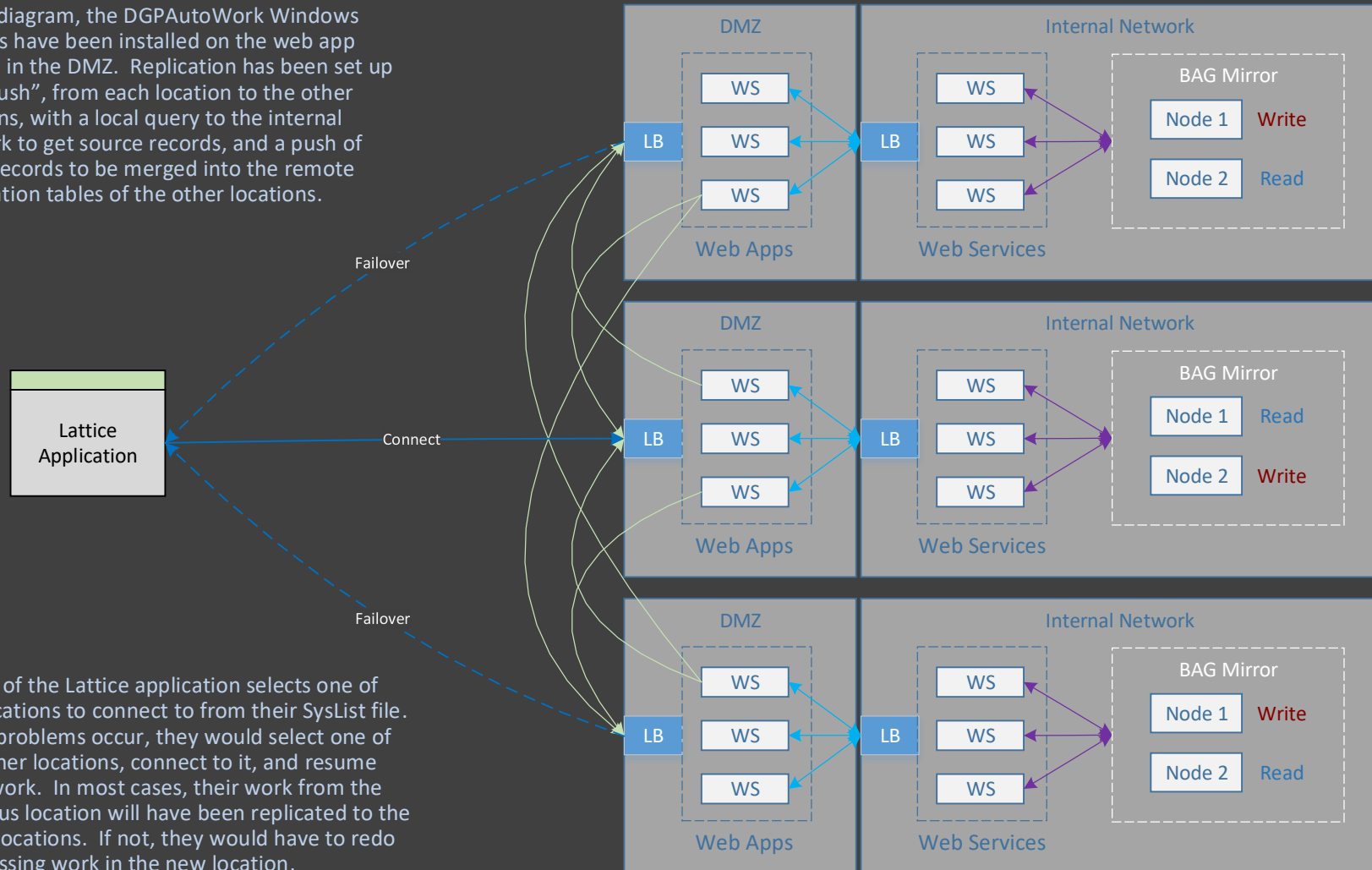
A user of the Lattice application selects one of the locations to connect to from their SysList file. If any problems occur, they would select one of the other locations, connect to it, and resume their work. In most cases, their work from the previous location will have been replicated to the other locations. If not, they would have to redo the missing work in the new location.

Lattice Application

Failover

Connect

Failover

DMZ

Web Apps

LB

WS

WS

WS

Internal Network

LB

WS

WS

WS

BAG Mirror

Node 1 — Write

Node 2 — Read

Web Services

DMZ

Web Apps

LB

WS

WS

WS

Internal Network

LB

WS

WS

WS

BAG Mirror

Node 1 — Read

Node 2 — Write

Web Services

DMZ

Web Apps

LB

WS

WS

WS

Internal Network

LB

WS

WS

WS

BAG Mirror

Node 1 — Write

Node 2 — Read

Web Services

Public (location 1)

APIUser Records

Loc 1 Data

Loc 2 Replica

Loc 3 Replica

Pull 3 to 2

Pull 1 to 2

Pull 2 to 3

Pull 1 to 3

Push 2 to 1

Push 3 to 1

Private (location 2)

APIUser Records

Loc 2 Data

Loc 1 Replica

Loc 3 Replica

Private (location 3)

APIUser Records

Loc 3 Data

Loc 1 Replica

Loc 2 Replica

Chain replication uses the fact that each database table is the union of records from the equivalent tables in the other locations.

Since locations 2 and 3 are private, they are able to push and pull data to the public location 1, but not to each other.

However, private locations are able to pull records from other locations as if they were the source of those records, when the actual source cannot be reached directly.

Configuring Data Repair Detection

The GeneralWork UI creates detection configuration records stored in the GeneralWork table, which acts as a queue to control the scheduled execution of the DupeCheck and CountCheck processes.

The steps of the CountCheck process are:

1. Call an API method at the destination URL to query for a count of all records that have been replicated from the source table, and the maximum src_id value.
2. Call an API method at the source URL to query for a count of all records in the source table less than the maximum src_id from the destination table. This eliminates discrepancies caused by replication lag.
3. Log the results of the count comparison to the AutoWorkLog table, and then update the queue record to schedule its next run and release it.

The DUPECHECK process calls an API method for each database table to query for any duplicate records. It only uses the "source" information, since there is no comparison to a destination.

To create configuration records for data repair detection, some information from various sources needs to be collected.

*   The schemaname.tablename combination
*   The name of the source database, which is used for several of the queries in the process (the name of the destination database is not needed, and is determined by the destination URL and destination method name used).
*   The URL of the web service used to get the source counts
*   The name of the method used to calculate the counts in the source table
*   The URL of the web service used to get the destination counts
*   The name of the method used to calculate the counts in the destination table

When creating new configuration records, the schemaname.tablename combinations and the names of the API methods for replication, count checks, etc. are documented in the API section API section, AutoWork API, Schema Tables. The URL's of the source and destination web services are determined by the network topology of the various locations of an environment for a given system.

The scheduling for data repair detection checks is different from the replication schedule in that it will generally be scheduled to run at a specific time each day, or on a specific day of the week rather than as constantly repeating iterations. The detection check processes will create log entries containing the results of the checks, and error log entries if any problems are found.

*Important Note: the URL's used in the configuration records must be able to be resolved from wherever the DGPAutoWork app will be run. This will frequently include URL's that are only able to be called from the DMZ of a location, which means that testing those records remotely using the AutoWork test harness will not work correctly.

Edit GeneralWork Configuration				✕
Global ID	81a42440c7ac4aa999586fed6c28c8b5	StartID*	0	
Network*	LOCALHOST	FinalID*	0	
SchemaTable*	SysInfo.APIMethod	Interval Type*	TIMEOFDAY	
SrcDBName*	DevWrk1VM_Dev_SysInfo	Interval Val*	120	
Work Type*	COUNTCHECK	Next Run	0	
Source URL*	http://localhost/DGPWebSvc/DGPCntrl.a	Run State*	STOPPED	
Source Method*	APIMethod.GetSrcCounts.base	Logging*	ON	
Dest URL*	http://Desktop2/DGPWebSvc/DGPCntrl.a	Max Duration*		
Dest Method*	APIMethod.GetDestCounts.base	Cancel	Delete	Save

Field Name	Field Values	Description
Network	localhost, Internal, DMZ, External	View the diagram above to see the different network areas within a DGP location
SchemaTable	SchemaName.TableName	The identifier label for the database name and table name
SrcDBName		The database name that is unique within an environment
WorkType	COUNTCHECK, DUPECHECK, used in GeneralWork queue	• COUNTCHECK runs queries to count source records in a destination table, and then run the same query in the source table to compare the number of records found • DUPECHECK queries a database table for duplicate records
Source URL		The URL used to call the API method to query for source records
Source Method		The name of the API method to call to query for source records
Dest URL		The URL used to call the API method to merge a batch of source records into the destination table
Dest Method		The name of the API method to call to merge source records into the destination table
StartID	64-bit integer	This value is used as the starting value of a process placeholder. It will generally be set to zero for most processes. However, large sets of records can be segmented using the StartID and FinalID fields.
FinalID	64-bit integer	This value sets the upper limit of a process placeholder for some processes. It will generally be set to zero, and is ignored in most cases, but when paired with the StartID, it can be used to break up very large numbers of records into smaller segments.
IntervalType	TimeOfDay, DayOfWeek	The value associated with the TOD type is the number of minutes offset from midnight on a 24 hour clock. The DOW value includes two comma-delimited parts. The first is an integer that represents the day of the week (with Sunday = 0) and the second is the same as a TOD value.
IntervalVal	Int	The value of the selected interval type for scheduling the nextrun time, used to calculate the NextRun value.

NextRun	64-bit integer	The Next Run value is the Unix Time in milliseconds that the process is scheduled to be run – initially it can be set to zero.
RunState	READY,	Setting the Run State value to READY is equivalent to turning a process on to begin execution.
Logging	ON, OFF	Logging set to "ON" will write details of each iteration of the automated process to the event viewer, and should only be turned on when debugging a process.
MaxDurMs	Int	The maximum allowed time in MS that an iteration of a process should take to complete. If an iteration processing time is longer than the maximum it will log an error.

The basic premise of in-place data repair is that asynchronous data replication in a distributed system built using collections of unreliable parts contains an almost infinite number of things that could go wrong and end up compromising the accuracy and/or integrity of the replicated data. Rather than trying to prevent each and every one of those almost infinite potential problems, the logic only focuses on the things that can be easily prevented (low hanging fruit), after which the emphasis is shifted to 1) quickly detecting problems in the data and 2) fixing those problems incrementally, in-place, and in the background. As problems occur, they are analyzed to see if they would be likely to occur again, and if so, whether or not there is a reasonable way to build logic to prevent subsequent occurrences. Otherwise, the emphasis remains on quick detection and repair.

The GeneralWork queue is used to manage the scheduling and execution of detection processes that are part of the In-Place Data Repair functionality in DGP systems. They consists of duplicate replica record checks, and count checks of the data in each source compared to the same count checks in each destination table.

Briefly, in-place data repair refers to the processes that first detect that a problem exists in the multiple copies of data distributed amongst the different locations, and then runs repair processes to correct those problems in-place and in the background, while people continue to use the system (as opposed to resetting the data with a snapshot or some other bulk operation). Gaps in replication can be fixed by running a VERIFY scan. Duplicate records require the intervention of administrators to decide which records to keep and which to remove.
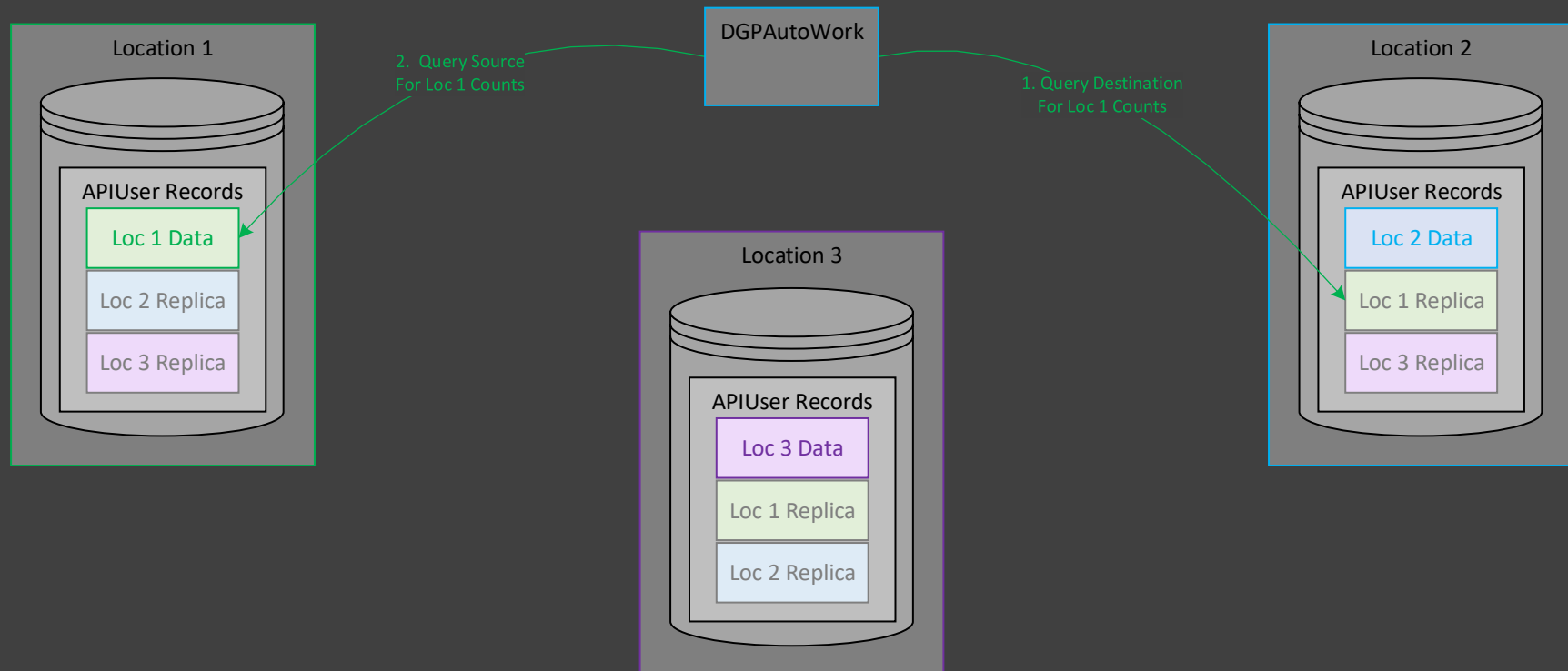
The count checks are the detection part of the in-place data repair processes. At the time the detection process is run, in the example above it starts by querying for the maximum src_id value for location 1 records in the location 2 database table. It then queries for the total count of records that are less or equal to than the maximum ID value, and the total count of active records.

It then runs all of these same count queries at the source, but uses the maximum src_id from the destination (not from the source) for the total rows and total active row counts. The difference between the maximum destination src_id and the maximum row_id of the source indicates the size of the replication lag. The row count values should be the same at the location and the destination. A lower count at the destination indicates missing replicated records, which is fixed by running a Verify scan. A higher count at the destination indicates some sort of duplication that must be investigated by admin staff.