



Distributed Geodatabase for Developers (Best Practices)

Khaled Hassen
Gary MacDougall

Prerequisites

- Prerequisites
 - Understand versioning
 - Have reviewed the geodatabase replication [on-line help](#) and [developer documentation](#)
- All code examples and demos are in C#

Overview

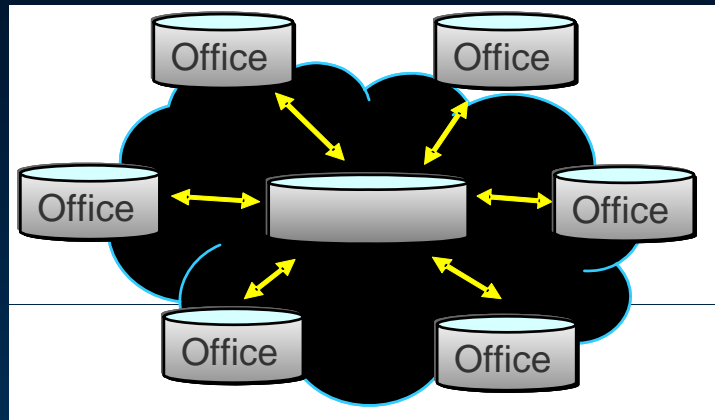
- **Distributed geodatabase use cases and techniques**
- **Geodatabase replication**
 - Replication introduction
 - Replication internals and developer API
 - DEMO: Replica creation with custom workspace extension
- **Interoperability with non-geodatabase formats**

Distributed Geodatabase Use Cases

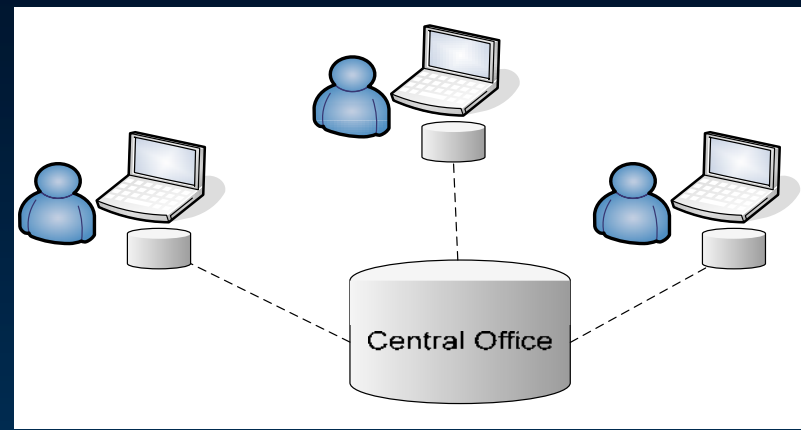
- **Mobile Users and Field Crews who need to be disconnected from the network**
- **Users who need to maintain copies of data at different organizational levels (city, county, state)**
- **Users who want to maintain copies of data at different geographic facilities**
- **Users who need to distribute work to contractors**
- **Production and publication geodatabases**
- **Fail over systems**

Distributed Geodatabase Use Cases

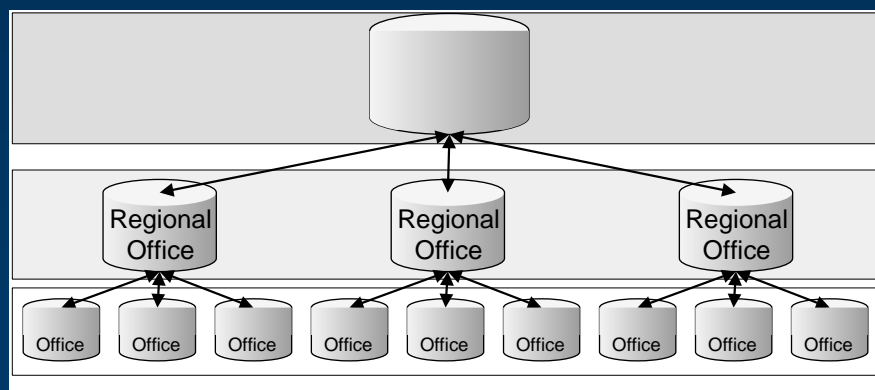
Regional Offices



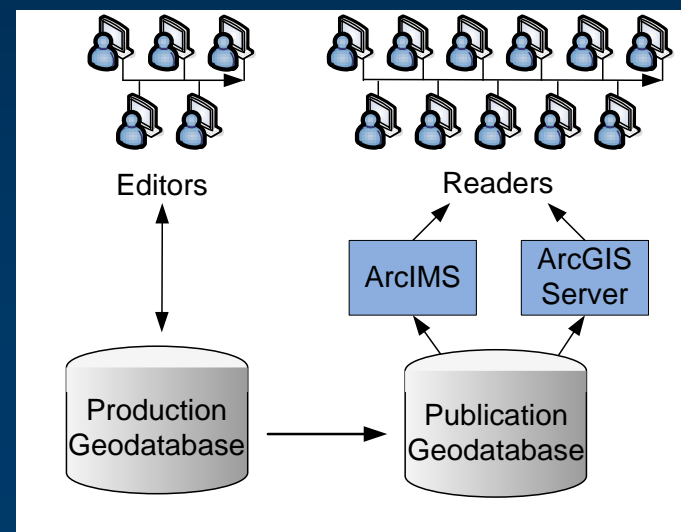
Mobile Users



Multiple levels



Production / Publication



Data Copying and Loading

- **Involves simply using data copying and export/import tools to copy data from one geodatabase to another**
- **Works well for systems with simple requirements**
 - **Example: A field worker updates a feature class and simply needs to copy that feature class to the ArcSDE geodatabase in the office each night**
- **Limitations**
 - **Custom solution**
 - **No safeguard against data lose**
 - **No safeguards against data redundancy**

DBMS Replication with Geodatabases

- **Requirements and limitations**

- Requires knowledge of how the geodatabase\ArcSDE system tables work
- No tools provided in ArcGIS to support it
- Limited support for cross DBMS replication
- Does not support or has limited support for complex geodatabase data types and limited filters to define the data to replicate

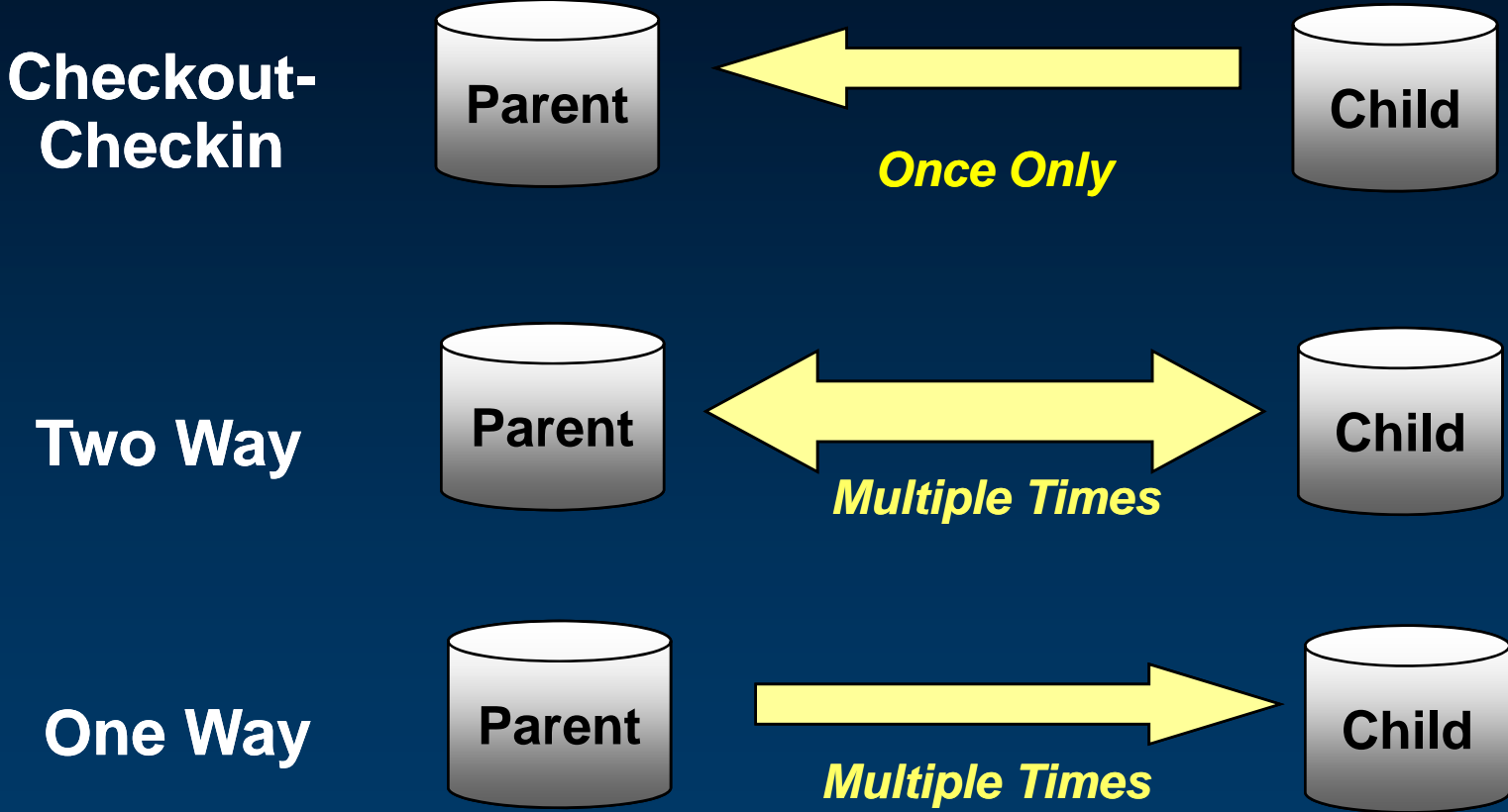
- **Advantages**

- Can work with non-versioned data
- Can replicate entire database
- Can be configured to provide synchronous replication

Geodatabase Replication

- **Allows you to distribute copies of data across 2 or more geodatabases**
- **You can edit the databases independently and synchronize them as needed**
- **Uses geodatabase connections directly (LAN) or using geodata services publish on ArcGIS server (WAN)**
- **Works in a connected and a disconnected environment**
- **Can replicate subsets and uses versioning**
- **Released at 9.2 - Builds upon disconnected editing from earlier releases (8.3)**

Geodatabase Replication Types

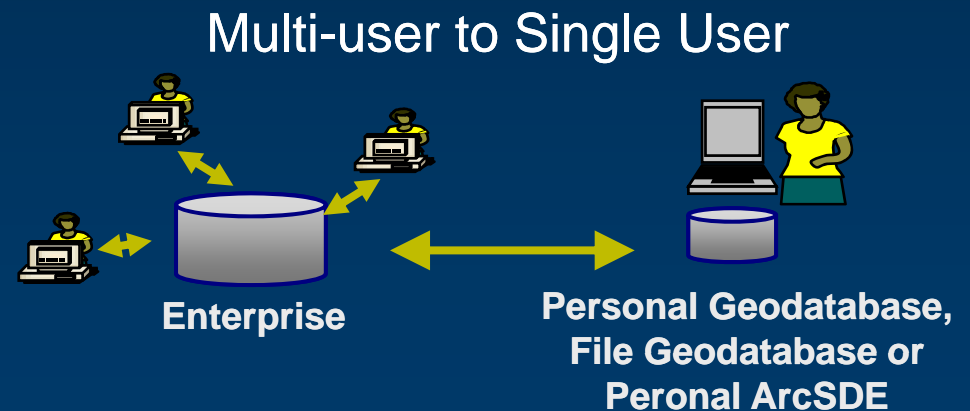
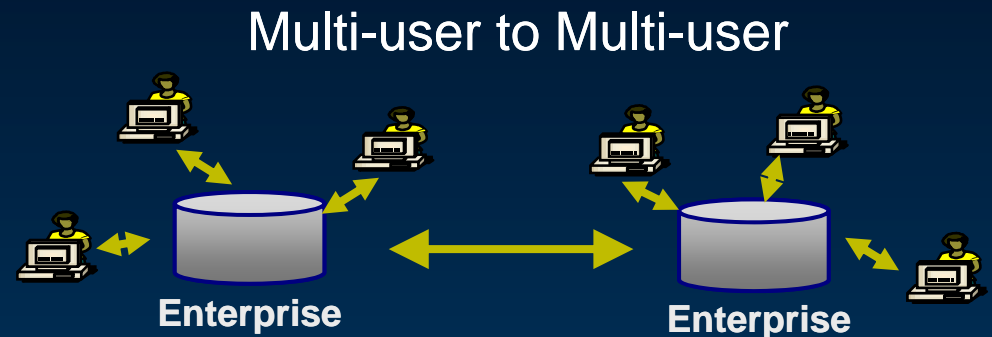


Geodatabase Replication Workflows

- Workflows can involve Multi-User geodatabases and single user geodatabases

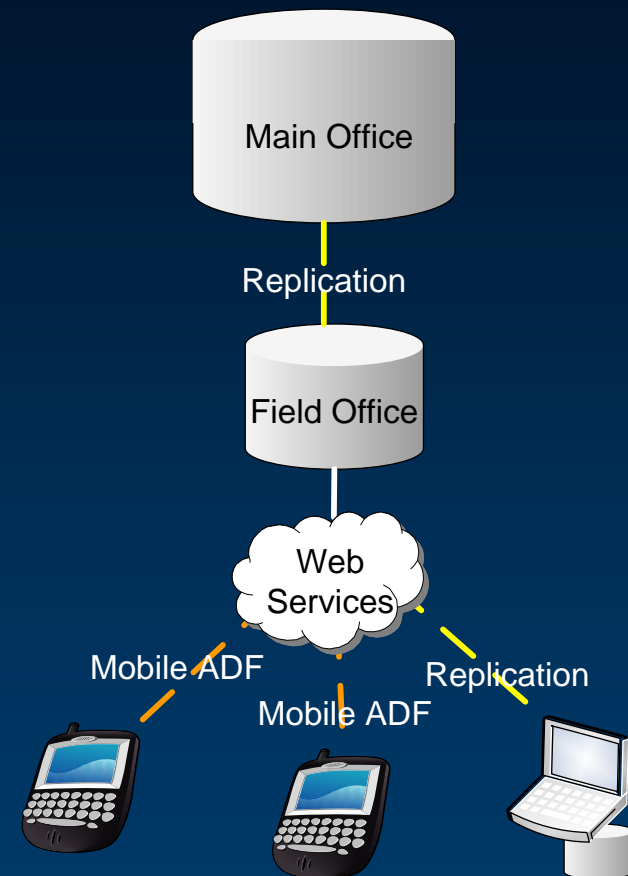
- Multi-user geodatabase – Multi-user ArcSDE geodatabase accessed locally or remotely through ArcGIS server

- Single user geodatabase – Personal ArcSDE, file geodatabase or personal geodatabase on a local machine



Distributing Data

- **Geodatabase Replication can be used in conjunction with other data distribution techniques**
- **Example:**
 - **Use geodatabase replication to synchronize changes between offices**
 - **Use mobile ADF for field workers with light weight mobile devices**
 - **Use geodatabase replication for field workers who need ArcGIS desktop or engine in the field**



Geodatabase Replication – Getting Started

- **Anticipate future needs when defining the data to replicate**
- **Have a well defined data model before creating replicas**
- **Choose the right replica type**
 - **Consider 2 way replicas with personal ArcSDE instead of check-out replicas**
 - **Use 1 way replicas over 2 way replicas when possible**

Geodatabase Replication – Getting Started

- **Use models or scripts for replicas you plan to create on a regular basis**
 - You can use the `create replica` and `create replica from server` geoprocessing tools to build models
- **Consider using the following replica creation options**
 - Re-use schema (check-out replicas) – uses existing schema
 - Register only – replicates pre-copied data
 - Relationship classes processing is optional
- **Schedule Synchronizations**
 - You can use geoprocessing models exported to python and the windows scheduler
 - Consider synchronization order

Geodatabase Replication – Getting Started

- **Integrate synchronization with version management strategy**
 - Synchronize before running reconcile and compress services
 - Reconcile service should cover replicas with a manual reconcile policy
- **Network speed**
 - Use geodatabases directly over fast networks (LAN)
 - Use ArcGIS server and geodata services on slower networks (WAN – i.e. DSL)
 - Use disconnected synchronization techniques over very slow networks (slow dial-up modem) or where there is no network connectivity

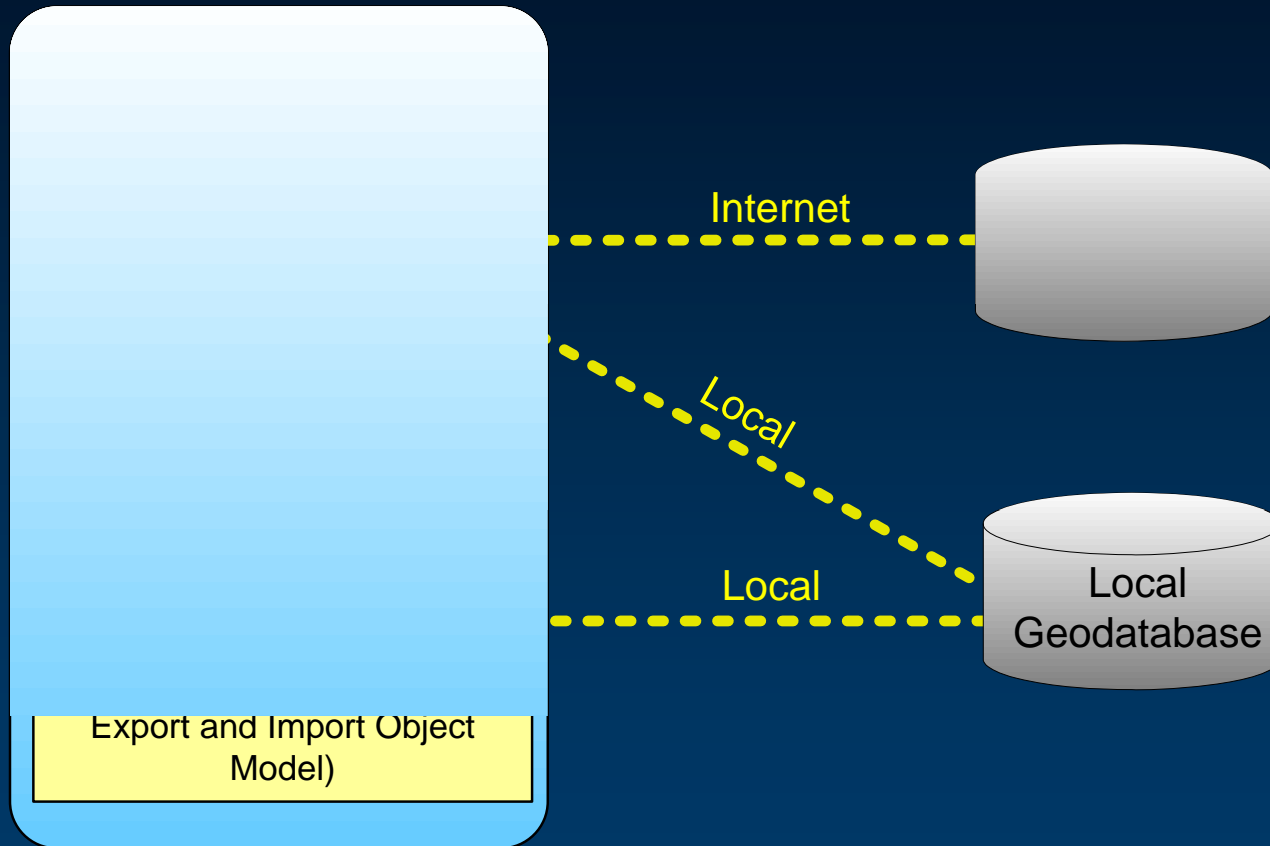
When to write code

- **Basic creation and synchronizations using geoprocessing models require little or no coding**
- **Write code when...**
 - **Integrate geodatabase replication into larger applications**
 - **Example: integrate synchronization with reconcile service**
 - **Extend the replica creation and synchronization process**
 - **Methods and properties not exposed through the UI or geoprocessing tools**
 - **Recommended synchronization order**
 - **Browse changes before synchronization**
 - **Advanced replica creation options**
 - **Export and Import version differences**

Geodatabase Replication – API

- **GeoDatabaseDistributed library**
- **Coarse Grained API (*Recommended*)**
 - **GeoDataServer Object Model – new at ArcGIS 9.2**
 - **Stateless object model**
 - **Geodatabase replication, browsing, querying, data extraction**
- **Fine grained API**
 - **Pre-ArcGIS 9.2 object models**
 - *Data Extraction and Check out/Check in Object Model, XML Export and Import Object Model*
 - **Object models for specialized operations such as browsing the contents of a delta file**
 - *Data Changes Object Model, Schema Change Export and Import Object Model*

Geodatabase Replication – API

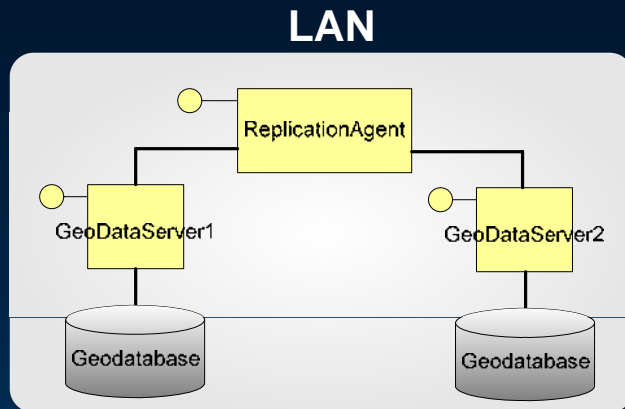


Supports Connected and Disconnected Environments

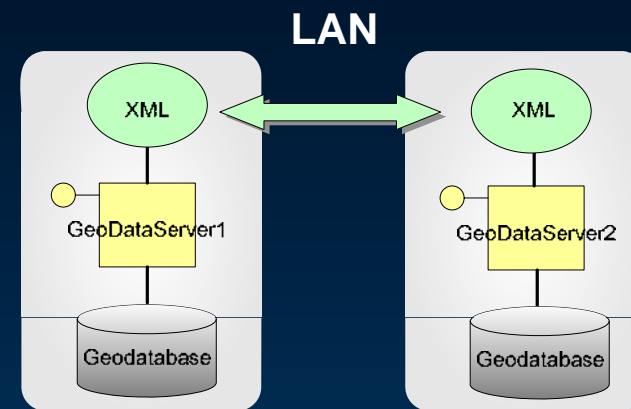
- **Connected**
 - All replicas accessible on the network (LAN or WAN)
 - Always connected or intermittently connected
 - Example: to synchronize, use the synchronize wizard in ArcCatalog
- **Disconnected**
 - Replicas are not on the same network
 - Message exchange is performed by the end user
 - Operations are performed by export, file transfer and import
 - Example: to synchronize, export changes to a delta XML file, transfer the file (ftp, CD through the mail, etc.), have the file imported on the relative replica when it arrives, send acknowledgement message back to sender

Connected and Disconnected Environments

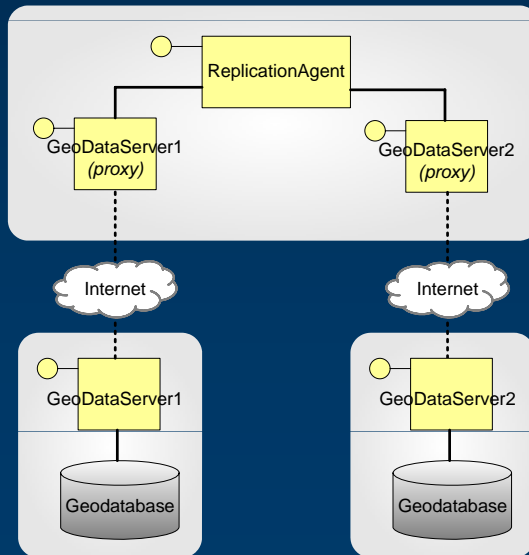
Connected



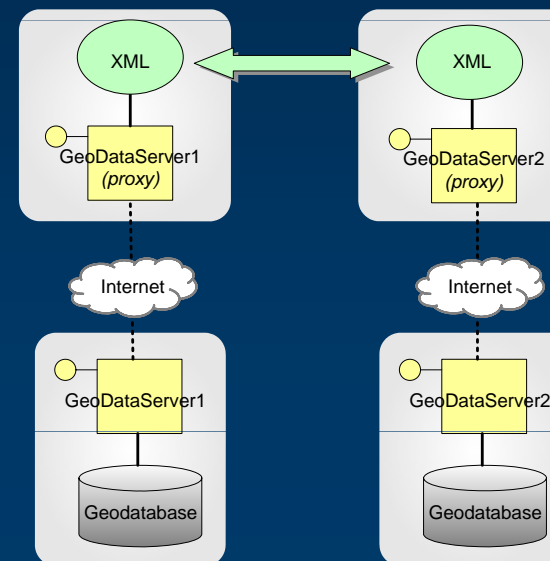
Disconnected



WAN (ArcGIS Server)



WAN (ArcGIS Server)



Geodatabase Replication API

- **Replica creation**
- **Replica synchronization**
- **Replication extensibility**
- **Optimization, performance and best practice**

Example: Create Replica in a Connected Environment

- **Steps**

- ① **Initialize a `GeoDataServer` for the source and target geodatabases**
- ② **Create a `GPReplicaDescription` to define the data to be replicate from the source**
- ③ **Use the `GPReplicaOptionsClass` to define replica options such as the replica type**
- ④ **Use the `ReplicationAgent` to create the replica**

Initializing GeoDataServers

① GeoDataServers represent local or remote geodatabases

- Local geodatabases are initialized directly
- Remote geodatabases are accessed from geodata services published by ArcGIS server on the internet

Initializing from a local geodatabase

```
IGeoDataServer iGDS = new GeoDataServerClass();
IGeoDataServerInit iGDSInit = iGDS as IGeoDataServerInit;
iGDSInit.InitFromConnectionString("SERVER=bobmk;INSTANCE=5151;
    VERSION=sde.DEFAULT;USER=gdb;PASSWORD=gdb");
```

Accessing a geodata service on the internet

```
IPropertySet iCProps = new PropertySetClass();
iCProps.SetProperty("URL",@"http://baza/arcgis/services");
IAGSServerConnectionFactory iSCF = new AGSServerConnectionFactoryClass();
IAGSServerConnection iSvrConn = iSCF.Open(iCProps, 0);
IAGSServerObjectName iSObjName = null;
While ((iSObjName = iSvrConn.Next()) != null) {
    if (iSObjName.Name == "Sample_GeoDataService") return iSObjName; }
```

Define the data to be replicated

② Create a GPReplicaDescription to define the data to be replicated

```
public IGPReplicaDescription CreateReplicaDescription(IGeoDataServer iGDS)
{
    IGPReplicaDescription iGPRDescription = new GPReplicaDescriptionClass();
    IGPReplicaDatasets iGPRDatasets = new GPReplicaDatasetsClass();
    IGPReplicaDataset iGPRDataset = new GPReplicaDatasetClass();
    iGPRDataset.Name = "Roads"; iGPRDataset.Type = "esriDTFeatureClass";
    iGPRDatasets.Add(iGPRDataset);

    try {
        IGPReplicaDatasets iERDatasets =
            iGDS.ExpandReplicaDatasets(iGPRDatasets);
    }
    catch (Exception e) { return null; }
    iGPRDescription.GPReplicaDatasets = iERDatasets;
    iGPRDescription.SingleGeneration(false);
    ...
    return iGPRDescription;
}
```

Set the replica options

- 3 Use the `GPReplicaOptionsClass` to define replica options such as the replica type

```
// Set the replica options
IGPReplicaOptions iReplicaOptions = new GPReplicaOptionsClass();

// Set type to 2 way replica
iReplicaOptions.AccessType = esriReplicaAccessType.esriReplicaBothReadWrite;

// Important only for replicas in disconnected environments
iReplicaOptions.IsChildFirstSender = true;

// Copy the data during creation
iReplicaOptions.RegisterReplicaOnly = false;
```

Create the replica

④ Create the replica using the ReplicationAgent

```
CreateReplicaConnected(iGDSParent, iGDSCild, iRepDescription,
    iReplicaOptions)
...
public void CreateReplicaConnected(IGeoDataServer iGDSParent, IGeoDataServer
    iGDSCild, IGPREplicaDescription iRepDesc, IGPREplicaOptions iRepOptions)
{
    String rVersion    = "MyWork",
        rName         = "MyReplica;
    IReplicationAgent iRepAgent = new ReplicationAgentClass();

    try {
        iRepAgent.CreateReplica(rVersion, iGDSParent, iGDSCild, rName,
            iRepDesc, iRepOptions);
    }
    catch (Exception e) {...}
}
```

See [How to create a replica in a connected environment](#)

Example: Create Replica in a Disconnected Environment

- **Steps**

- ① **Create the replica to a transport file**

- initialize geodataserver, create replicadescription, set replica options (as shown in connected)
 - Use the `GDSEExportOptionsClass` to define export options such as output format
 - Use the `GeoDataServer` to create the replica to an output file

- ② **Transfer the transport file to the target**

- ③ **Import the transport file on the target to complete replica creation**

Create the replica to a transport file

① Create the replica to a transport file from source

```
CreateReplicaDisconnected(GDSParent, RepDescription, ReplicaOptions)
...
public void CreateReplicaDisconnected(IGeoDataServer iGDSParent,
    IGPReplicaDescription iRepDesc, IGPReplicaOptions iReplicaOptions)
{
    String rVersion    = "MyWork",
        rName         = "MyReplica;
    esriGDSTransportType trType = esriGDSTransportTypeUrl;
    IGDSExportOptions iGDSEOptions = new GDSExportOptions();
    iGDSEOptions.ExportFormat = esriGDSEExportFormat.esriGDSEExportFormatXml;
    iGDSEOptions.Compressed = true;
    iGDSEOptions.BinaryGeometry = false;
    try {
        IGDSData iGDSDData = iGDSParent.CreateReplica(rVersion, rName,
            iRepDesc, iReplicaOptions, iGDSEOptions, esriGDSTransportTypeUrl);
    }
    catch (Exception e) {...}
}
```

Transport the file and Import

② File can transported manually or automatically

- Manual example: burn file to CD and send through US Mail
- Automated example: use a data distributed service to FTP files on a regular basis

③ Importing the transport file on the target

- Initialize a GeoDataServer for the target and use `ImportData` method

```
// reference the file to import with the GDSDData object  
IGSDData iGDSDData = new GDSDDataClass();  
iGDSDData.TransportType = esriGDSTransportType.esriGDSTransportTypeURL;  
iGDSDData.Compressed = true;  
iGDSDData.URL = @"D:\filestoimport\myreplica.zip";  
// Import the replica workspace document to create the replica  
iGDSSchild.ImportData(iGDSDData,  
    esriGDSImportFormat.esriGDSImportFormatXmlWorkspace)
```

See [How to create a replica in a disconnected environment](#)

Creating Replicas - Best Practices

- **Define your replica description for what you want to synchronize**
 - **Replica schema synchronization only supports adding simple feature classes and tables**
- **Use replica registration option for large dataset replica creation**
- **Getting related objects is expensive so use this option only when it is necessary**
- **Use GeoDataServer instead of ReplicationAgent when creating checkout to a new Personal GDB or FileGDB**

Creating Replicas - Replica Creation Events

- Developers may wish to extend the core replication behavior
- The replication creation process can be extended by implementing a Workspace extension that supports `IWorkspaceReplicaEvents`
- See [About Workspace Extensions](#)

```
Interface IWorkspaceReplicaEvents : IUnknown
{
    HRESULT BeforeCreatingReplicaEvents(BSTR replicaName,
                                         esriReplicaType rType,
                                         IReplicaDescription* iRDesc,
                                         IWorkspace* childWS);

    HRESULT AfterCreatingReplicaEvents(BSTR replicaName,
                                        esriReplicaType rType,
                                        IReplicaDescription* iRDesc,
                                        IWorkspace* iChildWS);
};
```

WorkspaceExtension With ReplicaEvents

- WSExtension needs to support IWorkspaceExtensionControl, IWorkspaceExtension and IWorkspaceReplicaEvents interfaces
- WSExtension needs to be registered as COM object (regasm.exe, regsvr32.exe)
- WSExtension needs to be registered with the workspace

```
public void RegisterExtension(IWorkspace iWS, String extGUID)
{
    IWorkspaceExtensionManager iWSMgr = iWS as IWorkspaceExtensionManager;
    UID uid = new UID();
    uid.Value = extGUID;
    IWorkspaceExtension iwSExt = iWSMgr.FindExtension(uid);
    if (iwSExt == null) {
        try { iWSMgr.RegisterExtension(uid); }
        catch (COMException ce) {...}
    }
}
```

Altering Replica Definition

Replica description can be changed after creating a replica

```
public void AlterReplica(IWorkspace iWS, string tableName, string dbase,
    string owner
) {
    IWorkspaceReplicas iWR = iWS new IWorkspaceReplicas;
    IReplica iReplica = iWS.get_ReplicaByName("MyReplica");
    IReplicaDescription iRD = iReplica.Description;
    int index = iRD.FindTable(tableName);

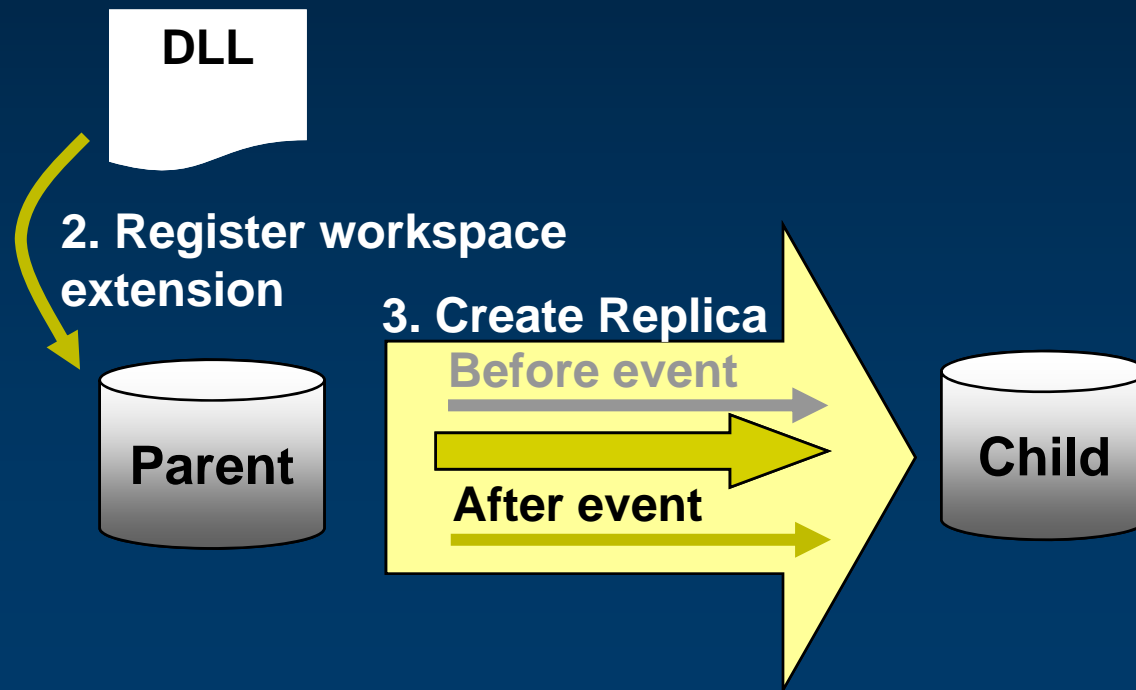
    IReplicaFilterDescriptionEdit iFD = iRD as IReplicaFilterDescriptionEdit;
    iFD.put_RowsType(index, esriRowsType.esriRowsTypeAll);

    IWorkspaceReplicasAdmin iRAdmin = iWR as IWorkspaceReplicasAdmin;
    try {
        iRAdmin.AlterReplica(iReplica);
    }
    catch (COMException ce)
    {
        ...
    }
}
```

Replica Creation Demo

- Build a custom workspace extension to augment replica creation with non-versioned tables using C#

1. Build workspace extension



Synchronizing Replicas

- **Replica can be synchronized in connected and disconnected environments**
- **Synchronize replica changes using ReplicationAgent requires specifying a sync direction (Child->Parent, Parent->Child or Both)**
- **Use the GeoDataServer to synchronize replica changes from delta Updategram (xml, fileGDB, or Personal GDB)**
- **Explicit versus implicit acknowledgment during replica synchronization**
- **Replica synchronization does not allow synchronizing out-of-order messages (e.g., Sync(Gen1)->Sync(Gen2), Sync(Gen4))**

Example: Synchronize Replicas in a Connected Environment

- Steps

- ① Initialize a GeoDataServer for the source and target geodatabases
- ② Use the ReplicationAgent to Synchronize the replica

```
// Sync the replica  
IReplicationAgent iRepAgent = new ReplicationAgentClass();  
iRepAgent.SynchronizeReplica(iParentGDS, /* Parent geodataserver */  
    iChildGDS, /* child geodataserver */  
    iGPReplicaParent, /* Parent replica */  
    iGPReplicaChild, /* Child replica */  
    esriRAResolveConflictsInFavorOfReplica1, /* Reconcile policy */  
    esriReplicaSynchronizeBoth, /* Sync Direction */  
    true); /* true = column level conflicts detection */
```

See [How to synchronize a replica in a connected environment](#)

Example: Synchronize Replicas in a Disconnected Environment

- **Steps**
 - ① **Export replica changes to a transport file**
 - ② **Transfer the transport file to the target**
 - **Manual example: burn file to CD and send through US Mail**
 - **Automated example: use a data distributed service to FTP files on a regular basis**
 - ③ **Import the transport file on the target to complete replica synchronization**

See [**How to synchronize a replica in a disconnected environment**](#)

Synchronizing Replicas (Exporting Changes)

Replica changes can be exported to xml, fileGDB, or personal GDB

```
public void ExportReplicaChanges(IGeoDataServer iGDS, String rName)
{
    IGDSExportOptions iExOptions = new GDSEExportOptions();
    iExOptions.ExportFormat      = esriGDSEExportFormat.esriGDSEExportFormatXml
    iExOptions.Compressed       = true;
    iExOptions.BinaryGeometry   = true;
    esriGDSTransportType tType = esriGDSTransportType.esriGDSTransportTypeUrl;
    esriExportGenerationsOption gOption =
        esriExportGenerationsOption.esriExportGenerationAll;

    IGDSData iGDSDData =
        iGDS.ExportReplicaChanges(rName, iExOptions, tType, gOption,
        false); /* swtich role to be a receiver */

    System.Net.WebClient wc = new System.Net.WebClient();
    wc.DownloadFile(iGDSDData.URL, @"C:\delta.zip");
    wc.Dispose();
}
```

Synchronizing Replicas (Importing Changes)

Replica changes can be imported from xml, fileGDB, or personal GDB

```
public void ImportReplicaChanges(IGeoDataServer iGDS, String rName)
{
    String fileName = new String(@"C:\delta.zip");
    FileInfo fileInfo = new FileInfo(fileName); int len = fileInfo.Length;
    byte [] bytes = new byte[len];

    FileStream fs = File.Open(fileName, FileMode.Open, FileAccess.Read);
    BinaryReader r = new BinaryReader(fs);
    r.Read(bytes, 0, len);
    r.Close(); fs.Close();

    IGDSData iGDSData = new GDSDDataClass();
    iGDSData.TransportType = esriGDSTransportTypeEmbedded;
    iGDSData.set_EmbeddedData(ref bytes);
    esriReplicaInputSource rSrc = esriGDSReplicaImportSourceDeltaXmlFile;
    esriReplicaReconcilePolicyType recPolicy = esriReplicaDetectConflicts;
    bool conflicts = iGDS.ImportReplicaChanges(rSrc, recPolicy,
        true, /* column conflicts detection */
        iGDSData);
}
```

Synchronize Replicas – Replica Sync Events

- The replication sync process can be extended by implementing a workspace extension that supports `IWorkspaceSyncReplicaEvents`

```
Interface IWorkspaceReplicaSyncEvents : IUnknown
{
    ...
    HRESULT BeforeSynchronizingDataChanges(IReplica* iTargetReplica,
                                           IUnknown* iDeltaDataChanges)

    HRESULT AfterSynchronizingDataChanges(IReplica* iTargetReplica,
                                          IUnknown* iDeltaDataChanges,
                                          ITable* iOIDMappingTable,
                                          ITable* iChangesTable);
};
```

Fine Grained API: Inspecting Replica Changes

Developers may need to get the replica changes without exporting them

```
public ArrayList ReadReplicaChanges(IReplica iReplica, IWorkspaceName iName)
{
    IReplicaDataChangeInit2 iInit = new ReplicationDataChangesClass();
    esriExportGenerationsOption gOption =
        esriExportGenerationsOption.esriExportGenerationsNew;

    try {
        iInit.Init2(iReplica, iName, gOption);
    }
    catch (COMException e) {...}

    IDataChanges iRDC = iInit as IDataChanges;
    IEnumModifiedClassInfo iMCInfos = iRDC.GetModifiedClasses();
    ArrayList al = new ArrayList();
    IModifiedClassInfo iMCI = null ;
    while ((iMCI = iMCInfos.Next()) != null)
        al.Add(mcInfo.Name);
    return al;
}
```

Synchronization Algorithm Internals

- Inserts maybe applied as updates if already exist
- Updates maybe applied as inserts if not already exist
- Synchronization preserves only GlobalIDs (not OIDs)
- OIDs Mapping table is created for new rows
- To preserve relationships where the origin key is an ObjectID, foreign keys are swizzled for newly allocated ObjectIDs

Synchronization Algorithm Internals

- **Except for creating topology dirty areas and rebuilding GN connectivity, no behaviors are executed when synchronizing changes**
- **Validating topology might be needed after replica synchronization**
- **Sync replica rebuilds geometric network connectivity only for the changes and the area affected by the changes**
- **Related objects are synchronized based on the relationship directions specified at the replica creation**
- **Conflict version is created when Rec/Post the changes with the replica version fails**
- **Overlapping replica generations is accepted (Gen1, Gen1->3)**

Synchronize Replica - Best Practice

- **Use acknowledgment after disconnected synchronization if possible**
- **Export only new changes if you know that the prior sent changes were received by the target replica**
- **Synchronize changes as often as possible**
- **Use delta FileGDB as an updategram unless you need to work with XML**
- **Use `IWorkspace3::get_RecommendedSyncOrder` to find out which replicas need to be synchronized before compressing the geodatabase**

Replica Sync Order

Developers may need to sync replicas before compressing the geodatabase

```
public ArrayList GetRecommendedReplicasSyncOrder(IWorkspace iWS)
{
    IVersionedWorkspace3 iVWS = iWS as IVersionedWorkspace3;
    IEnumBSTR iRepNames = null;
    try {
        iRepNames = iVWS.RecommendedSyncOrder;
    } catch {
        return null;
    }
    IWorkspaceReplicas iWSReplicas = iWS as IWorkspaceReplicas;
    ArrayList al = new ArrayList();
    string rName;
    while ((rName = iRepNames.Next()) != null) {
        IReplica iReplica = iWSReplicas.get_ReplicaByName(rName);
        al.Add(iReplica);
    }
    return al;
}
```

Fine Grained API: Version Differences

Developers may need to import version differences to the replica version

```
public void ExportVersionChanges(IWorkspaceName iSRC, IWorkspaceName iTrg)
{
    IVersionDataChangesInit iVInit = new VersionDataChangesClass();
    try {
        iVInit.Init(iSRC, iTrg);
    }
    catch (COMException e) {...}
    IExportDataChanges iEDC = new DataChangesExporter();
    esriExportDataChangesOption exOption =
        esriExportDataChangesOption.esriExportToXML;
    try {
        iEDC.ExportChanges(@"C:\VersionDiff.xml",
            esriExportDataChangesOption.esriExportToXML,
            iVInit as IDataChanges,
            false); /* overwrite if exists */
    }
    catch (COMException ce)
    { ... }
}
```

Fine Grained API: Importing Versions Differences

Developers may need to import version differences to the replica version

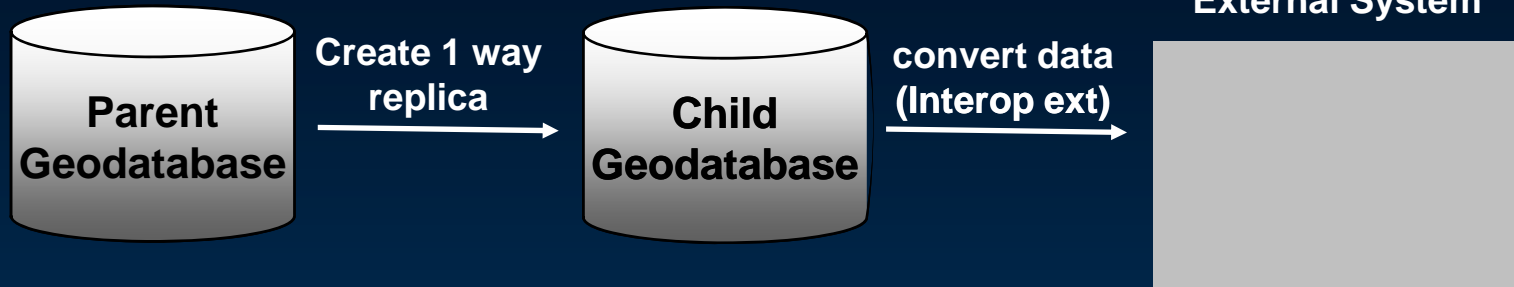
```
public void ImportVersionChanges(IWorkspaceName iReplicaWorkspace)
{
    IDeltaDataChangesInit iInit = new DeltaDataChangesClass();
    try {
        iInit.Init(@"C:\VersionDiff.xml", esriExportToXML);
    }
    catch (COMException e) {...}
    IImportDataChanges2 iImportDC = new DataChangesImporterClass();
    try {
        bool conflicts = iImportDC.ImportDataChanges2(
            iRepWorkspace,
            iInit as IDeltaDataChanges,
            true, /* reconcile with the parent version */
            esriReplicaReconcilePolicyType.esriReplicaDetectConflicts,
            true, /* column level conflicts detection */
            true); /* create oid mapping table */
    }
    catch (COMException ce) {...}
}
```

Replication Interoperability with non-geodatabase formats

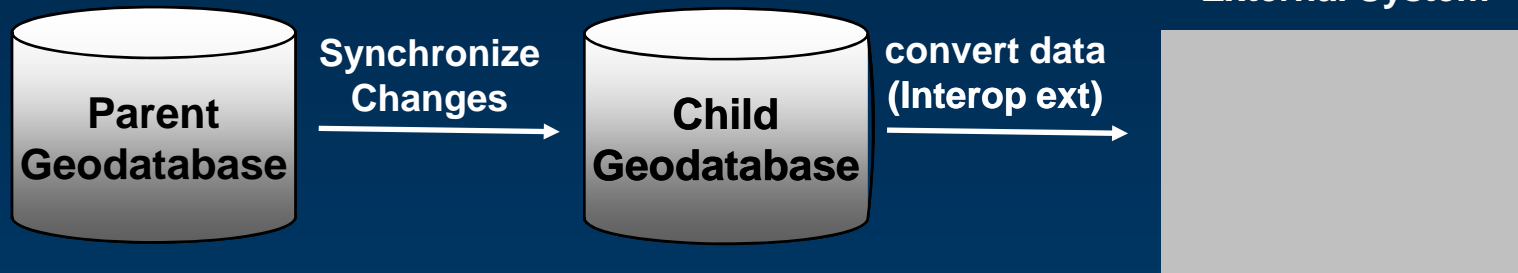
- Replication where the parent is a geodatabase and the child is a non-geodatabase format
- Use the data interoperability extension to build 1 way replication and check-out replication workflows
- To check in it is required to generate an ESRI XML data change file programmatically
 - Use ESRI's published [XML schema of the geodatabase](#)

1 way replication workflow

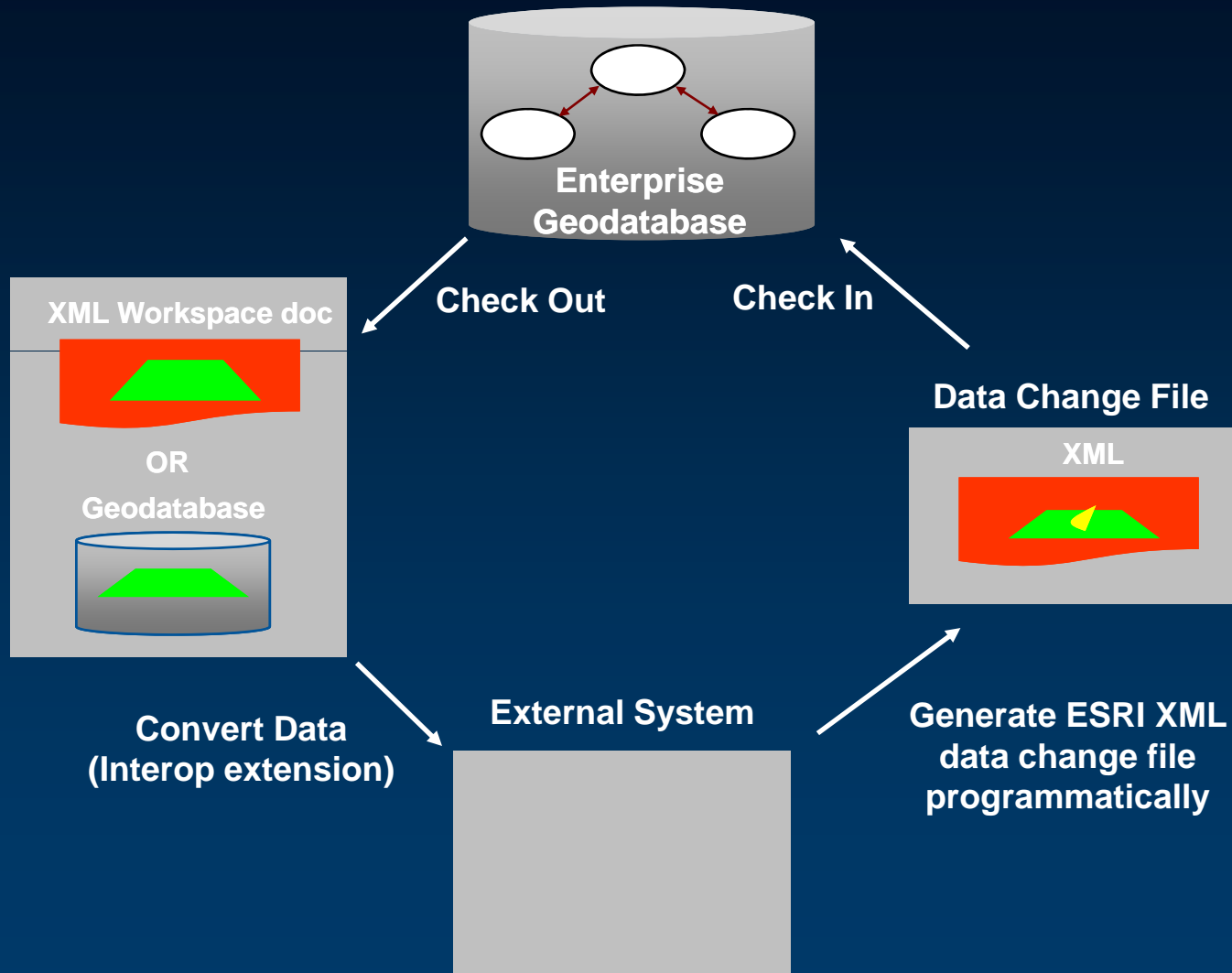
- **Create Replica**



- **Synchronize Replica**



Check-out replication workflow



Replication Interoperability with External Systems

- **Simple versus full geodatabase replication model**
- **Check in supports both simple and full geodatabase model**
- **Check in from external system requires a knowledge of the checkout guid**
- **Check in synchronization order**

ESRI XML Data Change File

```
<esri:UpdateGram xmlns:esri=http://www.esri.com/schemas/ArcGIS/9.2... >
  <GeodatabaseRelease>...</GeodatabaseRelease>
  <GUID>D6A7E5C3-9EF0-45DE-B412-55502DAEEEC4</GUID>
  <ParentID>70</ParentID>
  <ParentConnectionInfo>...</ParentConnectionInfo>
  <ModelType>esriModelTypeFullGeodatabase</ModelType>
  <UpdateGramDefinition>
    <ChangedDatasetDefinition>
      <DatasetName>states</DatasetName>
      <DatasetType>esriDTFeatureClass</DatasetType>
      <ParentDB />
      <ParentOwner>BOB</ParentOwner>
    </ChangedDatasetDefinition>
  </UpdateGramDefinition>
  <UpdateGramTopologyDefinition />
  <UpdateGramData>...</UpdateGramData>
</esri:Updategram>
```

Questions