

Gotchas ...

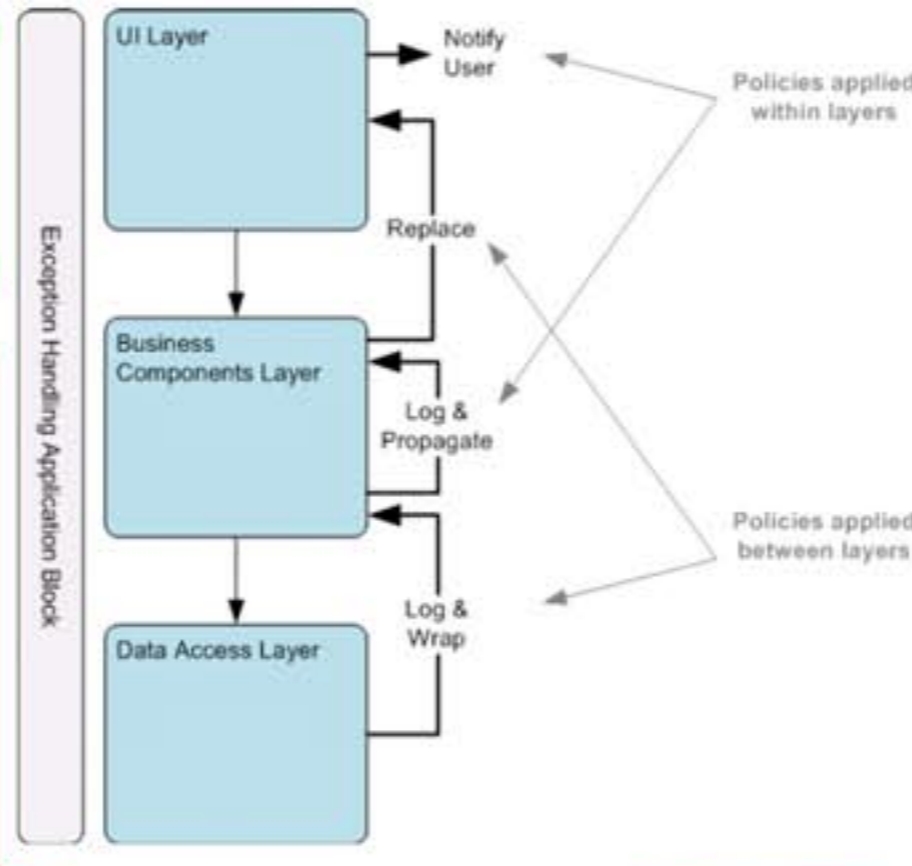
- 1. Configuration AB**
 - ✓ Avoid configuration sections with spaces in the name.
- 2. Exception AB**
 - ✓ Exception Policy handlers are executed in the sequence in which they are defined. If you want to log and wrap, for example, be sure to configure logging first.

Exception Sample Code

```
catch ( Exception ex )
{
    if ( ExceptionPolicy.HandleException ( ex, policy ) )
    {
        throw;
    }
}
```

HandlingInstanceID

- Exception Access Block**
1. Open solution ExceptionDemo and show:
 2. Add an App.config application configuration file
 3. Add references to Enterprise Libraries Configuration Logging Exception
 5. Configure Solution for EL
 - Add new Exception Handling Application Block Config
 - Add new Logging and Instrumentation Application Block Config
 - Add Exception policies
 6. Add exception "catch" code



Logger Sample Code

```
using ( new Tracer("Trace") ) // Trace activity
{
    LogEntry logEntry = new LogEntry();

    // Standard Properties
    logEntry.Message = "Log me now.";
    logEntry.Category = "General";
    logEntry.Priority = 3;
    logEntry.Severity = Severity.Information;
    logEntry.EventId = 55;
    logEntry.Title = "Log Entry";

    // Extended Properties
    Hashtable dictionary = new Hashtable();
    string resolution = string.Format("{0}x{1}",
        Screen.PrimaryScreen.Bounds.Width,
        Screen.PrimaryScreen.Bounds.Height);
    dictionary.Add("Screen resolution", resolution);
    logEntry.ExtendedProperties = dictionary;

    // Log Event
    Logger.Write(myLogEntry);
    Logger.Write("Log me all in one line",
        "General", 2, 56,
        Severity.Information);
}
```

- Configuration File**
1. Setup application configuration file (app.config)
- Logging Code**
1. Reference Logging and Logging.Tracing
 2. Instantiate a LogEntry object
 3. Set Properties, i.e. message, category, severity, ...
 4. LoggerObject.Write (... overloaded ...)
- Extend Logger**
1. Set Extended Properties using a Hashtable
- Asynchronous**
1. Specify private queue (only one per machine)
 2. Setup MSMQ Distributor Service (See ReadMe)
- Notes:**
1. MSMQ Distributor Services uses own config file in Enterprise Library bin directory

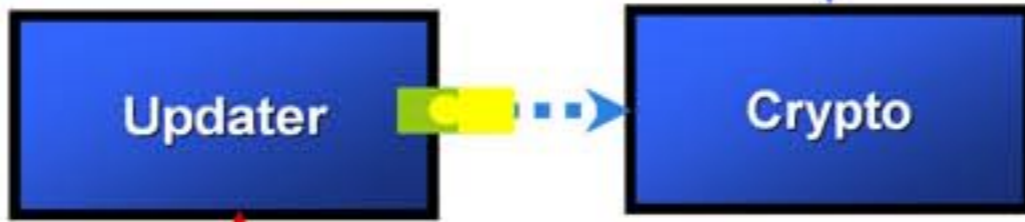
Updater II Block Sample Code

```
private void Form1_Load(object sender, System.EventArgs e)
{
    updater = ApplicationUpdaterManager.GetUpdater();
    updater.DownloadCompleted += new DownloadCompletedEventHandler(updater_DownloadCompleted);

    Manifest[] manifests = updater.CheckForUpdates();
    if (manifests.Length > 0)
    {
        foreach (Manifest m in manifests)
        {
            m.Apply = true;
        }
        updater.Download(manifests, TimeSpan.FromMinutes(2));
    }
}

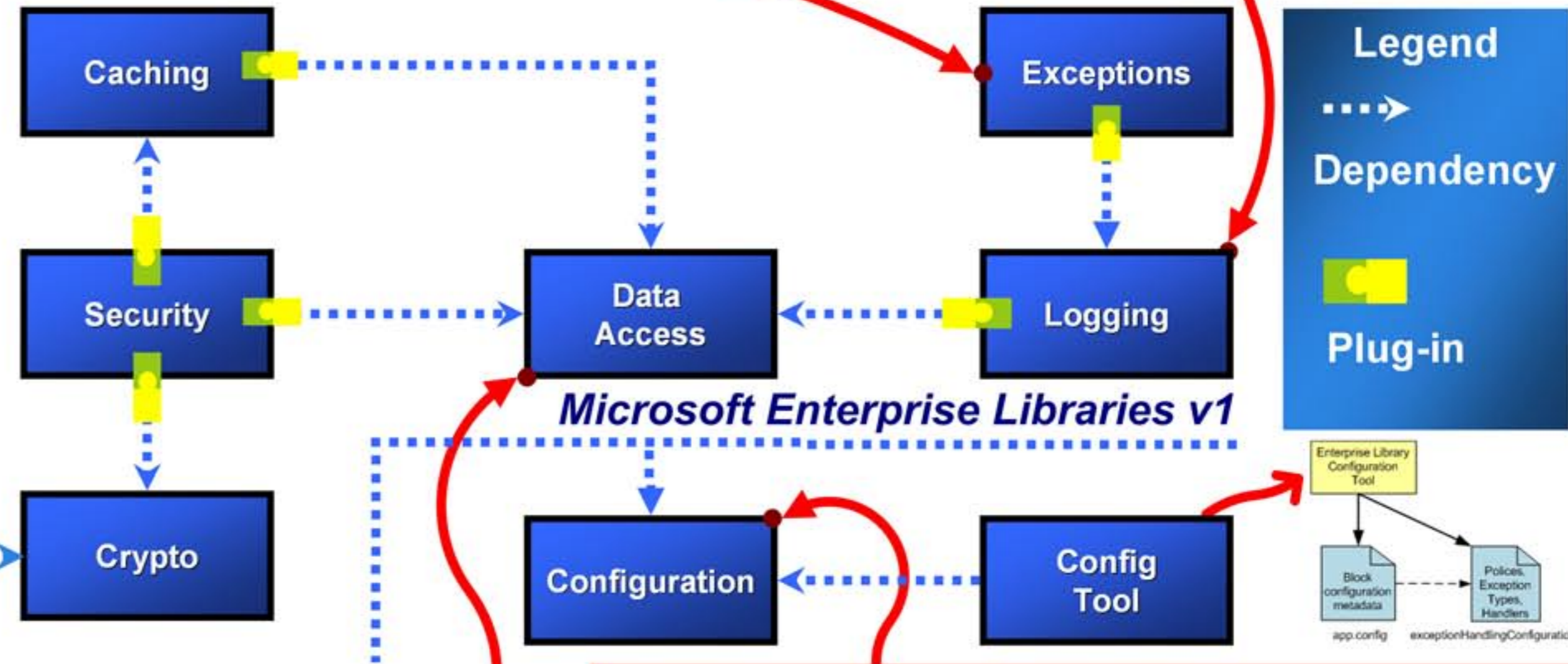
private void updater_DownloadCompleted(object sender, ManifestEventArgs e)
{
    updater.Activate(new Manifest[] { e.Manifest });
}
```

1. Add a normal App.config file
2. Add references to:
 - Microsoft.ApplicationBlocks.Updater.dll
 - Microsoft.ApplicationBlocks.Downloaders.dll
 - Microsoft.Practices.EnterpriseLibrary.Common.dll
 - Microsoft.Practices.EnterpriseLibrary.Configuration.dll
 - Microsoft.Practices.EnterpriseLibrary.Security.Cryptography.dll
 - Microsoft.ApplicationBlocks.Updater.ActivationProcessors.dll
3. For MSI update packages the following dll must be referenced: Interop.WindowsInstaller.dll
4. Configure App.Config
 - Add Updater Application Block Config section.
 - Add Bits Downloader.
5. Configure Manifest
 - Add manifest id (must be GUID) and description.
 - Add application information.
 - Add activation processors.



1. Add Application Configuration File
2. Add References (Configuration, Common, Data)
3. Configure
 - Connection Strings (String Used To Connect To Database)
 - Database Type (Type of database Used MSSQL, Oracle or DB2)
 - Database Instance (Name Used By Factory, Uses Connection string and Database Type)
4. Add Using Microsoft.Practices.EnterpriseLibrary.Data
6. Create Database Using Factory With Instance Name
6. Create Command From Database Instance
7. Set Parameters
8. Execute Command Against Database
9. Read Results (DataReader, Return Parameters)

```
Database db = DatabaseFactory.CreateDatabase("Northwind");
DBCommandWrapper command =
    db.GetStoredProcCommandWrapper("CustOrdersOrders");
command.AddInParameter("@CustomerID", DbType.String, "QUICK");
using (IDataReader dataReader = db.ExecuteReader(command))
{
    while (dataReader.Read())
    {
        Console.WriteLine("OrderID: {0} ", dataReader.GetInt32(0));
    }
}
```



1. Create application configuration file: normal app.config
 2. Add Configuration Block section using EL Configuration GUI by pointing to the app.config.
 3. Name the Configuration section (X)
 4. Create a configuration state class, exposing properties for all configurable attributes
 - A. Populate configuration section (X) using method WriteConfiguration. i.e. ConfigurationManager.WriteConfiguration("X", stateClass);
 - B. Query configuration section (X) using method ReadConfiguration. i.e. StateClass stateClass = (StateClass)ConfigurationManager.GetConfiguration("X");
- Notes:**
1. Configurator "polls" for config changes, resulting in a 1.5+ second lag in detection if the delegate:

```
// Initial Data -- typically in setup program
ConfigurationManager.WriteConfiguration ("DemoConfig",config);
// Register on changed event
ConfigurationManager.ConfigurationChanged += new
    ConfigurationChangedEventHandler(ConfigurationManager_ConfigurationChanged);
// Retrieve configuration
Configuration config = ConfigurationManager.GetConfiguration("DemoConfig") as Configuration;
// Update state and write configuration
config.RunCount++;
ConfigurationManager.WriteConfiguration("DemoConfig",config );
// Link to configuration changed event to reload configuration
ConfigurationManager.ConfigurationChanged +=new
    ConfigurationChangedEventHandler(ConfigurationManager_ConfigurationChanged);
```