PRODUCTION READY®

ELEFLEX®

ELEFLEX VERSION 3 SOFTWARE DESIGN SPECIFICATION

Version 1.1 - 1/7/2016

Abstract

This document is a software design specification for the PRODUCTION READY® ELEFLEX® Version 3 software platform. ELEFLEX® is an open source software platform for building modular, domain-driven, service-oriented applications and services. ELEFLEX® is intended to be used as a foundation to rapidly build and integrate multiple applications together using a services-based approach utilizing code generation, standards and governance to form a robust and scalable infrastructure.

Author

Danny R. Logsdon Jr.

http://www.ProductionReady.com

Revision History

Version	Date	Name	Description
1.0	12/15/2015	Danny Logsdon	Public release of document and website documentation
1.1	1/7/2016	Danny Logsdon	Added "Creating New Eleflex Module" section

Table of Contents

Introduction

Abstract

This document is a software design specification for the PRODUCTION READY® ELEFLEX® Version 3 software platform. ELEFLEX® is an open source software platform for building modular, domain-driven, service-oriented applications and services. ELEFLEX® is intended to be used as a foundation to rapidly build and integrate multiple applications together using a services-based approach utilizing code generation, standards and governance to form a robust and scalable infrastructure.

Scope

This document provides an initial high-level overview of the ELEFLEX® platform and later delves into the low-level aspects. It is intended for business executives as well as software development professionals. This document is accurate as of the ELEFLEX v3.2.1 release.

What is ELEFLEX®?

The ELEFLEX® software platform was created by Danny R. Logsdon Jr. and officially released to the public June 14, 2013, from Production Ready, LLC located at http://www.ProductionReady.com. ELEFLEX® is the culmination of several years of work and research in large scale, enterprise software

application design and is intended to be used as a foundation to rapidly build service oriented architecture (SOA) applications, with the ultimate goal of standardization to form a robust and scalable infrastructure. The ELEFLEX® software platform consists of an application software framework built using Microsoft® .NET C# that provides a standardized, reusable set of libraries, models and tools that enable software developers to rapidly develop applications.

The ELEFLEX® software platform was developed with the following guiding principle and is also how it received its name:

"Elegance in Simplicity, Flexibility in Design"

This implies easily comprehensible components, structure and usage that is flexible to evolve to changing technology and business needs over time gracefully. ELEFLEX®, as it has been developed, will be an ever-evolving refactor using the best of breed open source components to solve complex business problems in an elegant way.

Software Licensing

Visit http://www.ProductionReady.com/About to learn more about Production Ready and for licensing information.

Free and Open Source Software Licensing

ELEFLEX® is free open source software and is released under the GNU General Public License, included in this document. The complete source code can be downloaded from GitHub located at https://github.com/ProductionReady/Eleflex

References

This document makes references to the PRODUCTION READY® website located at http://www.ProductionReady.com as well as the following documents on the website or included as artifacts in the ELEFLEX® downloadable distribution.

Name	Location
Production Ready Website	http://www.ProductionReady.com/
GNU General Public License v3	http://www.ProductionReady.com/GPL

Requirements

Overview

The goal of this software platform is to provide a foundation to easily design, develop and deploy applications subscribing to a service oriented architecture (SOA). Building service-based applications promote reuse, can easily be modified for customized infrastructures and are scalable.

Services are loosely coupled applications that expose functionality for a domain area. Service commands provide single units of work via a formally defined, stateless operations through its interface. Once a reusable set of services are instituted, orchestrations, workflows, composite services and ESB architectures can then be created to realize the full power of integrating and automating processes over multiple service boundaries.

The platform also utilizes code generation techniques to allow developers to quickly build applications. Software code generation reduces repetitive typing, eliminates common coding mistakes, misspellings and provides standardization to coding guidelines, policies and governance. Using this software platform as a foundation can provide the standardization and intrinsic interoperability needed when building large scale applications and services for an infrastructure.

The core concepts and design considerations for the software platform are programming language independent and each language that implements the foundation may utilize different constructs and methods to accomplish the same architecture goals.

The Requirements have been separated into four categories:

- Organization Requirements
  - These requirements are based upon organizational and strategic goals.
- Architecture Requirements
  - These requirements impact the overall design and implementation of the software platform and are shared throughout all applications.
- Module Requirements
  - These requirements represent a domain or section of an application built to solve a business problem.
- Feature Requirements
  - These requirements effect a Module and are areas of functionality or business logic that must be supported.

The following sections convey the design considerations and decisions needed to fulfil each of the software requirements as specified.

Organization Requirements

RO1 Free Software

Requirement

The software platform must be licensed as free, open source software.

Background

The philosophy of Production Ready resembles that of the Free Software Foundation. We hope to provide a software platform that will be useful to the community using software development best practices, patterns and tools to speed development implementing high quality solutions.

Solution

This software platform will be released under the GNU General Public License version 3.

Licensing information has been added to the project and the source code has been released, currently available on GitHub at https://GitHub.com/ProductionReady/Eleflex.

Architecture Requirements

RA1 Modern Application Architecture

Requirement

Design a modern application architecture with the latest software development principles, patterns and practices using the best of breed open source projects.

Background

Production Ready® is routinely asked to help develop, enhance or fix existing applications for our customers. Many of these applications are poorly written with no clear separation of concerns, have security issues, use older technology that is outdated, or not upgradeable.

Our goal is to produce a modern application architecture and framework template that can be used as a starting point to produce high quality software for the community. This software platform will additionally become the application infrastructure for the company and the basis for new applications developed in the future.

Solution

ELEFLEX® is an open source software platform for building modular, domain-driven, service-oriented applications and services. ELEFLEX® is intended to be used as a foundation to rapidly build and integrate multiple applications together using a services-based approach utilizing code generation, standards and governance to form a robust and scalable infrastructure.

To promote community collaboration, NuGet was selected to be used as the primary method to publish and integrate with the platform. NuGet allows installation of open source projects, source code, assemblies, files and custom logic to help configure with a project while the ELEFLEX® platform provides the framework code foundation to quickly develop modules.

Several open source components have been selected to provide structure to the application for dependency injection, inversion of control, object mapping, service location, auditing, user interface and more. These will be discussed in later sections.

RA2 Service Oriented Architecture

Requirement

Design a service oriented architecture with a standardized framework of services that can be easily expanded upon for new application modules.

Background

A service oriented architecture helps building large scale software easier. An application can be divided up into domains, with each domain exposing self-contained business activities. These stateless operations are easy to invoke over a network and are easily testable. The ELEFLEX® platform's governance has ensured that modules developed have a solid foundation to build from. Furthermore, these can be customized to realize other organizational requirements, processes and goals.

Solution

The ELEFLEX® software platform was developed utilizing Windows Communication Foundation (WCF) to provide the primary communication and security. Services exposed from the platform will operate on a request/response paradigm, with each method exposing a unique request object and a response object. The request object will be used to route to a configured service containing the business logic to process it. This allows new modules to be developed and integrated within the service pipeline.

RA3 Code Generation

Requirement

Design must incorporate code generation to quickly develop modules in the platform.

Background

The platform must be easy to integrate with and be able to quickly get new functionality up and running. Code generation is used to quickly create code files required by the platform's architectural layers. Using template-based techniques, templates can be customized and code re-generated to add or remove functionality. Templates can change to reflect coding guidelines and standards, reduce remediation, common coding mistakes and increases intrinsic interoperability.

Solution

Pre-processing of metadata to create individual code blocks, files or complete solutions will be required to reduce the burden of developing the number of code files needed to support the

platforms architectural layers. The platforms currently make use of T4 text templates to help produce code files required, although other tools are in development as well.

RA4 End to End Solution

Requirement

Design a templated solution that encompasses all layers of the platform to create a working application with minimal metadata.

Background

In order for the platform to be useful, developers must be able to build applications and application modules quickly. Code generation will be used to create a standardized, templated approach to building code files needed for the platform's architectural layers.

Solution

The ELEFLEX® platform has been developed to provide interfaces, objects and business logic functionality that are used in the various layers of the n-tiered application design. A complete, end-to-end solution is generated utilizing NuGet, as well as integrating other modules. T4 text templates are used to generate files needed to create a custom module.

RA5 Microsoft .NET C#

Requirement

Implementation of the design must be developed using Microsoft® .NET C#.

Background

The Microsoft® platform is one of the largest software development technologies available and aligns with organizational priorities and personnel.

Solution

The inherent design of the platform is language independent, however each language may use different constructs and techniques to accomplish the same task. Microsoft® .NET C# will be used as the default implementation of the software platform.

RA6 Microsoft SQL Server and Azure

Requirement

Design must support Microsoft SQL Server and Microsoft Azure database engine for data persistence, however it should provide the ability to incorporate other database engines.

Background

Hosting of the platform should be able to be deployed on an internal environment or one in the cloud with minimal changes.

Solution

We provide a clear separation of model and storage interfaces that give us the ability to target applications to differing backend storage databases as needed. By default, the platform supports Microsoft SQL Server as well as Microsoft Azure database engines out of the box.

RA7 Dynamic SOA Data Manipulation

Requirement

Design a standardized way to dynamically query and modify data within the SOA design.

Background

This requirement is to ensure the platform provides a standardized operations that allow for querying and manipulation of model data over service boundaries.

Solution

The ELEFLEX® software platform provides a canonical model that enables dynamic data manipulation over service boundaries. The *StorageQueryBuilder* object provides the implementation of the foundation data contract operations used for dynamic data manipulation and querying functionality. Additionally, Web API or other RESTful services can be added on top of the SOA model to expose information in their formats.

Module and Feature Requirements

RM1 Security

Requirement

Incorporate a security mechanism to authenticate and authorize users in the system.

Background

Applications developed on the platform should be secure, providing adequate protection for both users and for hosted services.

Solution

The ELEFLEX® platform utilizes Microsoft ASP .NET Identity and OWIN as the default security mechanism. This is integrated into the service command execution pipeline that allows authenticating and authorizing during the request lifetime.

RM1F1 Users, Roles and Permissions

Requirement

The security mechanism must allow for user, role and permission entities.

Background

Existing applications that will be upgrade to the ELEFLEX® platform currently use a notion of users, roles, and permissions. In some cases, a large effort was made to create security matrixes that allow an application to fully secure itself.

Solution

The ASP.NET Identity security mechanism has dropped support for permissions and created a new entity for claims. We have expanded the identity model to additionally keep track of permissions, adding them along with roles in the new model. This allows transparent use with current security checking attributes that use roles, but can also be expanded with custom objects.

RM1F2 Effective Date Enforcement

Requirement

The security mechanism must allow for effective dates to determine when a user assigned role or permission is available.

Background

Existing applications that will be upgrade to the ELEFLEX® platform currently use this notion of effective dates on assignments.

Solution

The ASP.NET Identity security mechanism does not have support for effective date assignments. The platform has changed the underlying identity persistence stores to additionally keep track of effective dates and only return those items that are currently valid.

RM1F3 Dependent Role Hierarchy

Requirement

The security mechanism must allow for roles to be assigned to other roles, so that when one role is assigned to a user, any child roles of that role are automatically applied.

Background

Existing applications that will be upgrade to the ELEFLEX® platform currently use this notion of applying dependent roles based on an assigned role.

Solution

The ASP.NET Identity security mechanism does not have support for parent/child dependencies. The platform has changed the underlying identity persistence stores to additionally keep track of dependencies and return child roles of the user's assigned roles.

RM2 Logging

Requirement

Incorporate a logging mechanism into the design to store errors, audits and warning messages.

Background

Logging provides application health and monitoring. Storing this information provides developers the ability to help discover application defects, execution and processing debugging, warnings and other related information.

Solution

The platform exposes a logging service to store application messages during the execution of the application. A service contract is provided that allows sending and exposing message information while providing a centralized storage mechanism for all system services.

Developers access application logging my accessing the ***Eleflex.Logger*** object. Calling the Current property allows execution of error, debug, warning, info, and fatal methods with standard overloads.

RM2F1 Required Data

Requirement

The logging mechanism must allow storage of data:

- Create date (UTC)
- Application name
  - Multiple applications may be using centralized logging and this will allow narrowing search results for specific application messages.
- Server name
  - Multiple servers or clients may be sending messages and this will allow narrowing search results for a specific machine messages.
- IsError
  - Overall, determine if this message is an error and should be reviewed.
- Severity
  - This will be one of the methods that the developer will call on the logger, such as error, warning, info, etc.
- Message
  - Developer message describing current process execution or variables in question
- Exception

- The exception message generated from the .NET runtime.

Background

Including the data mentioned above will provide greater monitoring and debugging of application health within the infrastructure.

Solution

The logging mechanism includes these data fields on the ***LogMessage*** object and messages can be sent to the application log using the ***Eleflex.Logger*** object.

RM3 Versioning

Requirement

Incorporate a versioning mechanism into the design to allow modules to self-upgrade after upgrading their NuGet package.

Background

Modules will often require data persistence, usually with a database engine. Often the schema or data may change between released versions of software. This versioning module will provide a path so the module can upgrade itself.

Solution

The platform exposes a versioning service to store an installed module's version information. On application startup, the system will dynamically load all the module's upgrade patches, determine the execution tree for patches and execute each in order.

Design

Development Standards and Naming Conventions

The software platform has been developed in conjunction with the *Production Ready Programming Standards* document to ensure standardized language syntax conventions, best practices and naming conventions.

N-Tiered Architecture

The following diagram depicts the high-level view of the ELEFLEX® N-Tiered solution architecture separated into six layers that will constitute an end to end solution to support the architecture requirements. Each tier represents an assembly (or set of assemblies) that comprise the solution and arrows show the relationship where an assembly references another assembly or service communication occurs (loosely coupled via inter-process communication using WCF).

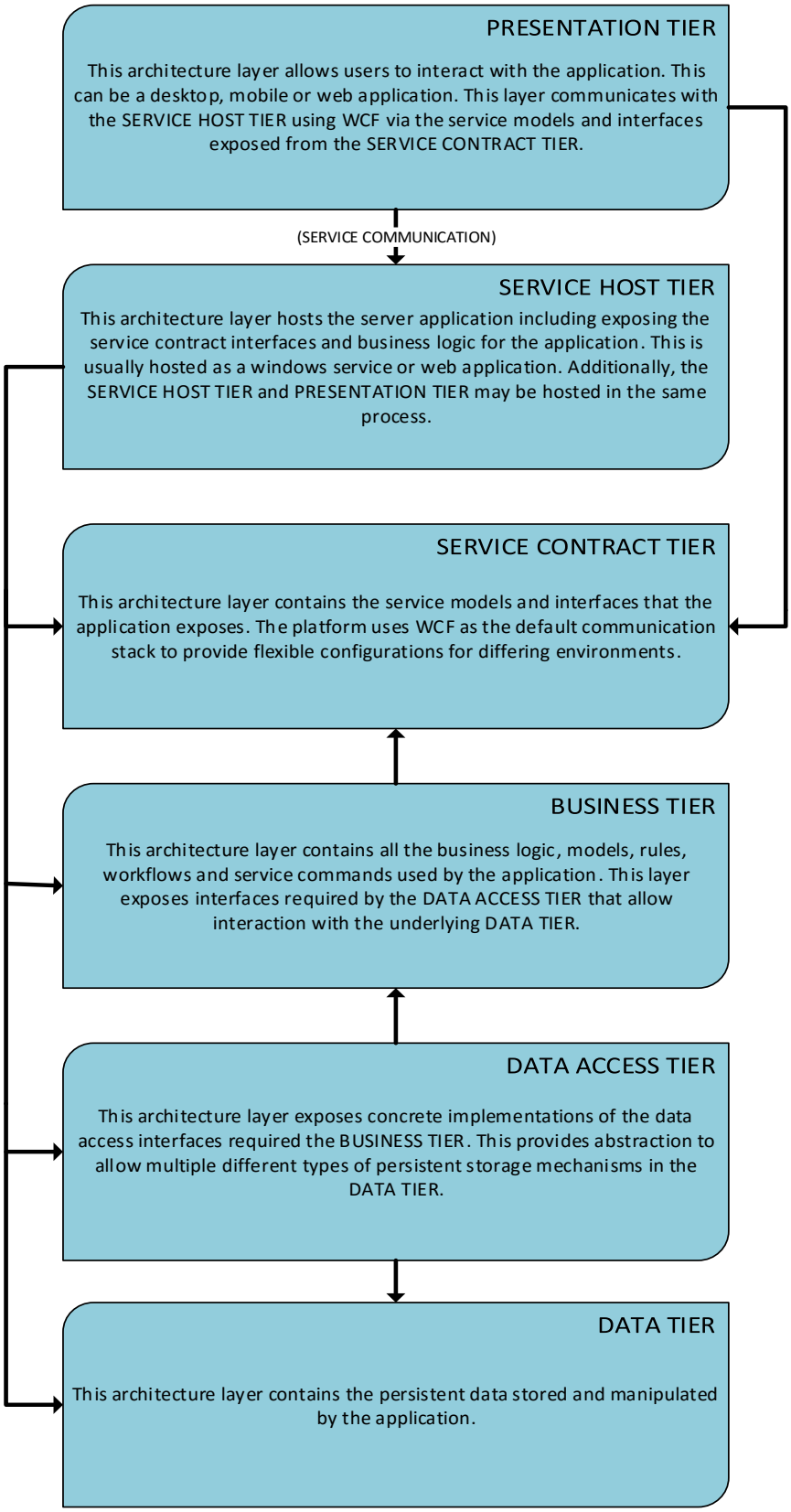

FIGURE 1- ELEFLEX N-TIERED HIGH LEVEL OVERVIEW

The following diagram depicts the component-level overview of the ELEFLEX® N-Tiered solution.

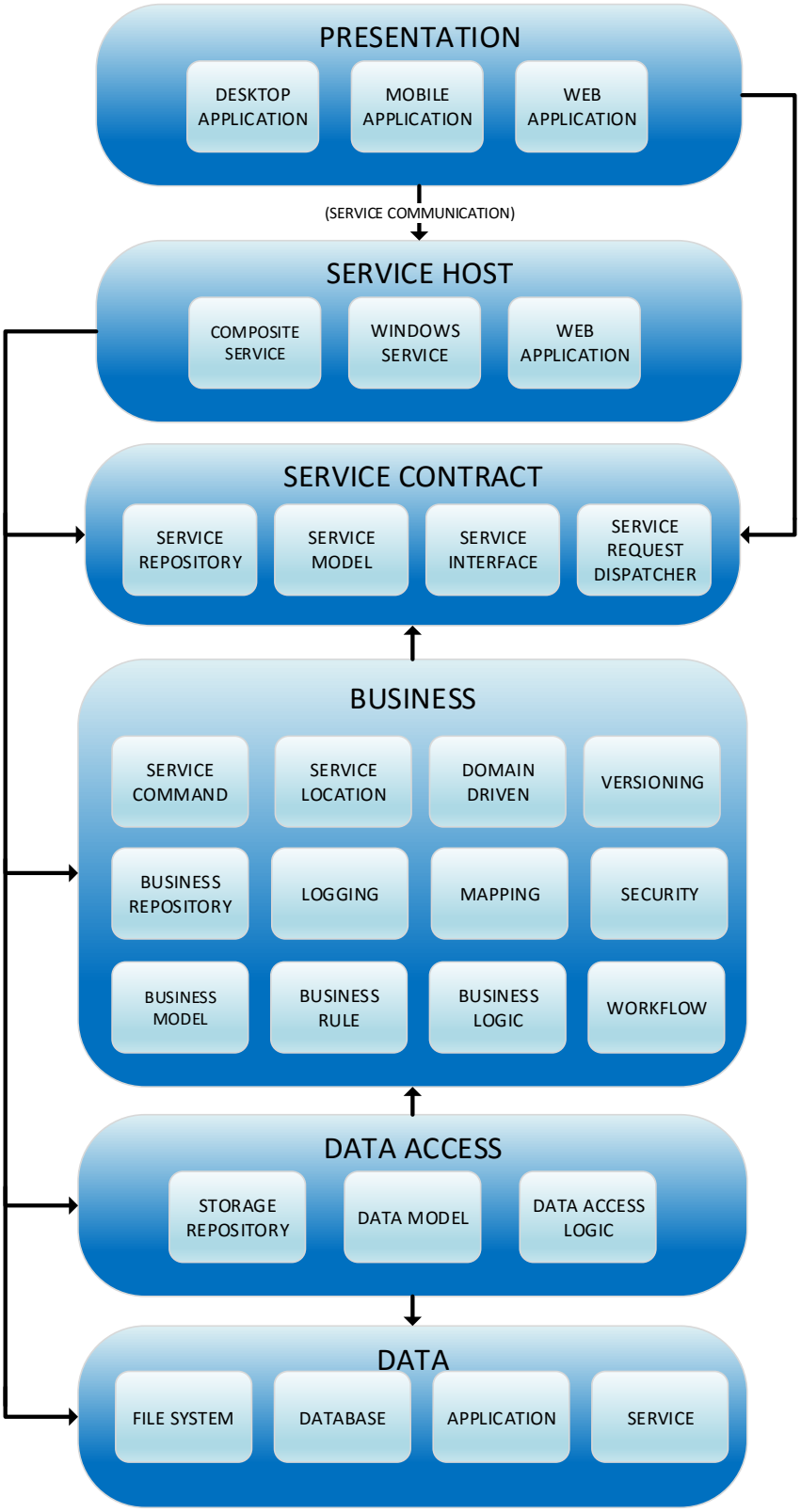

FIGURE 2- ELEFLEX N-TIERED COMPONENT OVERVIEW

System Startup, Startup Tasks and Registration Tasks

The platform provides an entry point for handling the complicated tasks of starting up, registering, configuring, updating and shutting down the system. Creating an instance of the ***Eleflex.SystemStartupShutdown*** object and calling the *Start()* or *Stop()* methods will perform system initialization or termination. These methods would be called in the *Application_Start()* or *Application_End()* methods of a web application's global.asax file or executed within the *Main()* method of a windows application or service. An example of calling the methods would be the following:

```
//Start the system
Eleflex.SystemStartupShutdown startup = new Eleflex.SystemStartupShutdown();
startup.Start(null);

//Shutdown the system
Eleflex.SystemStartupShutdown shutdown = new Eleflex.SystemStartupShutdown();
shutdown.Stop(null);
```

Using a convention over configuration approach similar to the open source project ***Bootstrapper***, the system defines an ordered approach to run various startup tasks, as executed by the ***Eleflex.SystemStartupShutdown*** object. The process will automatically find all objects inheriting from either ***Eleflex.StartupTask*** or ***Eleflex.StartupTaskWithRegistration*** and execute the tasks in order. The system contains the following pre-defined ordered tasks required by the system:

- Object Location – This startup task allows registering StructureMap configurations
- Mapping – This startup task allows registering AutoMapper configurations.
- Business Rules – This startup task allows registering business rules and events.
- Service Communication – This startup task allows registering WCF service commands.
- System Patching – This startup task dynamically loads all registered components and executes update logic for patching an installation.
- Logging – This startup task is used to register system logging.

The following example demonstrates how to create a custom startup task. Integrators creating a new startup task can order their task to execute before or after other tasks by setting the ***Priority*** property. See the ***StartupConstants*** object to get a list of predefined startup order values that can be used.

```
using Eleflex;

public partial class ExampleStartupTask : StartupTask
{

        public ExampleStartupTask() : base()
        {
            Description = @"Example startup task.";
            Priority = StartupConstants.PRIORITY_CUSTOM;
        }

         public override bool Start(ITaskOptions taskOptions)
         {

             //Execute custom logic here!

             return base.Start(taskOptions);
         }

}
```

Integrators using the ***Eleflex.StartupTaskWithRegistration*** object must also define a custom class attribute that registration tasks must decorate with so they can be dynamically discovered and executed when the corresponding startup task executes. The usage is demonstrated when registering object location and object mapping configurations in the next sections.

Inversion of Control, Dependency Injection and Service Location

The ELEFLEX® platform uses the open source component ***StructureMap*** for inversion of control and dependency injection. This system component helps to keep modular software designs maintainable by dynamically allowing configuration and resolution of objects and their dependencies. The system also incorporates the open source component ***CommonServiceLocator*** for service locator to resolve objects as other integrated open source components of the system rely on this.

The platform provides a static entry-point object, ***Eleflex.ObjectLocator,*** which is consumed throughout the various layers of the application for dynamic object resolution. This object contains two properties:

- ***Container*** (object)
    - This property holds the core ***StructureMap*** container object for the application. This property should ideally only be accessed when registering configurations at startup.
- ***Current*** (IObjectLocatorService)
    - This property holds the object locator instance used for resolving objects at runtime. This object and property is used throughout the application to resolve object construction and dependencies. An internal startup task takes care of setting this property on system startup.

Object location registration takes place when the system is startup (discussed in the next sections). Integrators can register configurations for object location by inheriting from the ***Eleflex.RegistrationTask*** object and adding the [***ObjectLocationRegistrationTask***] class attribute. The following examples demonstrates how to create a class used to register any object location configurations you define.

```
using Eleflex;

[ObjectLocationRegistrationTask]
public partial class ExampleObjectLocationRegistrationTask : RegistrationTask
{
       public ExampleObjectLocationRegistrationTask()
       {
            Description = "This tasks registers MyApp object location configurations.";
       }
       public override bool Register(ITaskOptions taskOptions)
       {
            StructureMap.IContainer container = ObjectLocator.Container as StructureMap.IContainer;
            container.Configure(x =>
            {
                x.For<MyApp.Interfaces.IExampleService>().Use<MyApp.Model.ExampleService>()
            });

            return base.Register(taskOptions);
       }
}
```

The next example shows how to get an instance of the ***IExampleService*** configured in the above registration object example. The system uses the ***Eleflex.ObjectLocator*** object and ***Current*** property:

```
//Get Example Service
var exampleService = ObjectLocator.Current.GetInstance<MyApp.Interfaces.IExampleService>();
```

Object Mapping

The ELEFLEX® platform uses the open source component ***AutoMapper*** for object to object mapping. By default, the standard naming conventions of objects within the various layers will use the same property names as each other. This is done so that when properties are added later, they use the same name within the various layered models and are then subsequently automatically mapped.

The platform provides a default mapping service, ***Eleflex.IMappingService,*** which is used to provide dynamic object-to-object mapping and is used through the various application layers. Calling one of the various ***Map()*** methods and supplying a source object and optionally a destination object, will move data between the two. An internal startup task takes care of registering the default mapping provider during system startup (discussed in the next sections).

Integrators can register configurations for mapping by inheriting from the ***Eleflex.RegistrationTask*** object and adding the [***MappingRegistrationTask***] class attribute. The following examples demonstrates how to create a class used to register any object mapping configurations you define.

```
using Eleflex;

[MappingRegistrationTask]
public partial class ExampleMappingRegistrationTask : RegistrationTask
{
	public ExampleMappingRegistrationTask()
	{
		Description = "This tasks registers MyApp object mapping configurations.";
	}
	public override bool Register(ITaskOptions taskOptions)
	{
		AutoMapper.Mapper.CreateMap<MyApp.Model.EditPerson, MyApp.Model.Person>();
		AutoMapper.Mapper.CreateMap<MyApp.Model.Person, MyApp.Model.EditPerson>();

		return base.Register(taskOptions);
	}
}
```

The next example shows how to map data between two objects using the ***Eleflex.IMappingService*** object.

```
//Use object locator to get mapping service
var mappingService = ObjectLocator.Current.GetInstance<Eleflex.IMappingService>();
MyApp.Model.EditPerson objA = new MyApp.Model.EditPerson();
objA.Name = “test”;

//Map Data Between Two Objects
MyApp.Model.Person objB = mappingService.Map<MyApp.Model.EditPerson,MyApp.Model.Person>(objA);
```

Logging

The platform provides a static entry-point object, ***Eleflex.Logger,*** which is consumed throughout the various layers of the application for storing system logging information. By using logging throughout your module, you can detect system health or error conditions that require further debugging. The ***Eleflex.Logger*** class contains the following property:

- ***Current*** (*ILoggingService*)
  - This property holds the logging service instance used to store log messages. This property is set by an internal startup task.

The application integrates with the open source project ***Common.Logging*** as it is used by other open source projects used in the system. The ***ILoggingService*** service contains multiple overloads of the following logging methods:

- Debug
- Error
- Fatal
- Info
- Warn
- Trace

The following examples show the various overloads to call on a method for the ***Logger*** object:

```
Logger.Current.Debug<MyClass>("Something happened");
Logger.Current.Debug<MyClass>(exception);
Logger.Current.Debug<MyClass>("Something happened with an exception", exception);
Logger.Current.Debug("MyClass","Something happened");
Logger.Current.Debug("MyClass","Something happened with an exception", exception);
```

While processing a request, we don't want log messages to be tied to the current Unit of Work as it may be rolled back instead of committed. The underlying mechanism for the ***ILoggingService*** uses a background thread to process log messages, ensuring that they are operated on a separate thread outside the scope of the calling process that created the log message. This additionally provides the benefit to the calling process not having to wait for log messages to be stored and can continue on with its own processing.

Messages stored with the ***ILoggingService*** using the ***Eleflex.LogMessage*** object, which are stored in with database in the table ***EleflexV3.LogMessage.*** To manipulate the LogMessage objects in storage, you would use with the ILogMessageServiceRepository, the ILogMessageBusinessRepository, or the ILogStorageRepository.

Security

The platform uses ***Microsoft ASP .NET Identity*** and ***OWIN*** to provide security for the system. The ASP.NET identity model required for implementation is done with the following classes:

- Eleflex.Security.ASPNetIdentity.IdentityUser
  - This is the base class that implements identity model requirements for a User.
- Eleflex.Security.ASPNetIdentity.IdentityRole
  - This is the base class that implements identity model requirements for a Role.
- Eleflex.Security.ASPNetIdentity.IdentityUserManager
  - The IdentityUserManager class is used to manipulate a User object.
- Eleflex.Security.ASPNetIdentity.IdentityRoleManager
  - The IdentityRoleManager class is used to manipulate a Role object.
- Eleflex.Services.WCF.OWIN.IdentitySignInManager
  - The IdentitySignInManager class is used to log a user in and subsequently store an OWIN cookie for authentication on the client.

To support the ASP.NET Identity model, several database tables were created to store the required information. The model was expanded from feature requirements to support effective date availability and permissions. The tables created for security include:

- EleflexV3.SecurityPermission
  - This table stores permission information. This is the same concept as a role and permissions are loaded into the user identity as Roles as well. This allows a migration path from older application designs with defined security matrixes incorporating permissions.
  - Mapped to Eleflex.SecurityPermission object and repositories.
- EleflexV3.SecurityRole
  - This table stores role information along with effective dates.
  - Mapped to Eleflex.SecurityRole object and repositories.
- EleflexV3.SecurityRolePermission
  - This table stores role to permission membership along with effective dates.
  - Mapped to Eleflex.SecurityRolePermission object and repositories.
- EleflexV3.SecurityRoleRole
  - This table stores role to role membership along with effective dates.
  - Mapped to Eleflex.SecurityRoleRole object and repositories.
- EleflexV3.SecurityUser
  - This table stores user information.
  - Mapped to Eleflex.SecurityUser object and repositories.
- EleflexV3.SecurityUserClaim
  - This table stores user claims.
  - Mapped to Eleflex.SecurityUserClaim object and repositories.
- EleflexV3.SecurityUserLogin
  - This table stores user external login tokens for other services.
  - Mapped to Eleflex.SecurityUserLogin object and repositories.
- EleflexV3.SecurityUserPermission
  - This table stores user to permission membership along with effective dates.
  - Mapped to Eleflex.SecurityUserPermission object and repositories.
- EleflexV3.SecurityUserRole
  - This table stores user to role membership along with effective dates.
  - Mapped to Eleflex.SecurityUserRole object and repositories.

The ASP.NET Identity store classes are used to manipulate security objects. There is a ***IUserStoreBusinessRepository*** and an ***IUserStoreServiceRepository*** used to access security in the application. If you are a server application, you would use the BusinessRepository. If you are a client application, then you would use the ServiceRepository. The same is true for the ***IRoleBusinessRepository*** and ***IRoleServiceRepository***. They are configured with the application

using structuremap in your application's object location registration task. These repositories are configured and obtained in code through the following interfaces:

- ***IUserStore<IdentityUser>***
  - This allows creating new users, adding/removing roles to users, set password, etc.
- ***IRoleStore<IdentityRole>***
  - This allows creating new roles, updating, searching and deleting.

These interfaces should be used for most security routines such as creating a user, getting a user by email, getting a user's roles/claims, etc. However you can operate on the underlying table data by using the Eleflex.Security* objects noted above.

Service-Oriented Architecture and Exposing Services

A core design premise of the ELEFLEX® platform is to expose all service commands for all system processing events and to provide a completely service-based user interface so that multiple platforms can be targeted as technologies evolve. The design also includes the ability to support each service hosted on a shared or independent hosted processes using request dispatchers.

The ***Eleflex.Services.WCF.IWCFCommand*** interface provides the core means to exposing a service method within the platform. The command interface uses a simple Request/Response paradigm for sending and receiving information, allowing for other communication method abstractions as needed. This method signature utilizes the ***ServiceKnownType()*** attribute that allows us to dynamically get a list of service commands exposed in the system for integrating all modules. Module service commands will be automatically registered with the ***WCFCommandRegistry*** object and then dynamically accessed depending on the request object sent to the system.

```
[ServiceContract]
public partial interface IWCFCommand
{
        [OperationContract]
        [ServiceKnownType("GetKnownTypes", typeof(WCFCommandRegistry))]
        Response ExecuteServiceCommand(Request request);

}
```

Exposing a service in the platform requires the following objects:

- Request object
  - This is the object that is sent to the system. All service commands in the system are inherited from an ***Eleflex.Request*** object and only one request object can be linked to a service command, so they cannot be reused for multiple service commands.
- Response object
  - This is the object that is returned from the system. All service commands must return a response object that contains information such as call success, error messages, etc. You

can use the default ***Eleflex.Response*** object for your service response or create an inherited class to return any additional information that needs to be returned to a caller.

- Service Interface
  - This is an interface that exposes the method signature with a request and response object that the command and service objects will implement. This provides the contract of information being passed and returned.
- Request Dispatcher object
  - This is the object is used by the client service object and allows use of different endpoints for different services in the platform. This allows services to be hosted all together or each service independently using different configurations. Typically a module only has one request dispatcher, however multiple can be created depending on requirements. This is inherited from ***Eleflex.Services.WCF.WCFCommandRequestDispatcher***
- Client Service object
  - This is the object implements the Service Interface that sends the request from the client to the hosted service using the Request Dispatcher object.
- Service Command object
  - This is the object that performs the business logic on the service host against the request object sent to the system from a client. Service commands inherit from ***Eleflex.Services.WCF.WCFCommand*** to expose their functionality.

Calling a service requires a service endpoint to be defined. By default, all service calls in a module are made through a request dispatcher that is configured to the "EleflexDefault" endpoint in the application's config file. This allows all modules to expose service commands on the same endpoint as other modules. Client Service objects in each of the modules call their respective request dispatcher to get the configuration to send the request out on. Integrators can then change the endpoint for a particular service to point to a new server address or using a different security configuration. This is done by changing the object location configuration for each service. For the logging service, it is ***ILoggingRequestDispatcher***, for security it is ***ISecurityRequestDispatcher*** and so on.

The following example displays the classes needed to expose a service command in the system. By using the ***WCFCommandRegistration()*** class attribute, the system will automatically register the command, request and response objects dynamically on system startup with an internal startup task. For client-calling applications not in the main hosted process, a separate configuration class is generated in the Messages assembly to register request and response objects on the client side for service communication.

```
//REQUEST OBJECT
public partial class ExampleRequest : Request
{
       public string Input { get; set; }
}

//RESPONSE OBJECT
public partial class ExampleResponse : Response
{
       public string Output { get; set; }
}

//SERVICE INTERFACE OBJECT
public partial interface IExampleService
{

      ExampleResponse ExampleMethod (ExampleRequest request);

}

//CLIENT SERVICE OBJECT
public partial class ExampleService : IExampleService
{

        public ExampleResponse ExampleMethod (ExampleRequest request)
        {
            using (IExampleRequestDispatcher dispatcher =
ObjectLocator.Current.GetInstance<IExampleRequestDispatcher>())
            {
                return dispatcher.ExecuteServiceCommand<ExampleResponse>(request); //SERVICE CALL
            }
        }
}

//REQUEST DISPATCHER OBJECT
public partial interface IExampleRequestDispatcher : IWCFCommandRequestDispatcher
{
}

public class ExampleRequestDispatcher : WCFCommandRequestDispatcher, IExampleRequestDispatcher
{
        public ExampleRequestDispatcher()
            : base(WCFConstants.SERVICE_ENDPOINT_NAME_DEFAULT)
        { }
        public ExampleRequestDispatcher(string endpoint)
            : base(endpoint)
        { }
}

//SERVICE COMMAND OBJECT
[WCFCommandRegistration(typeof(ExampleRequest), typeof(ExampleResponse))]
public partial class ExampleCommand : WCFCommand<ExampleRequest, ExampleResponse>

{
        public override void Execute(ExampleRequest request, ExampleResponse response)
        {
            response.Output = “Hello ” + request.Input;
        }

}
```

Securing service command methods can be done with the ***PrincipalPermission()*** class attribute or with a custom implementation. The next example shows adding the ***PrincipalPermission()*** attribute on the *Execute()* method to require only user's that have the "Admin" role are allowed to call the method.

```
[PrincipalPermission(SecurityAction.Demand, Role = "Admin")]
public override void Execute(ExampleRequest request, ExampleResponse response)
{
	response.Output = "Hello " + request.Input;
}
```

By default, OWIN is configured in the WCF pipeline with the current requestor's user principal information for all requests. There may come times when an anonymous user or user without sufficient privileges is making a request, however the system needs to make a call that requires elevated privileges. In these cases, you can impersonate a specific user or the system admin account with the following objects:

- ***Eleflex.Services.WCF.OWIN.ImpersonateUser***
  - This allows impersonating an individual user using a specified username and password.
- ***Eleflex.Services.WCF.OWIN.ImpersonateSystem***
  - This allows impersonating the system admin account using the configured shared secret token located in the application's config file using the key **EleflexImpersonateSystemToken.** This is a secret key used by managed ELEFLEX client hosts connecting to your ELEFLEX service host and should be changed for your site before deploying your application!

Impersonation is handled by the system through a server WCF behavior, ***Eleflex.Services.WCF.OWIN.CookieSecurityServerBehaviorExtension*** that is responsible for receiving and processing the request and from the client using the **ELEFLEX.Services.WCF.OWIN.CookieSecurityClientBehaviorExension**. As the name implies, the OWIN cookie is passed from client to server allowing authentication and authorization with OWIN since it has been injected into the WCP pipeline during a service call.

The next example shows how to make a service call impersonating the system account. In this case, we request the ***ISecurityUserServiceRepository*** interface that will allow us to make a service call to get users in the platform, which requires the requesting user to have "Admin" access. By making service calls while the ***ImpersonateSystem()*** object is alive, the request will resolve to the system admin account, rather than the user the call originated from and the call will succeed.

```
var service = ObjectLocator.Current.GetInstance<ISecurityUserServiceRepository>();
using (var adminAccess = new ImpersonateSystem())
{
       var resp = service.Get(new RequestItem<Guid>() { Item = userKey });

}
```

Layered Repository Design

The platform exposes data from database tables in a layered repository design. The layers consist of the following:

- Storage Repository
  - This repository is located in the Data Access Tier and contains data access logic for accessing data from a specific data storage mechanism such as SQL Server, Oracle, MySQL, etc. This repository is only available in the server process and should rarely be used, and only when needing to bypass business rules and event logic.
  - Default naming convention: I[object name]StorageRepository
- Business Repository
  - This repository is located in the Business Tier and contains business rule and event processing while relaying commands to the Storage Repository for actual work to be performed. This repository is only available in the server process and should be the most commonly used object to access data.
  - Default naming convention: I[object name]BusinessRepository
- Service Repository
  - This repository is located in the Service Contract Tier and is responsible for sending client service requests to the server process. The server process when receiving the request, then calls the Business Repository to perform the work. This repository is only available in client processes that contain only the service contract.
  - Default naming convention: I[object name]ServiceRepository
  - The T4 text templates that generate the service commands for these objects require the "Admin" role in order to call them over the service boundary.

All repository classes derive from the ***IRepository<>*** interface. It follows the standard Request/Response design paradigm so that requests through all layers are handled the same way. It contains the following methods:

- Insert
  - Insert an object
- Get
  - Get an object by its primary key value

- Update
  - Update an object
- Delete
  - Delete an object
- Query
  - Query for objects, including paging (See next section on dynamic querying)
- Query Aggregate
  - Return an aggregate count of records based on a query

```
public partial interface IRepository<TObject, TPkDataType>
    where TObject : class
{
    IResponseItem<TObject> Insert(IRequestItem<TObject> request);

    IResponseItem<TObject> Get(IRequestItem<TPkDataType> request);

    IResponseItem<TObject> Update(IRequestItem<TObject> request);

    IResponse Delete(IRequestItem<TPkDataType> request);

    IStorageQueryResponseItems<TObject> Query(IRequestItem<IStorageQuery> request);

    IResponseItem<double> QueryAggregate(IRequestItem<IStorageQuery> request);
}
```

The following code demonstrates use of each of the Repository methods above on the ***SecurityUser*** class, which define users in the system.

```
var service = ObjectLocator.Current.GetInstance<ISecurityUserBusinessRepository>();
var newUser = new SecurityUser();
newUser.FirstName = “test”;

//INSERT
var respInsert = service.Insert(new RequestItem<SecurityUser>(){Item = newUser});

//UPDATE
var updateUser = respInsert.Item;
var respUpdate = service.Update(new RequestItem<SecurityUser>(){Item = updateUser});

//GET
var respGet = service.Get(new RequestItem<Guid>(){Item = updateUser.UserKey});

//QUERY
var builder = new StorageQueryBuilder();
builder.IsEqual(“UserKey”, updateUser.UserKey.ToString());
var respQuery = service.Query(new RequestItem<IStorageQuery>(){Item = builder.GetStorageQuery()});

//QUERY AGGREGATE
var respQA = service.Query(new RequestItem<IStorageQuery>(){Item = builder.GetStorageQuery()});

//DELETE
var respDelete = service.Delete(new RequestItem<Guid>(){Item = updateUser.UserKey});
```

The ***IMappingRepository< >*** interface defines an object that allows mapping an object to a designation object before calling a repository method. This allows creating repositories for classes that do not have an underlying data store, but that map to a base case that can. This provides a shorthand way to collapse several lines of code into just a couple. The following example shows an **IdentityUser** object (ASP.NET Identity User implementation) that maps to the internal ***SecurityUser*** class that is mapped to storage. The following shows the usage with a ***MappingRepository***:

```
//Starting object
IdentityUser identityUser = new IdentityUser();
identityUser.FirstName = "test";

//Update an IdentityUser method #1
var repoM1 = ObjectLocator.Current.GetInstance<ISecurityUserServiceRepository>();
var mapM1 = ObjectLocator.Current.GetInstance<IMappingService>();

SecurityUser securityUser = map.Map<IdentityUser, SecurityUser>(identityUser);

var respM1 = repoM1.update(new ResponseItem<SecurityUser>(){Item=securityUser}); //SAVE USER


//Update an IdentityUser method #2
var repoM2 = ObjectLocator.Current.GetInstance<IMappingRepository<IdentityUser, Guid,
ServiceModel.SecurityUser, ISecurityUserServiceRepository>>();

var respM2 = repoM2.update(new ResponseItem<IdentityUser>(){Item=identityUser}); //SAVE USER


//Update an IdentityUser method #3 - this uses a pre-defined interface, same as #2
public interface IIdentityUserServiceRepository : IMappingRepository<IdentityUser, Guid,
SecurityUser, ISecurityUserServiceRepository

{}

var repoM3 = ObjectLocator.Current.GetInstance<IIdentityUserServiceRepository >();

var respM3 = repoM3.update(new ResponseItem<IdentityUser>(){Item=identityUser}); //SAVE USER
```

Dynamic Querying Over Service Boundaries

The ELEFLEX® platform needed a way of dynamic querying of information that crosses application boundaries and could also be leveraged against varying back-end storage mechanisms to support the **RF3 Dynamic SOA Data Manipulation** requirement. The solution was to build a simple model that could contain queryable expressions that would then be translated to dynamic Linq statements using the open source project ***LinqKit***.

Creating query expressions is done with the ***Eleflex.StorageQueryBuilder*** object. It contains standard methods and expressions to develop complex queries that can be transferred and translated easily. Because the information may pass application boundaries, the property name and data associated with the filter are stored as string values. The translation engine will parse the values to their native datatypes as needed. Specifying an invalid filter or a property name not defined on an object will return a response with a storage error code.

Name	Description
Aggregate	Perform an aggregate operation against a property including Average, BinaryChecksum, Checksum, Count, Minimum, Maximum, and Sum

Between	Perform a comparison between a high and low value
Compare	This includes IsEqual, NotEqual, GreaterThan, GreaterThanOrEqual, LessThan, and LessThanOrEqual.
Distinct	Get a distinct item set
Expression	Filter comparers including And, Or, BeginExpression and EndExpression. It is important to note that if an expression combiner (as in "And()" as well as "Or()") is not defined between two where clause filters, the engine will automatically use an "And()" filter when combining the filter tree.
Like	Comparison filter for StartsWith, EndsWith, Contains() and notcontains() operations using wild cards
Null	Determine if the property value is null or not null
PropertyComparison	Allows comparisons between properties
Select	Filter defines which properties are to be returned from a response. By default, all properties are selected for queries.
Set	Filter for InSet, NotInSet, defines a set of values that a property must value or not have
Sort	Determines the sort direction for a specific property
Paging	Filter that defines the number per page and page number of items that should be returned

The following example demonstrates how to create a simple query to find all users with the name equal to "TestUser" and have an EffectiveStartDate of null or an EffectiveStartDate in the past.

```
//CREATE REPOSITORY SERVICE
var service = ObjectLocator.Current.GetInstance<ISecurityRoleBusinessRepository>();

//CREATE BUILDER
StorageQueryBuilder builder = new StorageQueryBuilder();

//WHERE Name = "TestUser" AND have an effective start date of null or a date in the past
builder
  .IsEqual("Name", "TestUser")
  .And()
  .BeginExpression()
  .IsNull("EffectiveStartDate")
  .Or()
  .IsLessThanOrEqual("EffectiveStartDate", DateTimeOffset.UtcNow.ToString())
  .EndExpression();

//CALL SERVICE
var respRoles = service.Query(new ResponseItem<IStorageQuery>(){Item = builder.GetStorageQuery()});
```

Transaction Usage and Unit of Work

All storage transactions are handled using the ***Eleflex.IStorageContextUnitOfWork*** object. This object holds a list of all transactions that occur during processing of the current thread/request and allow *Commit()* or *RollBack()* of individual or all transactions created during the lifetime. Storage Repositories automatically register themselves with the Unit of Work object on creation. It is up to the programmer to *Commit()* or *Rollback()* the unit of work when all processing has been completed.

The ***Eleflex.IStorageContextUnitOfWork*** is configured with ***StructureMap*** by default to only create one instance per thread. This allows the same object instance to be shared by all objects on the same thread when using the object locator.

An example of managing transaction usage on a thread would be the following:

```
IStorageContextUnitOfWork uow = ObjectLocator.Current.GetInstance<IStorageContextUnitOfWork>();
try
{
        //DO WORK HERE WITH MULTIPLE REPOSITORIES

        uow.Commit();
}
catch (Exception ex)
{
        uow.Rollback();
        Logger.Current.Error("My transaction", ex);
}
```

Business Rules and Events

The platform provides business rules and events to execute customized business logic. These class provide an easy way to separate logic for a cleaner and more maintainable project. Additionally, they expose a mechanism to allow for cross-module integration of components and customization.

Business events are raised objects that inform other modules in the system of an application event. Business events are registered in the system by creating a new class that inherits from ***Eleflex.BusinessRuleEvent***. To fire the event, an extension method was created so that you simply create the object, set any properties and call the *RaiseEvent()* method. This causes all business rules subscribed to this object to be executed against the object. If any business rule being processed returns with a ResponseSuccess equals false, a managed exception will be thrown that will contain a user error message from the business rule that caused the error.

Business rules are classes that contain validation or business logic that subscribe to business events. Business rules are registered in the system by creating a new class that inherits from ***Eleflex.BusinessRule*** and it must have the ***[Eleflex.BusinessRuleProcess]*** class attribute applied with the object you plan on validating. Rather than registering each business rule individually, you can use

the ***Eleflex.BusinessRuleSet*** object instead with a reference to all linked business rules. This also must have the ***[Eleflex.BusinessRuleProcess]*** class attribute applied in order for the system to dynamic register it.

The Business Repository layer will raise an event with the object being operated on when the insert or the update method is called. This provides a general event that will be called before an object is saved. Assuming this succeeds, the Business Repository will call the Storage Repository next, along with its more granular events.

- (Object)
  - This event is fired just before an object is inserted or updated in the Business Repository. This provides a way to integrate generic extensions with other data access mechanisms when not using the Repository.
  - Register a business rule against this event by creating a business rule with the attribute:
    - BusinessRuleProcessAttribute(typeof(object))

The Storage Repository layer will raise an event with the object for the following methods during processing:

- RepositoryInsertEvent<object>
  - This event is fired just before the object is sent to storage.
  - Register a business rule against this event by creating a business rule with the attribute:
    - BusinessRuleProcessAttribute(typeof(RepositoryInsertEvent<object>))
- RepositoryDeleteEvent<object, datatype>
  - This event is fired just before the object is deleted from storage.
  - Register a business rule against this event by creating a business rule with the attribute:
    - BusinessRuleProcessAttribute(typeof(RepositoryDeleteEvent<object, datatype>))
- RepositoryGetEvent<object, datatype>
  - This event is fired just before the object is retrieved from storage.
  - Register a business rule against this event by creating a business rule with the attribute:
    - BusinessRuleProcessAttribute(typeof(RepositoryGetEvent<object, datatype>))
- RepositoryUpdateEvent<object>
  - This event is fired just before the object is updated in storage.
  - Register a business rule against this event by creating a business rule with the attribute:
    - BusinessRuleProcessAttribute(typeof(RepositoryUpdateEvent<object>))
- RepositoryQueryEvent<object>
  - This event is fired just before a query is performed.
  - Register a business rule against this event by creating a business rule with the attribute:
    - BusinessRuleProcessAttribute(typeof(RepositoryQueryEvent<object>))
- RepositoryQueryAggregateEvent<object>
  - This event is fired just before a query aggregate is performed.

- Register a business rule against this event by creating a business rule with the attribute:
  - BusinessRuleProcessAttribute(typeof(RepositoryQueryAggregateEvent<object>))

The following example shows how to create a new business event and how to raise it to let other areas of the system know it has been fired. Note that using the *RaiseEvent()* extension method will throw a managed exception if any rules fail that it processes.

```
public partial class ExampleBusinessRuleEvent : BusinessRuleEvent
{
	public virtual string Data { get; set; }

}

ExampleBusinessRuleEvent myEvent = new ExampleBusinessRuleEvent ();
myEvent.Data = “test”;
myEvent.RaiseEvent();
```

The following example demonstrates how to create a class that subscribes to *ExampleBusinessRuleEvent* being fired, from the above example.

```
[BusinessRuleProcessAttribute(typeof(ExampleBusinessRuleEvent))]
public partial class ExampleBusinessRule : BusinessRule
{

	public ExampleBusinessRule()
	{
		ErrorMessage = “Data cannot be null”;
	}
	public override IResponse Execute(IRequestItem<IContext> request)
	{
		IResponse response = base.Execute(request);
		ExampleBusinessRuleEvent exampleEvent = (ExampleBusinessRuleEvent) request.Item.Item;
		If(exampleEvent.Data == null)
			Response.AddMessage(true,this.ErrorMessage);
	    return response;
	}
}
```

This example demonstrates how to create a class that subscribes to the Business Repository raising an event on the SecurityUser object. This event will only be called from the Business Repository when the *Insert()* or *Update()* method is called.

```
[BusinessRuleProcessAttribute(typeof(SecurityUser))]
public partial class SecurityUserBusinessRule : BusinessRule
{

	public SecurityUserBusinessRule()
	{
		ErrorMessage = “FirstName cannot be null”;
	}
	public override IResponse Execute(IRequestItem<IContext> request)
	{
		IResponse response = base.Execute(request);
		SecurityUser user = (SecurityUser) request.Item.Item;
		If(user.FirstName == null)
			Response.AddMessage(true,this.ErrorMessage);
	    return response;
	}
}
```

The next example shows how to create a business rule that will subscribe to the Storage Repository Insert business event.

```
[BusinessRuleProcessAttribute(typeof(RepositoryInsertEvent<SecurityUser>>))]
public partial class ExampleUserBusinessRule : BusinessRule
{

	public ExampleUserBusinessRule()
	{
		ErrorMessage = “First name cannot be system”;
	}
	public override IResponse Execute(IRequestItem<IContext> request)
	{
		IResponse response = base.Execute(request);
		RepositoryInsertEvent<SecurityUser> insertEvent =
(RepositoryInsertEvent<SecurityUser>)request.Item.Item;

		If(insertEvent.FirstName == “system”)
			Response.AddMessage(true,this.ErrorMessage);
	    return response;
	}
}
```

MVC Page Routing and Embedded Web Applications

One of the platform’s design requirements was to support a modular framework that allowed for web applications to be dynamically embedded. This requirement was achieved with an open source project called ***MVC Code Routing***. This component allows a web application to add a reference to another web application and bind those pages to an MVC path in the server application. There are some rules to follow when building web applications, such as:

- It is recommended to create your module web controllers, models and view in the main website for debugging purposes first, then move all related files to the embedded web application once completed.
- Views in the web application module must have the Build Action property set to be “Embedded Resource” so that the View is compiled as an assembly resource.
- 3rd party user interface components are not required in the embeddable web application if they are installed in the server web application.
- Dots (.) are used as path separators. For example, accessing a website path of “/Products/Software” is defined as the path "Products.Software" and will call the SoftwareController object.
- Don't use dots (.) in the view file name except for the extension. When attempting to map a virtual path to an assembly resource, dots are interpreted as path separators (/).
- Use the “~~” prior to a controller name to start in the application root. Use the “+” prior to a controller path to denote it is a child of the current path.

Deployment Considerations

In the simplest of deployment configurations, all presentation, services and logic can be housed within one application. In the most complex scenario, each service and presentation tier can be deployed to its process, machine or multiple machines for scalability and fault tolerance, as may be required by the customized infrastructure. WCF configuration bindings additionally provide more control of security and process handling of requests as customization may require.

Code Generation and Customization Considerations

One of the design considerations for the ELEFLEX® enterprise application framework has to do with the eventual customization of models and logic within the framework. All classes in the framework have been marked *partial* as well as all methods and properties marked *virtual* for overriding. Additionally, all variables should either be public or protected. Customizers should create new partial class files where ever possible to avoid conflicts with framework updates.

Application NuGet Packages

NuGet is the package manager for the Microsoft development platform including .NET. The NuGet client tools provide the ability to produce and consume packages. The NuGet Gallery is the central package repository used by all package authors and consumers.

The ELEFLEX® platform has been separated into several NuGet packages to allow module developers to reference certain portions of the platform that are required for a particular assembly or layer in the n-tiered architecture without needing to reference unused components. The following sections describe each package and their contents that are used to install the ELEFLEX® platform.

The following diagram displays which NuGet packages are installed on an ELEFLEX® web server and web client installation.

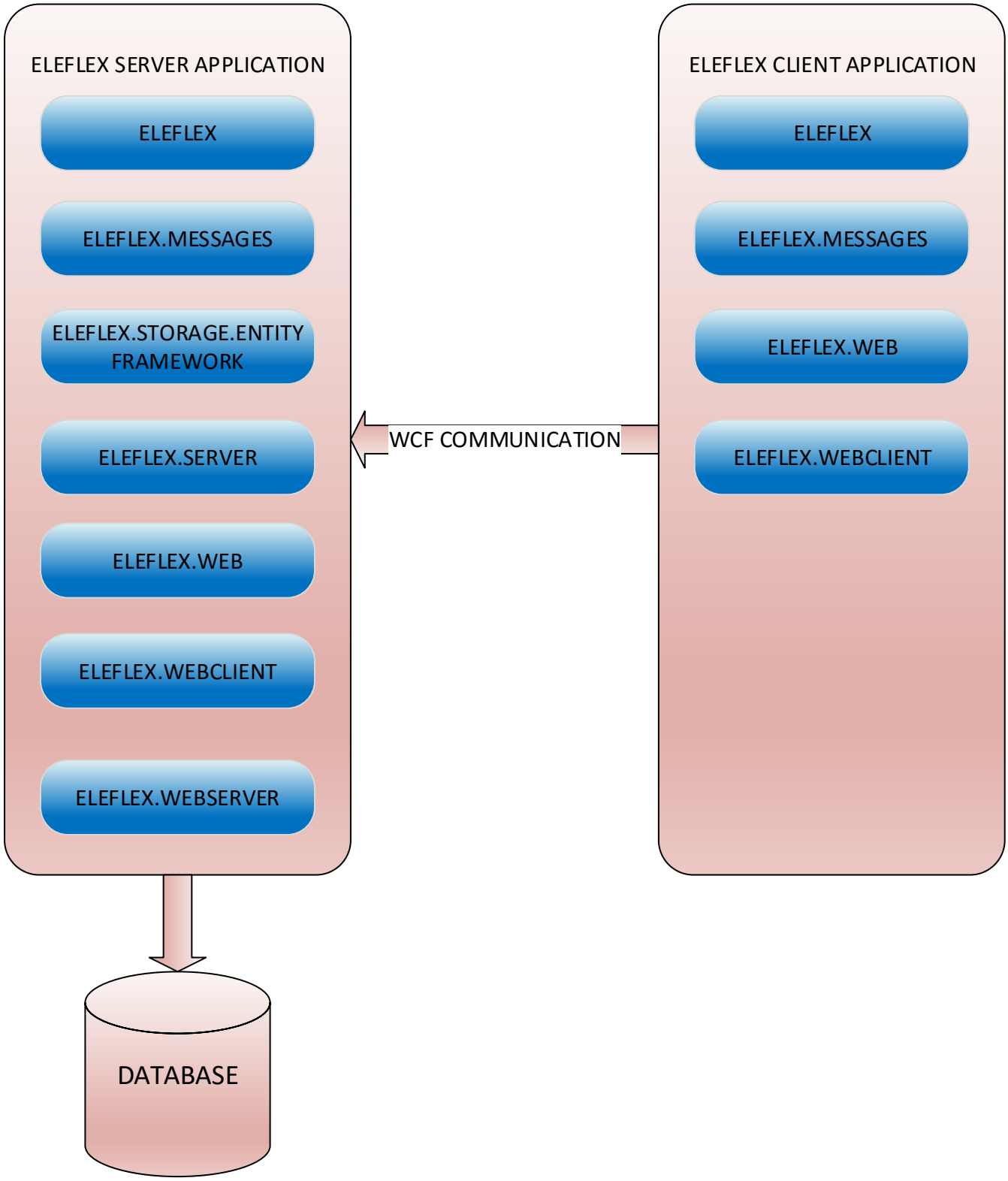

FIGURE 3- ELEFLEX NUGET PACKAGE INSTALLATION OVERVIEW

NuGet Open Source Project References

The ELEFLEX® platform utilizes serval open source projects in its design to fulfill architectural needs, design characteristics, system requirements or needed functionality. The table below lists all components, a general overview of the project and how and where they are used in the platform.

NuGet Project	Version	Description	ELEFLEX Usage
Antlr	3.5.0.2	ANother Tool for Language Recognition, is a language tool that provides a framework for constructing recognizers, interpreters, compilers, and translators from grammatical descriptions containing actions in a variety of target languages.	Required by web compilers.
AutoMapper	4.1.1	A convention-based object-object mapper. AutoMapper uses a fluent configuration API to define an object-object mapping strategy. AutoMapper uses a convention-based matching algorithm to match up source to destination values. Currently, AutoMapper is geared towards model projection scenarios to flatten complex object models to DTOs and other simple objects, whose design is better suited for serialization, communication, messaging, or simply an anti-corruption layer between the domain and application layer.	Used for object to object mapping. See Eleflex.IMappingService
bootstrap	3.3.6	The most popular front-end framework for developing responsive, mobile first projects on the web.	Used for web page styling.
bootstrap.chosen	1.0.0	An alternate stylesheet for Chosen 1.0. This one is supposed to integrate better with Bootstrap 3.0.	Used for web drop down pickers.
chosen	1.2.0	Chosen is a JavaScript plugin that makes long, unwieldy select boxes much more user-friendly. It is currently available in both jQuery and Prototype flavors.	Used for web drop down pickers.
chosen.jquery	1.2.0	Chosen is a JavaScript plugin that makes long, unwieldy select boxes much more user-friendly. It is currently available in both jQuery and Prototype flavors.	Used for web drop down pickers.
Common.Logging	3.3.1	Common.Logging library introduces a simple abstraction to allow you to select a specific logging implementation at runtime.	Used for application logging.
Common.Logging.Core	3.3.1	Common.Logging.Core contains the portable (PCL) implementation of the Common.Logging low-level abstractions common to all other Common.Logging packages.	Used for application logging.

CommonServiceLocator	1.3	The Common Service Locator library contains a shared interface for service location which application and framework developers can reference. The library provides an abstraction over IoC containers and service locators. Using the library allows an application to indirectly access the capabilities without relying on hard references. The hope is that using this library, third-party applications and frameworks can begin to leverage IoC/Service Location without tying themselves down to a specific implementation. This library contains a portable class library that targets .NET Framework 4, Windows 8, Windows Phone Silverlight 8, Windows Phone 8.1, and Silverlight 5.	Used for object location.
CommonServiceLocator.StructureMapAdapter.Unofficial	3.0.4.125	An unofficial StructureMap Adapter for the Common Service Locator	Used for object location tying CommonServiceLocator and Structuremap together.
EntityFramework	6.1.3	Entity Framework is Microsoft's recommended data access technology for new applications.	Used for Microsoft SQL Server and Microsoft Azure database access. See Eleflex.Storage.EF.EntityStorageService
FontAwesome	4.4.0	Iconic font designed for use with Twitter Bootstrap. Package Issues? Post them to https://github.com/JustLikeIcarus/Font-Awesome-NuGet/issues	Used for web page icons.
jQuery	2.1.4	jQuery is a new kind of JavaScript Library. jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.	Used for web scripting.
jquery.datatables	1.10.9	DataTables is a plug-in for the jQuery Javascript library. It is a highly flexible tool, based upon the foundations of progressive enhancement, which will add advanced interaction controls to any HTML table.	Used for web table display.
jQuery.UI.Combined	1.11.4	jQuery UI is an open source library of interface components — interactions, full-featured widgets, and animation effects — based on the stellar jQuery javascript library . Each component is built according	Used for web display elements.

		to jQuery's event-driven architecture (find something, manipulate it) and is themeable, making it easy for developers of any skill level to integrate and extend into their own code.	
jQuery.Validation	1.14.0	This jQuery plugin makes simple clientside form validation trivial, while offering lots of option for customization. That makes a good choice if you're building something new from scratch, but also when you're trying to integrate it into an existing application with lots of existing markup. The plugin comes bundled with a useful set of validation methods, including URL and email validation, while providing an API to write your own methods. All bundled methods come with default error messages in english and translations into 32 languages.	Used for web scripting validations.
LINQKit	1.1.3.1	LINQKit is a free set of extensions for LINQ to SQL and Entity Framework power users.	Used to translate dynamic SOA queries into Linq commands. See Eleflex.Storage.EF.EntityQueryBuilder
Microsoft.AspNet.Identity.Core	2.2.1	Core interfaces for ASP.NET Identity.	Used for security.
Microsoft.AspNet.Identity.Owin	2.2.1	Owin implementation for ASP.NET Identity.	Used for security.
Microsoft.AspNet.Mvc	5.2.3	ASP.NET MVC is a web framework that gives you a powerful, patterns-based way to build dynamic websites and Web APIs. ASP.NET MVC enables a clean separation of concerns and gives you full control over markup.	Used for web page display.
Microsoft.AspNet.Razor	3.2.3	Razor is a markup syntax for adding server-side logic to web pages. This package contains the Razor parser and code generation infrastructure.	Used for web page display.
Microsoft.AspNet.Web.Optimization	1.1.3	ASP.NET Optimization introduces a way to bundle and optimize CSS and JavaScript files.	Used for web script and styles bundling.
Microsoft.AspNet.WebPages	3.2.3	This package contains core runtime assemblies shared between ASP.NET MVC and ASP.NET Web Pages	Used for web page display.
Microsoft.CodeDom.Providers.DotNetCompilerPlatform	1.0.1	Replacement CodeDOM providers that use the new .NET Compiler Platform ("Roslyn") compiler as a service APIs. This provides support for new language features in systems using CodeDOM (e.g. ASP.NET runtime compilation) as well as improving	Used for web page compilation.

		the compilation performance of these systems.	
Microsoft.jQuery.Unobtrusive.Validation	3.2.3	jQuery plugin that unobtrusively sets up jQuery.Validation.	Used be web scripting validations.
Microsoft.Net.Compilers	1.1.0	.Net Compilers package. Referencing this package will cause the project to be built using the specific version of the C# and Visual Basic compilers contained in the package, as opposed to any system installed version.	Required for web applications.
Microsoft.Owin	3.0.1	Provides a set of helper types and abstractions for simplifying the creation of OWIN components.	Used for security.
Microsoft.Owin.Host.SystemWeb	3.0.1	OWIN server that enables OWIN-based applications to run on IIS using the ASP.NET request pipeline.	Used for security.
Microsoft.Owin.Security	3.0.1	Common types which are shared by the various authentication middleware components.	Used for security.
Microsoft.Owin.Security.Cookies	3.0.1	Middleware that enables an application to use cookie based authentication, similar to ASP.NET's forms authentication.	Used for security.
Microsoft.Owin.Security.OAuth	3.0.1	Middleware that enables an application to support any standard OAuth 2.0 authentication workflow.	Used for security.
Microsoft.Web.Infrastructure	1.0.0.0	This package contains the Microsoft.Web.Infrastructure assembly that lets you dynamically register HTTP modules at run time.	Used for startup (custom).
Modernizr	2.8.3	Modernizr adds classes to the <html> element which allow you to target specific browser functionality in your stylesheet. You don't actually need to write any Javascript to use it. Modernizr is a small and simple JavaScript library that helps you take advantage of emerging web technologies (CSS3, HTML5) while still maintaining a fine level of control over older browsers that may not yet support these new technologies.	Used for web page styling.
Moment.js	2.10.6	A lightweight javascript date library for parsing, manipulating, and formatting dates.	Used for web scripting.
MvcCodeRouting	1.3.0	Namespace-based Modularity for ASP.NET MVC and Web API. Convention over configuration Automatic Routing.	Used for MVC page routing and Module embedded web applications.
Newtonsoft.Json	7.0.1	Json.NET is a popular high-performance JSON framework for .NET	Used for web page display.
Owin	1.0	OWIN IAppBuilder startup interface	Used for security.

Respond	1.4.2	The goal of this script is to provide a fast and lightweight (3kb minified / 1kb gzipped) script to enable responsive web designs in browsers that don't support CSS3 Media Queries - in particular, Internet Explorer 8 and under. It's written in such a way that it will probably patch support for other non-supporting browsers as well (more information on that soon).	Used for web page styling.
smalot.bootstrap-datetimepicker	1.0.4	This library is developed by Smalot, this package is just copy the library to nuget.	Used for web date/time pickers.
structuremap	4.0.0.315	StructureMap is a Dependency Injection / Inversion of Control tool for .Net	Used for object location.
structuremap.web	4.0.0.315	ASP.Net specific functionality for use with the StructureMap IoC container	Used for object location.
WebGrease	1.6.0	Web Grease is a suite of tools for optimizing javascript, css files and images.	Used for web scripting and styling.

ELEFLEX NuGet Packages

Eleflex Package

This package contains the platform's core object model. This package should be added to all assemblies in the application. This consists of the following core interfaces and models:

- BusinessRules
  - This contains BusinessRules, BusinessEvents and the BusinessRuleService used for processing events.
- Context
  - This contains a model used to store the context of a call, including user information
- Core
  - This contains constants and base interfaces used in the platform.
- Exceptions
  - This contains managed exceptions used by the platform.
- Logging
  - This contains interfaces and objects for logging application messages
- Mapping
  - This contains interfaces used for mapping an object to another object.
- Messages
  - This contains the object model used for service messages.
- ObjectLocation
  - This contains the interfaces and model for the ObjectLocator object used for creating objects and their dependencies.
- RequestResponse

    - This contains the objects used for service request and response objects.
- Security
    - This contains the object used for security, including SecurityUser, SecurityRole, SecurityPermission and more.
- Services
    - This contains the interfaces for sending data over services.
- Startup
    - This contains objects used for application startup, registrations and shutdown.
- Storage
    - This contains the main repository interfaces for exposing persistent data in the platform.
- Versioning
    - This contains the object model used for versioning modules in the platform.

Eleflex.Messages Package

This package contains the WCF service components used for defining service contracts and communicating with the server application from a client application. This package should be added to assemblies that expose service contracts. This consists of the following namespaces:

- Eleflex.Services.WCF
    - This contains the core interfaces and objects for service commands and service request dispatchers.
- Eleflex.Logging.Services.WCF
    - This contains the core interfaces and objects for exposing logging information.
- Eleflex.Security.Services.WCF
    - This contains the core interfaces and objects for exposing security information.
- Eleflex.Versioning.Services.WCF
    - This contains the core interfaces and objects for exposing versioning information.

Eleflex.Server Package

This package contains the server commands and data access components required to host services in the platform for the platform's core object model. This contains the core classes for logging, security and versioning. This package should only be added to the server application that exposes service commands and data access for the core application components. This consists of the following namespaces:

- Eleflex.Logging.Services.WCF.Server
    - This contains service commands for the logging module.
- Eleflex.Logging.Storage.EF
    - This contains the entity framework data access layer for the logging module.
- Eleflex.Logging.Storage.EF.AutoMapper
    - This contains the business model to storage model AutoMapper mappings.

- Eleflex.Logging.Storage.EF.Azure
  - This contains the data access components for use with Microsoft Azure.
- Eleflex.Logging.Storage.EF.SQLServer
  - This contains the data access components for use with Microsoft SQL Server.
- Eleflex.Security.Services.WCF.Server
  - This contains service commands for the Security module.
- Eleflex.Security.Storage.EF
  - This contains the entity framework data access layer for the Security module.
- Eleflex.Security.Storage.EF.AutoMapper
  - This contains the business model to storage model AutoMapper mappings.
- Eleflex.Security.Storage.EF.Azure
  - This contains the data access components for use with Microsoft Azure.
- Eleflex.Security.Storage.EF.SQLServer
  - This contains the data access components for use with Microsoft SQL Server.
- Eleflex.Versioning.Services.WCF.Server
  - This contains service commands for the Versioning module.
- Eleflex.Versioning.Storage.EF
  - This contains the entity framework data access layer for the Versioning module.
- Eleflex.Versioning.Storage.EF.AutoMapper
  - This contains the business model to storage model AutoMapper mappings.
- Eleflex.Versioning.Storage.EF.Azure
  - This contains the data access components for use with Microsoft Azure.
- Eleflex.Versioning.Storage.EF.SQLServer
  - This contains the data access components for use with Microsoft SQL Server.

Eleflex.Storage.EntityFramework Package

This package contains the data access layer components used to expose data from Microsoft SQL Server and Microsoft Azure database platforms. This package should be added to Module server packages that contain data access components. This consists of the following namespaces:

- Eleflex.Storage.EF
  - This contains the core Entity Framework service repository and storage service objects used to connect to SQL Server and Azure database engines.

Eleflex.Web Package

This package contains the startup components for integrating logging with common logging, object location with Structuremap, object mapping for Automapper and ASP.NET Identity and OWIN security classes. This package should be added to Module embedded web application packages. This consists of the following namespaces:

- Eleflex.Logging.CommonLogging

    - This contains Common.Logging components that expose logging in the application using the Business Repository. This should be used for server applications.
- Eleflex.Logging.CommonLogging.WCF
    - This contains Common.Logging components that expose logging in the application using the Service Repository. This should be used for client applications.
- Eleflex.Mapping.AutoMapper
    - This contains the Automapper service and registration tasks.
- Eleflex.ObjectLocation.CSL
    - This contains the core CommonServiceLocator service.
- Eleflex.ObjectLocation.CSL.StructureMap
    - This contains the ObjectLocation and Business Rule startup tasks.
- Eleflex.ObjectLocation.CSL.StructureMap.Web
    - This contains web components for MVC dependency injection, ModelState extensions, and Unit of Work registration tasks.
- Eleflex.Security.ASPNetIdentity
    - This contains the core ASP .NET Identity model to expose security in the application. It exposes the IdentityUser, IdentityRole and Business Repositories to interact with data.
- Eleflex.Security.ASPNetIdentity.AutoMapper
    - This contains an Automapper registration task to register Security to Identity object mappings.
- Eleflex.Services.WCF.OWIN
    - This contains the WCF server and client behaviors for system impersonation and Service Repository objects for security services.
- Eleflex.Web
    - This contains web componenets such as web background processes, AjaxResponse and other web objects.

Eleflex.WebClient Package

This package contains web startup and registration tasks that configure a web application to communicate with an ELEFLEX® server application. It also contains dependencies to the web-based open source packages that are used in the platform, such as jquery, bootstrap, etc. This package should be added to an ELEFLEX® client application. This package should not be added to a Module's embedded web application.

Eleflex.WebServer Package

This package contains web startup and registration tasks that configure a web application to become an ELEFLEX® server application. This package relies on startup and registration tasks used in the Eleflex.WebClient package, and additionally overwrites other files that are specific to the server application. It contains the Eleflex.Server package to exposes the WCF commands and connect to the

data access providers. This package should be added to an ELEFLEX® server application. This package should not be added to a Module's embedded web application.

ELEFLEX Module Design

This section outlines an ELEFLEX Module that integrates with the platform.

Using Eleflex.Email Module as an Example

For the next sections, we are going to examine the Eleflex.Email Module. The source code for the Eleflex.Email module is included in the same release as the ELEFLEX complete source code. Download the complete source code online. The Eleflex.Email module project files are located in the solution folder under Release/NuGet.

The Email Module provides distributed storage and sending of emails. By default in the ELEFLEX platform, emails are sent directly to an SMTP server. If the SMTP server is offline, the email trying to be sent could be lost, or an application workflow could not proceed unless the email is sent. To alleviate this, the Email module provides the following functionality:

- It creates database tables used to store email messages temporarily.
- It creates an object location configuration that maps the ASP.NET Identity email store to use its service, which stores the email in the database table instead of trying to send it immediately.
- It creates a web process that runs every minute to see if any email messages need to be sent. It will then pull messages out in a queue fashion and attempts to send the email. If it succeeds, it marks the email as being sent. If a send error happens, the email stays in the queue and will be attempted later.
- It creates a web process that runs once every 8 hours to purge emails from the system that are older than a configured number of days. By default, emails are deleted after being 30 days old.

Module Assemblies

Assembly namespaces are laid out with the first tuple comprising the company name, the module name, and the remainder following the ELEFLEX project template name.

- Eleflex.Email
  - This assembly contains the core interfaces, business model, rules, events and workflow needed for the module.
- Eleflex.Email.Messages
  - This assembly contains the service contract interfaces, models and client services needed to call a hosted service that exposes this information.
- Eleflex.Email.Server
  - This assembly contains the WCF service commands exposed from the Messages assembly as well as the data access components needed to access the data directly.
- Eleflex.Email.WebClient

    - This assembly is optional, however it can be used to hold logic, as well as provide versioning information to be displayed in the platform.
    - The module uses this assembly to store an ASP.NET Identity Email Service using a Service Repository and also provide versioning information.
- Eleflex.Email.WebServer
    - This assembly is optional, however it can be used to hold logic, as well as provide versioning information to be displayed in the platform.
    - The module uses this assembly to store an ASP.NET Identity Email Service using a Business Repository and also provide versioning information.
- Eleflex.Email.Web.Admin
    - This assembly is a web application that will be embedded in the host application to displays views made for the Email module.

Database Tables and Entity Model

The Email Module creates the following tables to store email messages temporarily:

- EmailProcess
    - This contains the main email components such as from, to subject, body, etc., as well as columns for processing emails.
- EmailProcessAttachment
    - This contains any attachments to the email.

In order to manipulate these tables, we rely on an object relational mapping tool (ORM) from Microsoft called Entity Framework. The Eleflex.Storage.EntityFramework NuGet package contains framework classes that perform the work using this technology.

The Eleflex.Email.Server project contains the ADO .NET Entity Data Model for the database tables called “EmailDB.edmx”. This allows developers access to the database table programmatically. This file is important because the T4 text templates will generate all the default classes needed in the module using this, discussed in the next section.

T4 Text Templates Generate Needed Files

The platform utilizes T4 text templates to generate nearly all needed files for a Module. The templates require an Entity Data Model to be created first, to use as a data source. At the top of the template will be a configuration section where you will modify the input parameters to work with your project. By default, the Entity Data Model should be created in the Eleflex.Email.Server project.

The text templates will create partial interfaces and classes, as well as make sure methods are marked virtual to allow developers to create derived classes with the most overloading possibilities. Running the template will cause all files to be regenerated, so do not make changes in generated files. Whenever possible, create a new partial file to add functionality, otherwise modifying the template may be required.

The text templates for each of the assemblies is listed along with the files produced for each template.

- Eleflex.Email
  - Eleflex_Email_CodeGen.tt
    - (table name).cs
      - Business model object
    - I(table name).cs
      - Interface for business model object
    - I(table name)BusinessRepository.cs
      - Interface for a Business Repository for managed data access.
    - I(table name)StorageRepository.cs
      - Interface for a Storage Repository for managed data access.
    - (table name)BusinessRepository.cs
      - Business Repository class for managed data access.
    - I(table name)StorageServer.cs
      - Interface for a Storage Service. (Such as SQL Server access via Entity Framework)
    - (module name)Constants.cs
      - A class containing constants used by the various layers
- Eleflex.Email.Messages
  - Eleflex_Email_Services_WCF_Message_CodeGen.tt
    - (table name).cs
      - Service model object
    - (table name)DeleteRequest.cs
      - Request object for the delete service command.
    - (table name)DeleteResponse.cs
      - Response object for the delete service command.
    - (table name)GetRequest.cs
      - Request object for the get service command.
    - (table name)GetResponse.cs
      - Response object for the get service command.
    - (table name)InsertRequest.cs
      - Request object for the insert service command.
    - (table name)InsertResponse.cs
      - Response object for the insert service command.
    - (table name)QueryAggregateRequest.cs
      - Request object for the query aggregate service command.
    - (table name)QueryAggregateResponse.cs
      - Response object for the query aggregate service command.

        - (table name)QueryRequest.cs
            - Request object for the query service command.
        - (table name)QueryResponse.cs
            - Response object for the query service command.
        - (table name)UpdateRequest.cs
            - Request object for the update service command.
        - (table name)UpdateResponse.cs
            - Response object for the update service command.
        - I(table name)ServiceRepository
            - Interface for the Service Repository for service-based data access.
        - I(table name)ServiceRepository
            - Service Repository class for service-based data access.
        - I(table name)ServicesRegistrationTask
            - Registration task to configure WCF access for client applications.
        - I(module name)RequestDispatcher
            - Interface for a Request Dispatcher used to send WCF messages for the module.
        - (module name)RequestDispatcher
            - Request Dispatcher class used to send WCF messages for the module.
- Eleflex.Email.Server
    - Eleflex_Email_Services_WCF_AutoMapper_CodeGen.tt
        - (table name)ServicesAutomapperRegistrationTask.cs
            - Registers Automapper mapping between the business and service models
    - Eleflex_Email_Services_WCF_Server_CodeGen.tt
        - (table name)Delete.cs
            - WCF service command for repository delete.
        - (table name)Get.cs
            - WCF service command for repository get.
        - (table name)Insert.cs
            - WCF service command for repository insert.
        - (table name)Query.cs
            - WCF service command for repository query.
        - (table name)Update.cs
            - WCF service command for repository update.
        - (table name)QueryAggregate.cs
            - WCF service command for repository query aggregate
        -
    - Eleflex_Email_Storage_EF_CodeGen.tt
        - (table name)StorageRepository.cs

- Entity Framework storage repository class.
- (module name)StorageService.cs
  - Module data storage service so all storage repository use the same database connection
- (entity data model name)Custom.cs
  - Adds a partial class with a constructor taking in a connection string

- Eleflex_Email_Storage_EF_AutoMapper_CodeGen.tt
  - (table name)StorageAutomapperRegistrationTask.cs
    - Automapper configuration for business to storage model mapping

Embedded Web Application

An embedded web application allows deploying a web application as an assembly reference and mapping its root controller an MVC route in the host application. This is accomplished by using the MVC Code Routing open source component. Using namespace-based routing, instead of conventional MVC routing, routes don't need to be configured and are dynamically found by using the root controller namespace. This reduces NuGet upgrade issues with deploying individual files web application files in a host process as the whole module application can be deployed as a project reference.

Creating an embedded web application is the same as creating any other web application. One key difference is that you must set the property for all views in the application for "Build Action" to "Embedded Resource". This is found in the Visual Studio property window for the view. This will embed the view in the assembly so that it can be found by the dynamic routing engine in the host application. It is recommended for debugging purposes to create the controllers/models/views in the host application first, then once working, move then to the embedded web application.

Examining the Eleflex.Email.Web.Admin project, it contains controllers, models and views as any other web application would. In order to register this embedded web application in a host application, we need to add an MVC route configuration. For the default system components, this is found in web server and web client applications in the App_Start/Eleflex_Start/WebRoutesStartupTask.

When installing new ELEFLEX® Modules, developers should try not to modify any existing files in an ELEFLEX® installation. Instead, create new files specific to your module and use the startup conventions to configure and register your module.

The next example shows how the Eleflex.Email module configures its embedded admin web application route into the host process. This is found in the App_Start/Eleflex_Start/EleflexEmail/WebClientRoutesStartupTask.cs file. This will set the host web application route for "/Admin/Email" to point to the Eleflex.Email.Web.Admin project's AdminController and all web requests will be funneled to this application.

```
using System.Web.Routing;
using Eleflex;
using MvcCodeRouting;

namespace WebServer.App_Start.Eleflex_Start.EleflexEmail
{
    /// <summary>
    /// Represents a startup task for configuring routes in the web application.
    /// </summary>
    public partial class WebClientRoutesStartupTask : StartupTask
    {

        /// <summary>
        /// Constructor.
        /// </summary>
        public WebClientRoutesStartupTask() : base()
        {
            Description = @"This task registers mvc routes used for the ELEFLEX Email Module.";
            Priority = StartupConstants.PRIORITY_CUSTOM;
        }

        /// <summary>
        /// Start processing logic.
        /// </summary>
        /// <param name="taskOptions"></param>
        /// <returns></returns>
        public override bool Start(ITaskOptions taskOptions)
        {

            //CONFIGURE ROUTE FOR ADMIN/Email
            RouteTable.Routes.MapCodeRoutes(
               baseRoute: "Admin/Email",
               rootController: typeof(Eleflex.Email.Web.Admin.Controllers.AdminController),
               settings: new CodeRoutingSettings
               {
                   EnableEmbeddedViews = true,
               }
            );

            return base.Start(taskOptions);
        }
    }
}
```

Configure Startup and Registration Tasks

The Email Module needs to register object location and registration tasks. It creates these files in the server and client applications in the App_Start/Eleflex_Start/EleflexEmail folder. It creates the following:

- WebClientObjectLocationRegistrationTask.cs

    - This task registers the EmailRequestDispatcher for service-based access and registers a new email service for the ASP.NET Identity mail service used to store emails rather than sending them.
- WebClientRoutesStartupTask.cs
    - This task registers the embedded web application in the host process.
- WebServerObjectLocationRegistrationTask.cs (WebServer only)
    - This task registers the Email StorageService for database access.
- WebServerProcessStartupTask.cs
    - This task starts background processes for sending emails and purging emails.

Patches for Integration and System Updates

The Email module needs to create versioning patch files so that it can be packaged and deployed on other ELEFLEX® platforms. This is very important to keep database DDL changes in sync with the code. Current Patch version numbers are stored in the versioning table and displayed in the admin section of the web application.

ELEFLEX® web server applications have a startup task added to their application called SystemPatchStartupTask.cs. This startup task loads the patch manager and starts the upgrade processes for all installed modules. This following lists the Email assemblies and the patches created for each.

- Eleflex.Email.Messages
    - ModulePatch Folder
        - The patches in this module are created strictly for cosmetic reasons so that the Eleflex.Email.Messages package is registered in the versioning table for completeness.
- Eleflex.Email.Server
    - ModulePatch Folder
        - The patches in this module are created strictly for cosmetic reasons so that the Eleflex.Email.Server package is registered in the versioning table for completeness.
    - Eleflex.Email.Storage.EF.Azure Folder
        - The patches in this module are created to keep Microsoft Azure database DDL/schema changes in sync with the code.
    - Eleflex.Email.Storage.EF.SQLServer Folder
        - The patches in this module are created to keep Microsoft SQL Server database DDL/schema changes in sync with the code.
- Eleflex.Email.WebClient
    - ModulePatch Folder

        - The patches in this module are created strictly for cosmetic reasons so that the Eleflex.Email.WebClient package is registered in the versioning table for completeness.
- Eleflex.Email.WebServer
    - ModulePatch Folder
        - The patches in this module are created strictly for cosmetic reasons so that the Eleflex.Email.Webserver package is registered in the versioning table for completeness.

NuGet Packaging

Each NuGet package has an *.nuspec file that contains information about the package. This contains metadata such as the project id, name, summary, project references, etc. Additionally, NuGet includes a folder structure on adding assemblies, content and project references to a package. These are stored in version number folders found in each solution's NuGet folder using file explorer.

The Email module is broken up into the following NuGet packages:

- Eleflex.Email.Messages
    - This contains the service contracts for calling email services.
    - Contains Eleflex.Email.Messages assembly.
- Eleflex.Email.Server
    - This contains the service commands and data access logic
    - Contains Eleflex.Email and Eleflex.Email.Server assemblies.
- Eleflex.Email.WebClient
    - This contains web client-based logic, service calls, embedded web application, patches and startup tasks.
    - Contains the Eleflex.Email.WebClient and Eleflex.Email.Web.Admin assemblies.
- Eleflex.Email.WebServer
    - This contains web server-based logic, patches and startup tasks.
    - Contains the Eleflex.Email.WebServer assembly.

The following diagram displays an ELEFLEX Module package installation requirements for a web server and web client application.

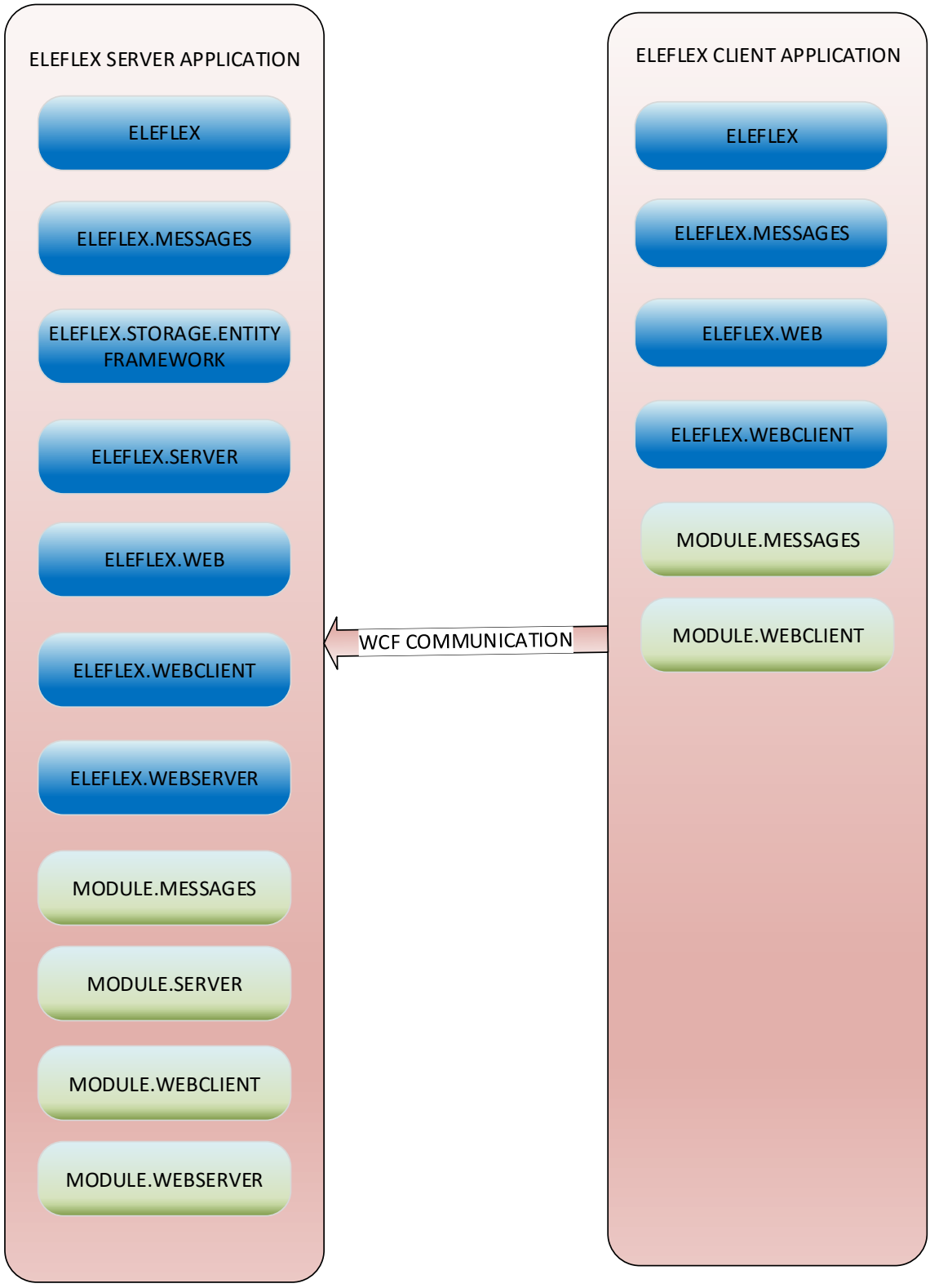

FIGURE 4- MODULE NUGET PACKAGE INSTALLATION OVERVIEW

The next diagram shows the package dependencies of an ELEFLEX Module with the platform released packages.

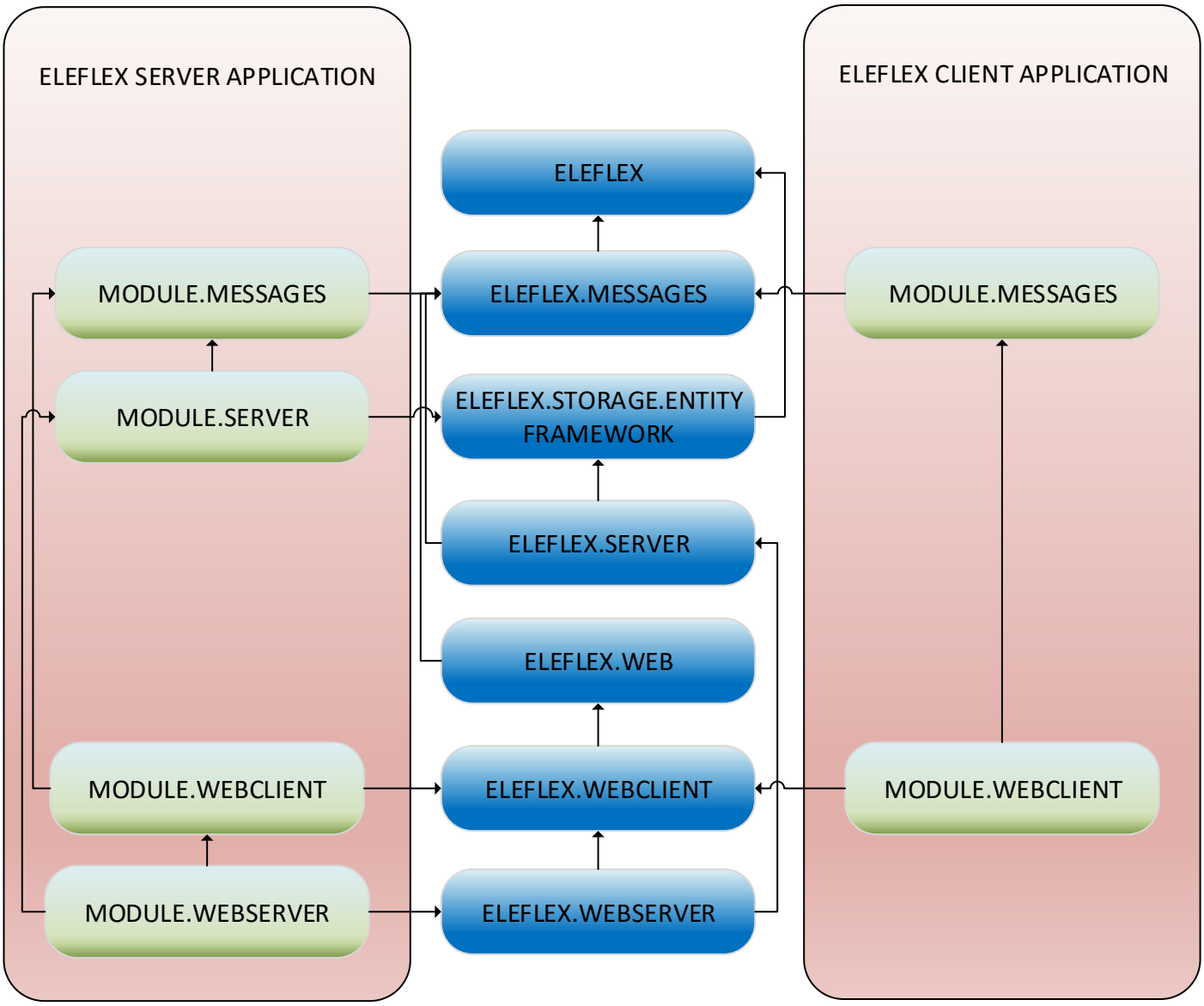

FIGURE 5- ELEFLEX MODULE RELATIONSHIPS

Implementation Details

This section outlines implementation details of the NuGet packaged applications.

Create an Eleflex.WebServer Application

Follow the steps below to create a new Eleflex.WebServer application.

1) Open Microsoft Visual Studio 2015 and create a new ASP.NET Web Application (Empty Template)
2) Right click your web application in the solution explorer and select "Manage NuGet Packages for Solution", search for "Eleflex.WebServer" and install the package.

   a. When prompted to overwrite any files, select “Yes to All”
3) Right click your web application in the solution explorer, select "Properties" and go to the "Web" tab on the left. Copy the port number of your web application.
   a. Update the web.config file for the system.servicemodel/client/endpoint address. Change the "localhost:16185" to be your web application's port number.
4) Open Microsoft SQL Management Studio and create a new database. Update the web.config file to change the connection string to connect to the database you just created.
5) Change the web.config file for the appSettings key of “EleflexImpersonateSystemToken” to be any new value.
   a. This allows system admin impersonation over services. This key should be changed before publishing your application.
6) Rebuild your web application and run it. The first user to register with the system receives the Admin role to administer the system.
   a. Go to the Login page and click the “Register New User” link. Enter your information and you will be logged in as an Admin of the system.

Create an Eleflex.WebClient Application

Follow the steps below to create a new Eleflex.WebClient application.

1) Open Microsoft Visual Studio 2015 and create a new ASP.NET Web Application (Empty Template)
2) Right click your web application in the solution explorer and select "Manage NuGet Packages for Solution", search for "Eleflex.WebClient" and install the package.
   a. When prompted to overwrite any files, select “Yes to All”
3) Update the web.config file for the system.servicemodel/client/endpoint address. Change the "localhost:16185" to be your Eleflex.WebServer’s port number.
4) Make sure that the web.config appSettings key for “EleflexImpersonateSystemToken” is the same as your Eleflex.WebServer’s app setting for the same key.
   a. This allows system admin impersonation over services. This key should be changed before publishing your application.
5) Rebuild your web application and run it. This web client application will run the same as the web server application, except that all calls for data will be sent to the web server application using WCF services.

Package Installed Files

Installing the Eleflex.WebClient or Eleflex.WebServer NuGet packages will modify the Global.asax and web.config files, as well as create several files in your web application. These files are used to configure the application to work with the platform and to start and stop it.

Global.asax

Both packages will update the Global.asax.cs file to add platform specific logic.

- Application_Start()
  - This method first load all assemblies into the application domain and create an instance of the Eleflex.SystemStartupShutdown object to call it's *Start()* method to start the platform.
- Application_End()
  - This method will create an instance of the Eleflex.SystemStartupShutdown object to call it's *Stop()* method to stop the platform.
- Application_Error()
  - This method will log an application error message calling the Eleflex.Logger object along with the HTTP server error that occurred.
- Application_EndRequest()
  - This method will commit and dispose the IStorageContextUnitofWork to release all transactions created during the course of the client web request. Since the client application only uses service's to communicate, this is only useful for the server application where storage methods may have been called during the request.

App_Start/Eleflex_Start Folder

The main differences between the Eleflex.WebClient and Eleflex.WebServer startup tasks is usually object location configurations. WebClient applications will register services that use ServiceRepository objects for service communication, whereas WebServer applications will register services that use BusinessRepository objects for direct access to data.

- OWINStartup (both)
  - This task configures OWIN and ASP .NET Identity security for the application.
- SystemLoggingStartupTask (both)
  - This task configures Logging for the application.
- SystemObjectLocationRegistrationTask (both)
  - This task configures object location configurations for the core system components of Logging, Security and Versioning.
  - This registers ServiceRequestDispatchers for service-based access
  - For Eleflex.WebServer, this additionally registers StorageServices to connect to underlying data persistence engines.
- SystemPatchStartupTask (Eleflex.WebServer only)
  - This task is used to patch the system for versioning.
- SystemShutdownTask
  - This task shuts down the system
- WebBundlesStartupTask
  - This task is used to create script and style bundles for the web application

- WebFiltersStartupTask
  - This task is used to create filters for the web application.
- WebRoutesStartupTask
  - This task registers MVC page routes for embedded web applications to be hosted in the main web application.

Web.config

- appSettings
  - Adds "webpages.Version" for MVC 5 configuration
  - Adds "enableSimpleMembership" to disable default ASP.NET Identity
  - Adds "EleflexImpersonateSystemToken" which allows impersonating the system account for service calls. This should be changed before deploying your web application!
- connectionStrings (Eleflex.WebServer only)
  - Adds the "EleflexDefault" database connection string used by all modules by default.
- system.net/mailSettings (Eleflex.WebServer only)
  - Adds default settings for sending email
- System.servicemodel/extensions and behaviors
  - EleflexCookieSecurityClientBehavior (Eleflex.WebClient only)
    - Adds behavior for sending impersonation token in service command header
  - EleflexCookieSecurityServerBehavior (Eleflex.WebServer only)
    - Adds behavior for receiving impersonation token or credentials to impersonate the system admin or a user during the service command request.
- System.servicemodel/bindings (Eleflex.WebClient only)
  - Adds bindings for the call to the web server application
- System.servicemodel/clients (Eleflex.WebClient only)
  - Adds the address to call the web server application
- System.servicemodel/services (Eleflex.WebServer only)
  - Adds the exposed service used by all service commands in the platform.

EleflexService.svc

This WCF service file is added to the root of the Eleflex.WebServer application and exposes the platform's WCF service commands for all modules.

Content Folder

- Eleflex.css
  - Specific themes for the default installed website.
- EleflexTheme.css
  - Bootstrap theme for the default installed website.

Controllers Folder

- EleflexHomeController.cs
  - Default controller for the web application. This can be changed by modifying the WebRoutesStartupTask.
- ErrorController.cs
  - Default controller for handling errors in the web application.
- Admin\AdminController.cs
  - Controller requiring the user to have the "Admin" role assigned to view.

Images/ProductionReady Folder

This contains Production Ready® images used for the default web application.

Scripts Folder

- Eleflex.js
  - This contains custom functions used in the website.
  - Load()
    - This will automatically configure datatables, select pickers, and date/datetime pickers and validators by using defined classes
  - eleflexClearInput()
    - This will clear input controls within a form.
  - eleflexGetRequestVerificationToken()
    - This will get the antiforgery token for use with ajax posts to the server.
  - eleflexHandleAjaxResponse()
    - This will handle an AjaxResponse from an ajax call and call success, warning, info or error messages as required.
  - eleflexShowSuccessMessage()
    - Display a success message.
  - eleflexShowErrorMessage()
    - Display an error message.
  - eleflexShowInfoMessage()
    - Display an info message.
  - eleflexShowWarnMessage()
    - Display a warning message.
  - eleflexDeleteMessages()
    - Remove all displayed messages.

Views Folder

- _ViewStart.cshtml
  - This configures all pages to use the EleflexDefaultLayout for the default application
- Admin Folder

    - Default views for the web application.
- EleflexHome Folder
    - Default views for the web application.
- Error Folder
    - Default views for the web application.
- Shared Folder
    - Default views for the web application.

Configuring an Eleflex.WebServer for Azure

Changing the web server application to use Microsoft Azure is simple. You will need to change the object location configurations for the storage services to use Azure instead of the default SQL Server configurations. Update the App_Start/Eleflex_Start/SystemObjectLocationRegistrationTask.cs class. For each storage service, change the constructor parameter for ISTORAGESERVICE_CONSTUCTORPARAM_VERSIONINGSTORAGECONFIG to use the Azure config for each library instead of SqlServer.

```
Eleflex.Logging.Storage.EF.SQLServer.LoggingSQLServerConstants.VERSIONING_STORAGE_CONFIG

Eleflex.Versioning.Storage.EF.SQLServer.VersioningSQLServerConstants.VERSIONING_STORAGE_CONFIG)

Eleflex.Security.Storage.EF.SQLServer.SecuritySQLServerConstants.VERSIONING_STORAGE_CONFIG
```

Should be changed to:

```
Eleflex.Logging.Storage.EF.Azure.LoggingAzureConstants.VERSIONING_STORAGE_CONFIG

Eleflex.Versioning.Storage.EF.Azure.VersioningAzureConstants.VERSIONING_STORAGE_CONFIG

Eleflex.Security.Storage.EF.Azure.SecurityAzureConstants.VERSIONING_STORAGE_CONFIG
```

Web Processes

As of the time of writing this document, Microsoft Azure does not support .NET 4.5.2 and the QueueBackgroundWorkItem object. The ***Eleflex.EleflexWebProcess*** object uses a simple timer object to routinely run a background process until the Microsoft Azure platform supports an upgraded framework.

This example shows how to create a startup task that creates a background web process.

```
public class ExampleProcessStartupTask : StartupTask
{
        protected EleflexWebProcess _exampleProcess = null;

        public ExampleProcessStartupTask() : base()
        {
            Description = @"This task starts the example background process.";
            Priority = StartupConstants.PRIORITY_CUSTOM;
        }

        public override bool Start(ITaskOptions taskOptions)
        {
            if (_exampleProcess == null)
            {
                _exampleProcess = new EleflexWebProcess(60000, //Run every minute (high priority)
                    () =>
                    {
                        try
                        {
                            //Do work here
                        }
                        catch { }
                    });
            }
}
```

Creating a new ELEFLEX Module

This section contains instructions on using the Eleflex.ModuleGenerator package to generate a complete Visual Studio 2015 solution, projects and code files for a new module.

Overview and Quickstart

1. Install the Eleflex.ModuleGenerator package to your Eleflex.WebServer and navigate to "Admin/ModuleGenerator"
2. Enter form fields and click "Generate" to download a zip file contains your solution files.
3. Extract the zip file to a directory and open the solution in Visual Studio 2015. Rebuild the solution to download missing NuGet packages and confirm the build is successful
4. In the root of the Server project, create a new ADO.NET Entity Data Model with the name from above, as in: [Entity Model Name].edmx
5. Right click and run all T4 text templates (*.tt) in the Business, Messages, and Server projects.
6. In the WebServer project, uncomment the WebServerObjectLocationRegistrationTask.cs. Rebuild the entire solution to confirm the build is successful
7. Create a new web application in your solution, right click the project, select "Manage NuGet Packages" and install the Eleflex.WebServer package. Complete setup of Eleflex.WebServer application and verify working

8. In the new Eleflex.WebServer application, click "References" and "Add a Reference" to all module projects in the solution
9. Rebuild and run the new web application. If the module is installed correctly, a default webpage should be available at the URL "/Admin/[Module Name]".

Generating the Module Template

We have developed a simple template-based solution for creating modules in the platform. To begin, install an Eleflex.WebServer and follow the instructions to complete setup. Next, add the "Eleflex.ModuleGenerator" NuGet package. Including this package will configure a web application route to "Admin/ModuleGenerator." Rebuild and run your solution, and navigate to "Admin/ModuleGenerator".

The module generator requires 3 pieces of information to create a new module.

- Module name
  - This is the general name of your module. If you were building a module to store names and addresses of people, it might be called "Contacts." If you were building a module to store employee clock ins and outs, it might be called "TimeClock."
- Namespace Prefix
  - This is usually the company name or product name. The resultant base namespace of the solution will be [Namespace Prefix].[Module Name]
- Entity Framework Data Model Name
  - This is the entity framework data model name. By default, the name is "[Module Name]DB". Click the checkbox to override the name you plan on using.

After entering the required information, click the "Generate" button to download a zip file of the generated solution. The zip file will contain the following files.

- Visual Studio 2015 solution file
- Business Project - core business logic and interfaces
- Messages Project – service models and interfaces
- Server Project – service commands and data access classes
- Web.Admin Project – embeddable web application for "Admin" related views
- WebClient Project – classes for Eleflex.WebClient installations
- WebServer Project – classes for Eleflex.WebServer installations

Extract the files to a folder and open the solution. Perform a rebuild on the solution so that it can download missing NuGet packages and confirm that the build is successful.

Add the Entity Data Model to Run the T4 Text Templates

The module we build will require a database to store information. In the default Eleflex database, create schemas, tables, stored procedures, views and data that is related to your module. For simplicity of this exercise, create a new table called “dbo.Contact” with an integer primary key called “ContactKey” and another string column for “Name.”

The T4 text templates in the generated projects are driven based on an Entity Framework Data Model. When running the templates, it will iterate through the objects in the entity data model and create code files in the project to fulfil the design tiers required by the architecture. When the module generator was run, an Entity Data Model Name was defined in the user interface, the default name is “[Module Name]DB.” This will be the name of the model created.

1. In the root of the Server project, create a new ADO.NET Entity Data Model with the name from above, as in: “[Module Name]DB.”
2. Right click and “Run Custom Tool” for all T4 text templates (*_CodeGenerator.tt files) in the Business, Messages, and Server projects.
3. In the WebServer project, uncomment the WebServerObjectLocationRegistrationTask.cs.
    a. This file was commented out because it relies on objects that need to be generated by the T4 text templates
4. Rebuild the solution to confirm successful

Verifying Module Setup

In order to verify your module is setup correctly, an admin web application was created with your module to demonstrate the embedded web application and also provide a starting point for your module.

1. Create a new web application in your solution, right click the project, select "Manage NuGet Packages" and install the Eleflex.WebServer package. Complete setup of Eleflex.WebServer application and verify working
2. In the new Eleflex.WebServer application, click "References" and right click to "Add a Reference" to all module projects in the solution
3. Rebuild and run the new web application. If the module is installed correctly, a default webpage should be available at the URL "/Admin/[Module Name]".

Copyrights, Licenses, Trademarks and Logos

Portions of this document contain source code and is licensed under the GNU General Public License. A copy of this license is included in the section entitled "GNU General Public License".

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```


GNU General Public License

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether

gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents.

States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source

form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically

linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do

not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This

License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no

further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user

actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for

the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work,

for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on

those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third

paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission

to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation

(including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent

(such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you

to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software

Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

GNU Free Documentation License

GNU Free Documentation License

Version 1.3, 3 November 2008

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely

available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these

copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.

- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections

of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.