DGP Applications

The deployment and maintenance of all of the tiers of a DGPDrive system is both standardized and greatly simplified using two concepts. The first is the XCOPY deployment option for .NET applications which is used to create a folder that contains the application assemblies and associated files, and the second is the immutable append-only pattern used so frequently in all parts of DGP systems. All parts of DGPDrive have been built with the full .NET Framework 4.8.x, and follow this same XCOPY immutable append-only deployment pattern.

The first step for client apps is to create a parent folder for the application on the client computer. Native client applications are built with the "release" option in Visual Studio or a build server (not published). This produces a streamlined assembly in the release folder of the solution that does not contain debug data. This release folder is then copied and renamed using the current date as its version number. Deployment of the app is done by copying the version folder under the parent application folder on the client computer.

The config file in the version folder is edited with values for the given environment. Frequently, the config file is simply copied from a previous version folder into the new version folder. Then a shortcut is created or edited to launch the "exe" (.NET assembly) in the new version folder. Rollbacks are accomplished by pointing the path of the shortcut back to the exe in the prior version folder, which remains unchanged (immutable) during the deployment process, and deleting the folder of the failed build.

DB Setup Utility

The DGP DB Setup utility is deployed like the other DGP applications by copying a version folder under a parent directory in local storage, and editing the .config file. After SQL Server has been installed and the empty databases have been created, the DB Setup utility builds and updates both schemas and core data in DGP systems. It is used during the initial setup of a system, and then periodically to append new elements to the schemas and "replicate" new core security data into the SysInfo tables. Some of the information created in the utility must be securely saved for periodic reuse by the DGP system.

Field Name	Field Values	Description
AppName	DBSetup	The name of the DGP app, used to identify log entries
EventSource	.NET Framework	The default event source to use for logging info to the Event Viewer if a custom event source has not been created
EventID	1000	The event ID to use when logging info to the default Event Viewer

Note: Since some changes to database schemas can interfere with the use of the database, DB Setup should generally only be run when a location is offline for maintenance.

Refer to the DBSetup documentation under the Client Tier heading for instructions on how to use the DBSetup utility.

DGPDrive

The DGPDrive application is a .NET WinForm native UI used to administer, configure, test and maintain a DGP system, and is also has an optional subsystem for collaborating and sharing files similar to OneDrive or DropBox. It is deployed like any of the other native DGP applications by copying a folder to local storage, editing the .config file, and editing the path to the new executable.

In addition, several subfolders should exist below each version folder. A folder named “Data” should exist as a subfolder of the version folder and contain a TestFiles subfolder. The TestFiles folder has multiple subfolders and contain all of the test files needed by the API Tester test harness to test all of the API methods in the Lattice web service. In addition, the Data folder also contains a sample system list file named SysList.xml. The SysList entries must be edited to match the endpoint information of the local system. Refer to the Lattice application documentation for more information about the system list files.

Once the system list files have been edited correctly, the next step is to connect to the web service of the new system. This is done in Lattice using the Connect form. Browse for the system list file, and open it. Select the system, the location, and the URL of the web service. Enter the system admin username and password, and click the Login button. If the system has been setup correctly, the Lattice app will be able to connect to the web service successfully.

Field Name	Field Values	Description
Network	LOCALHOST, INTERNAL, DMZ, EXTERNAL, OFF	The network area where the AutoWork Test harness application is running
AutoWorkLogging	ON, OFF	A flag value to turn logging for AutoWork processes on or off
AutoWorkMaxDurMS	50	The max duration for the logic to claim and process queue records (used by the AutoWorkTester test harness)
EventSource	.NET Framework	The default event source to use for logging info to the Event Viewer if a custom event source has not been created
EventID	1000	The event ID to use when logging info to the default Event Viewer
LocFilePath		Default path to root folder containing work directories, test files, etc.
TestFilePath		The default path to the local folder storing test files

Refer to the Lattice documentation under the Client Tier heading for instructions on how to use the application.

AutoWork App/Service

Refer to the AutoWork documentation for instructions on how to configure and use the windows service.

The DGP AutoWork application is a Windows Service that can also be run as a console app for debugging, etc. It is deployed like any of the other native applications (by copying a folder to local storage), with the added step of registering it as a Windows service using InstallUtil.exe. Information about the use of InstallUtil.exe to register and unregister the .NET application can be found in Microsoft documentation.

The AutoWork app/sevice will use the records in the ReplicaWork and GeneralWork tables to run iterations of the various automated DGP processes. Information about how to create these records are in the Configuration section of the Lattice app documentation. Also, information about testing and maintaining these configuration records are in the AutoWorkTester test harness section of the Lattice documentation.

DGPAutoWork App.config Keys

Web.Config Key	Sample Value	Description
SvcURL	http://localost/DGPWebSvc	The URL of the DGPWebSvc used by the AutoWork service to call the API methods for each type of work queue
SvcAcctName		The DGP system account to use when calling the web service API's
SvcAcctPword		The DGP system account password
ClaimID		The unique ID used by the AutoWork instance to claim queue records in the work tables
ReplicaWork	LocalHost	The network area where the AutoWork instance is deployed, and should only claim queue records for the same area
ReplicaWorkMS	Int	The number of MS between each scheduled ReplicaWork interval
ReplicaMaxBatch	Int	The maximum number of records to be claimed in each batch
GeneralWork	LocalHost	The network area where the AutoWork instance is deployed, and should only claim queue records for the same area
GeneralWorkMS	Int	The number of MS between each scheduled GeneralWork interval
GeneralMaxBatch	Int	The maximum number of records to be claimed in each batch
QueueCheck	LocalHost	Flag value that turns the QueueCheck timer functionality ON or OFF
QueueCheckMS	Int	The number of MS between each scheduled QueueCheck interval
ErrIntervalSec	LocalHost	The long error interval that slows down scheduled iterations without disabling the automated process

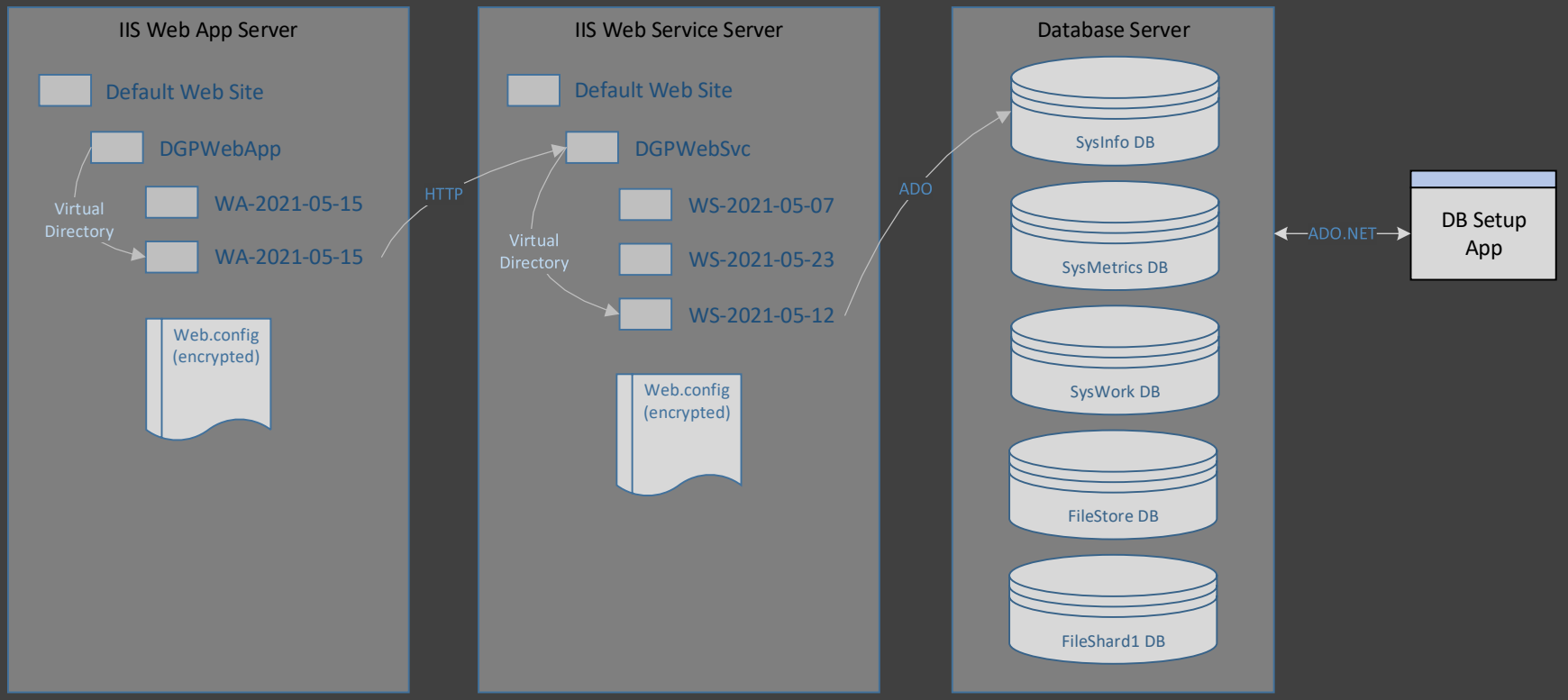
The values of the App.config file must be edited to match the location and network area in which the application is deployed.

Note: small systems installed on a single server should deploy more than one instance of the AutoWork application. If the system and locations grow to have more computers, then the number of instances of AutoWork running in the location should be adjusted.

Refer to the AutoWork documentation under the Client Tier heading for instructions on how to use the application, and the Configuration section under the Lattice application for information about creating configuration data.

Web Application Deployment

The process to deploy ASP.NET web apps and web services is identical to client applications. A parent folder is created on the web server below the appropriate web site. Web apps and web services are "published" in Visual Studio or a build server into a local folder. The publish process in Visual Studio allows for several pre-compilation (Ahead of Time) options. This folder is then copied and renamed using the current date as its version number.



New releases repeat this process, copying a new release folder under the parent folder of the web application. The .config file is edited in the new release folder. Frequently, the config file is simply copied from a previous version folder into the new version folder. Then the path to the virtual directory for the web app/service is then edited to point to the new version folder. To rollback a new version, the path to the virtual directory is pointed back to the prior version folder.

This is among the simplest mechanisms possible for ease of deployments and rollbacks. Each native app, web app and web service follow the immutable append-only pattern internally, which means that each new version contains all of the functionality of previous versions, plus some new functionality and/or bug fixes. This enables continuous system evolution and the CI/CD/CT processes without ever breaking backward compatibility for client applications and integrated systems.

Native client applications are configured to connect to the URL of either a reverse proxy in the web app tier, or directly to the URL of a web service in the web service tier. The endpoint URL's used by native applications are stored in app.config files and in system list files. Web applications store the endpoint URL of web services in their web.config file. Web services store ADO.NET connection strings, system account info, and encryption key info and file storage paths in their web.config files. Many of these values were created using the DB Setup utility when a system was first set up, and saved securely afterward.

The content of the web service web.config files can be encrypted using the aspnet_regiis.exe utility in the locations of selected environments. This utility can also be used to encrypt the app.config files of native apps whenever appropriate.

Web App and Web Service Configuration

DGPWebApp Web.config Keys

Web.Config Key	Sample Value	Description
LocState	ONLINE	Current state of the web app reverse proxy, which can be used to effectively disable a web service for new applications calling the Login method
SvcURL	http://localost/DGPWebSvc	The URL of the DGPWebSvc used by the web applications and reverse proxy pages (if applicable)

DGPWebSvc Web.config Keys

Web.Config Key	Sample Value	Description
SvcKeyVersion	SvcKeyV1	The label of the current encryption key
SvcKeyV1	(32-byte encryption key)	The value of the specified encryption key (maintains a list of current and all previous encryption keys)
LocState	ONLINE	Current state of the web service, which can be used to effectively disable a web service for new applications calling the Login method
System	DGP	The name of the system that owns the web service
Environment	Dev	The environment that owns the web service
Location	Win10Dev	The location that owns the web service
WebSvcName	DGPWebSvc	The name of the web service to be returned by the Login method
WebSvcVersion	2020-11-22	The date string of the web service version to be returned by the Login method
EventSource	.NET Runtime	The name of the event source to be used to log entries to the Event Viewer (the default value works if no custom event source has been created
EventID	1000	The event ID that works with the event source, when an event ID value is needed
TTLCheckFlag	ON	Turns TTL check on or off in the message processing pipeline
TTLMS	10000	The maximum MS allowed for the TTL check
UserCacheFlag	ON	Turns caching of the UserInfo object on or off in the message processing pipeline
UserCacheSec	600	How long the UserInfo object can be cached until it becomes obsolete and is removed
RateLimitFlag	OFF	Turns the rate limit check on or off
MaxMethBatch	10	Sets the maximum number of methods in a single API request message allowed by the message processing pipeline
MaxReqMsgKB	64	Sets the maximum size of an API request message allowed by the message processing pipeline

MaxRespMsgKB	64	Sets the maximum size of a response message allowed by the message processing pipeline
MaxFailedLogin	5	Sets the maximum number of failed authentications allowed by the message processing pipeline
PasswordLength	8	Sets the minimum password length allowed for password resets
ExpireDays	90	Sets the number of days until a password expires
MaxClaimBatch	5	Sets the maximum number of records that can be claimed by the AutoWork test harness
MaxFileSize	10000000	Sets the maximum size of a file in bytes that is allowed to be stored in Lattice
MaxSegSize	45000	Sets the maximum size of a file segment when uploading and downloading a file in Lattice
MaxFavorites	100	Sets the maximum number of favorites that are allowed per user in Lattice
SysInfo	ADO.NET connection string	The ADO.NET connection string for the SysInfo database
SysWork	ADO.NET connection string	The ADO.NET connection string for the SysWork database
SysMetrics	ADO.NET connection string	The ADO.NET connection string for the SysMetrics database
FileStore	ADO.NET connection string	The ADO.NET connection string for the FileStore database
FileShard1	ADO.NET connection string	The ADO.NET connection string for the FileShard1 database
TestDB1	ADO.NET connection string	The ADO.NET connection string for the TestDB1 database
TestDB2	ADO.NET connection string	The ADO.NET connection string for the TestDB2 database

Configuration files are used to store system and application “secrets” needed to initialize and run a DGP system. The aspnet_regiis.exe utility is used to encrypt to AppSettings section of the various .config files in the locations of environments whenever it is necessary for security. This type of encryption is used for DGP web apps and web services, but can also be used for client applications (like the AutoWork service, for example) when they contain potentially sensitive data.