



Distributed Geodatabase for Developers (Best Practices)

*Khaled Hassen
Gary MacDougall*



Schedule

- 75 minute session
 - 60 – 65 minute lecture
 - 10 – 15 minutes Q & A following the lecture
- Cell phones and pagers

Please!
Turn OFF cell phones
and paging devices



- Please complete the **session survey** – we take your feedback very seriously!

Prerequisites

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

Overview

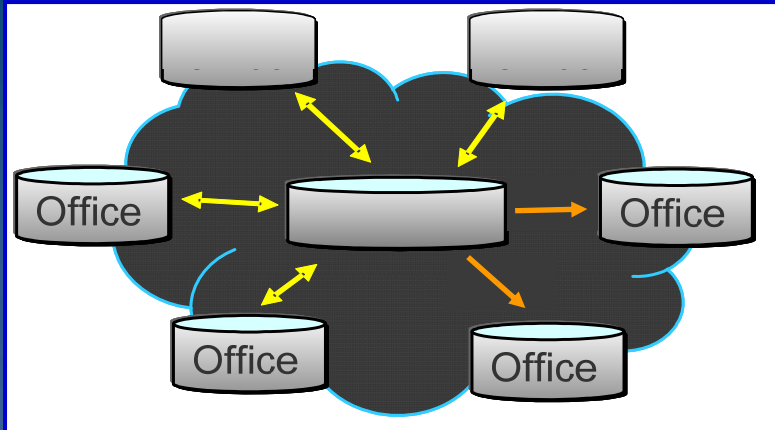
- **Distributed geodatabase use cases and techniques**
- **Geodatabase replication**
 - Replication introduction
 - Replication internals and developer API
 - DEMO: Replica creation with custom workspace extension

Distributed Geodatabase Use Cases

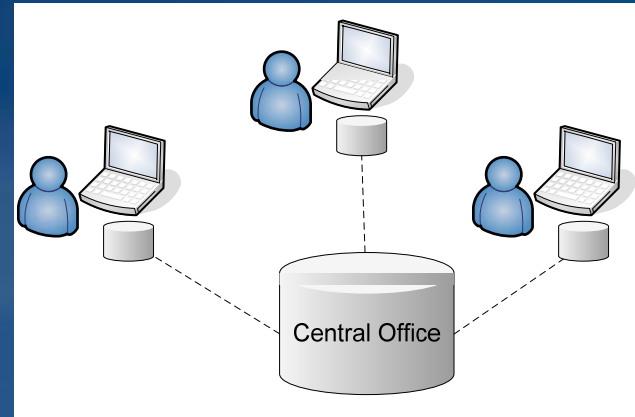
- **Mobile Users and Field Crews who need to be disconnected from the network**
- **Users who need to maintain copies of data at different organizational levels (city, county, state)**
- **Users who want to maintain copies of data at different geographic facilities**
- **Users who need to distribute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

Distributed Geodatabase Use Cases

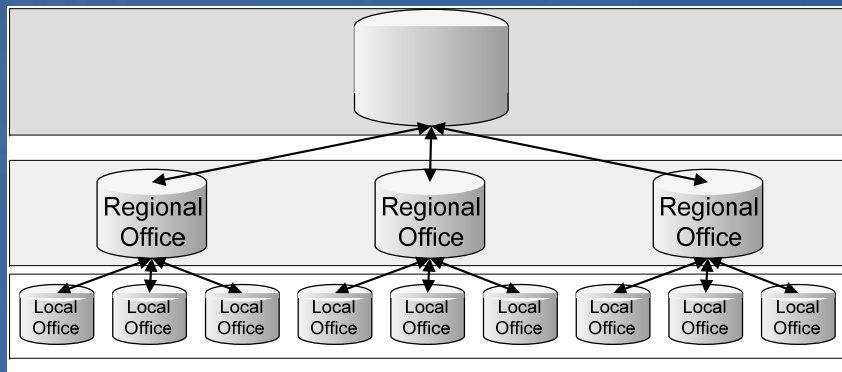
Regional Offices



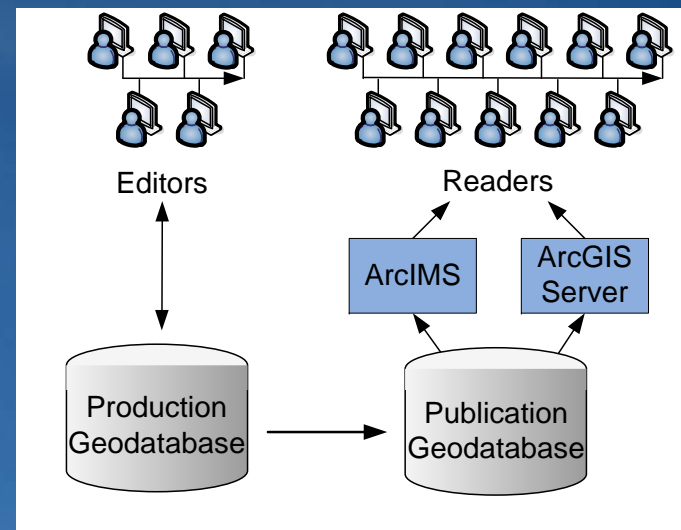
Mobile Users



Multiple levels



Production / Publication



Distributing 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
 - No safeguard against data loss
 - No safeguards against data redundancy

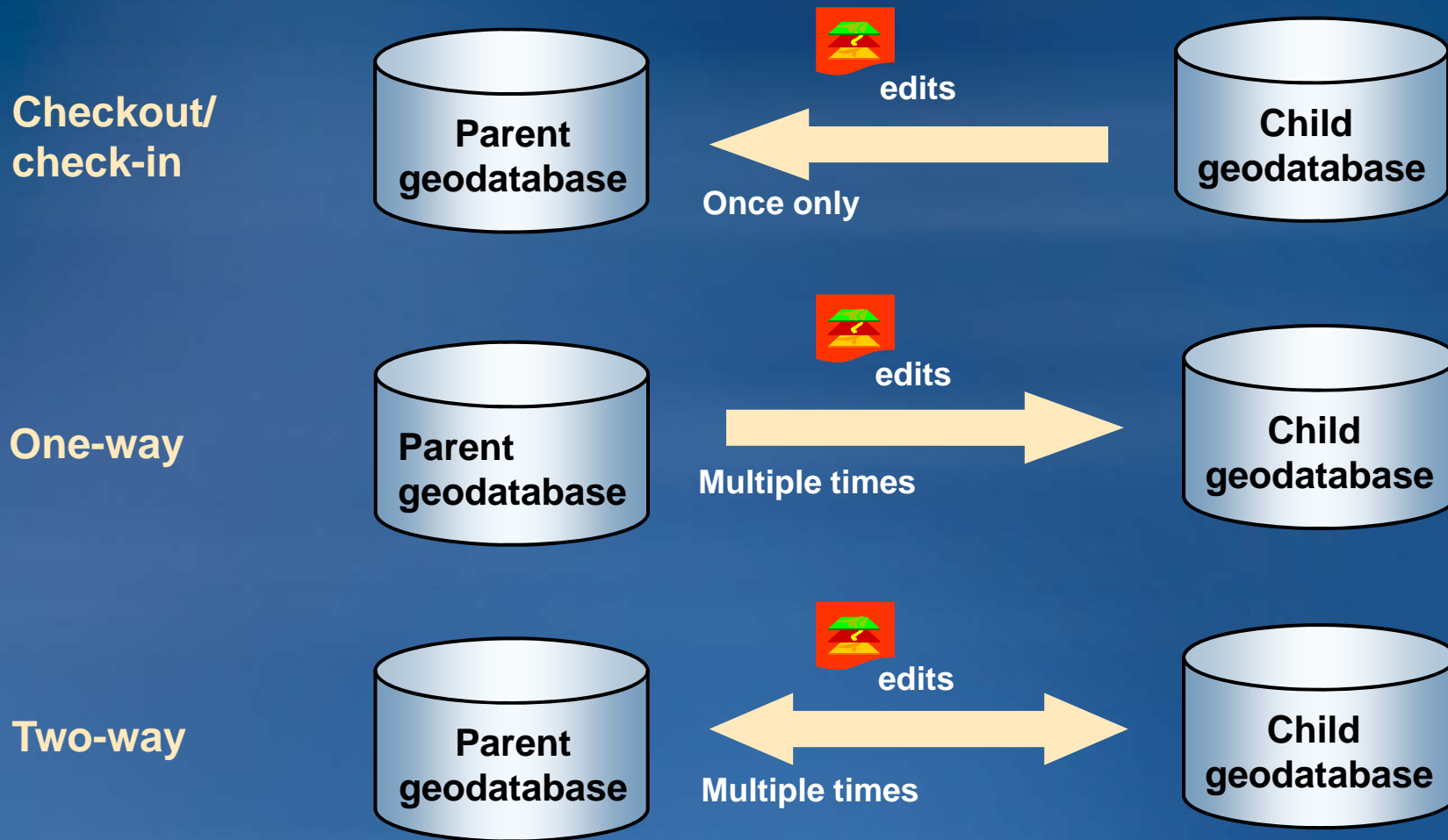
Distributing Data: DBMS Tools

- **Use DBMS replication tools to distribute data**
- **Requirements**
 - 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

Distributing Data: Geodatabase Replication

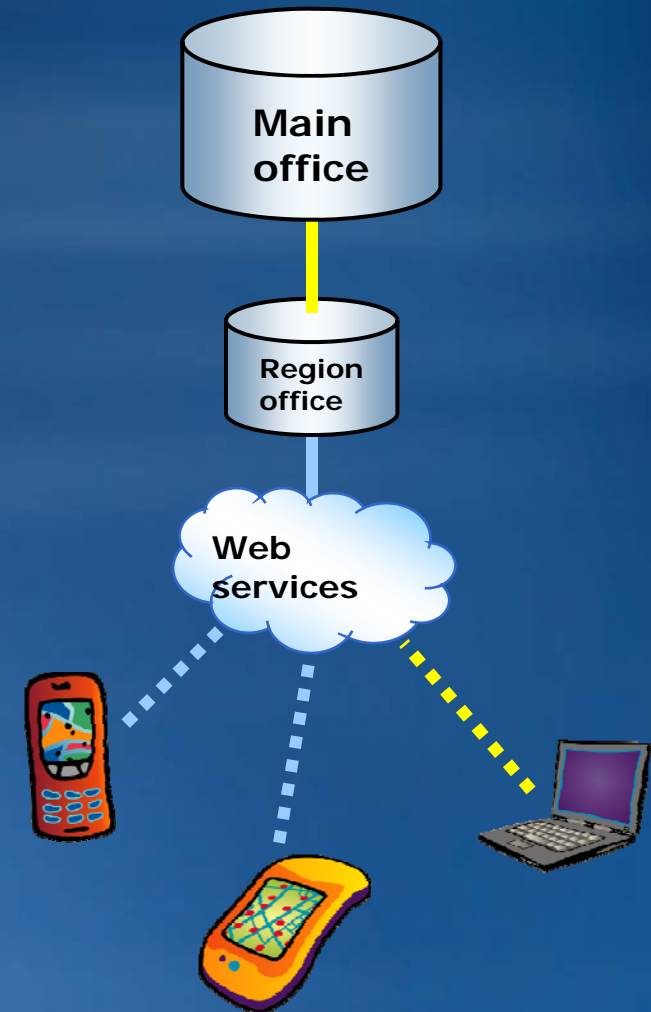
- **Allows you to distribute copies of data across 2 or more geodatabases**
- **You can edit the databases independently and synchronize them as needed**
- **Uses geodatabase connections directly (LAN) or using geodata services publish on ArcGIS server (WAN)**
- **Works in a connected and a disconnected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

Geodata Services: replica types



Data distribution in Enterprise systems

- Geodata services can be used in conjunction with other data distribution techniques
- Scenario
 - Use geodatabase replication to synchronize changes between offices
 - Use Mobile services for field workers with lightweight mobile devices
 - Use geodata services for field workers who need ArcGIS Desktop or ArcGIS Engine in the field



Geodatabase Replication – Best Practices

- 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 ArcSDE for SQL Server Express instead of check-out replicas
 - Use 1 way replicas over 2 way replicas when possible

Geodatabase Replication – Best Practices

- **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 – Best Practices

- Integrate synchronization with version management strategy
 - See [Geodatabase Replication and Compress](#)
- Network speed
 - Use geodatabases directly over fast networks (LAN)
 - Use ArcGIS server and geodata services on slower internet connections
 - Use disconnected synchronization techniques over very slow networks (slow dial-up modem) or where there is no network connectivity
- Upgrade to the latest release or service pack
 - 9.3.1 includes performance improvements and a number of general bug fixes

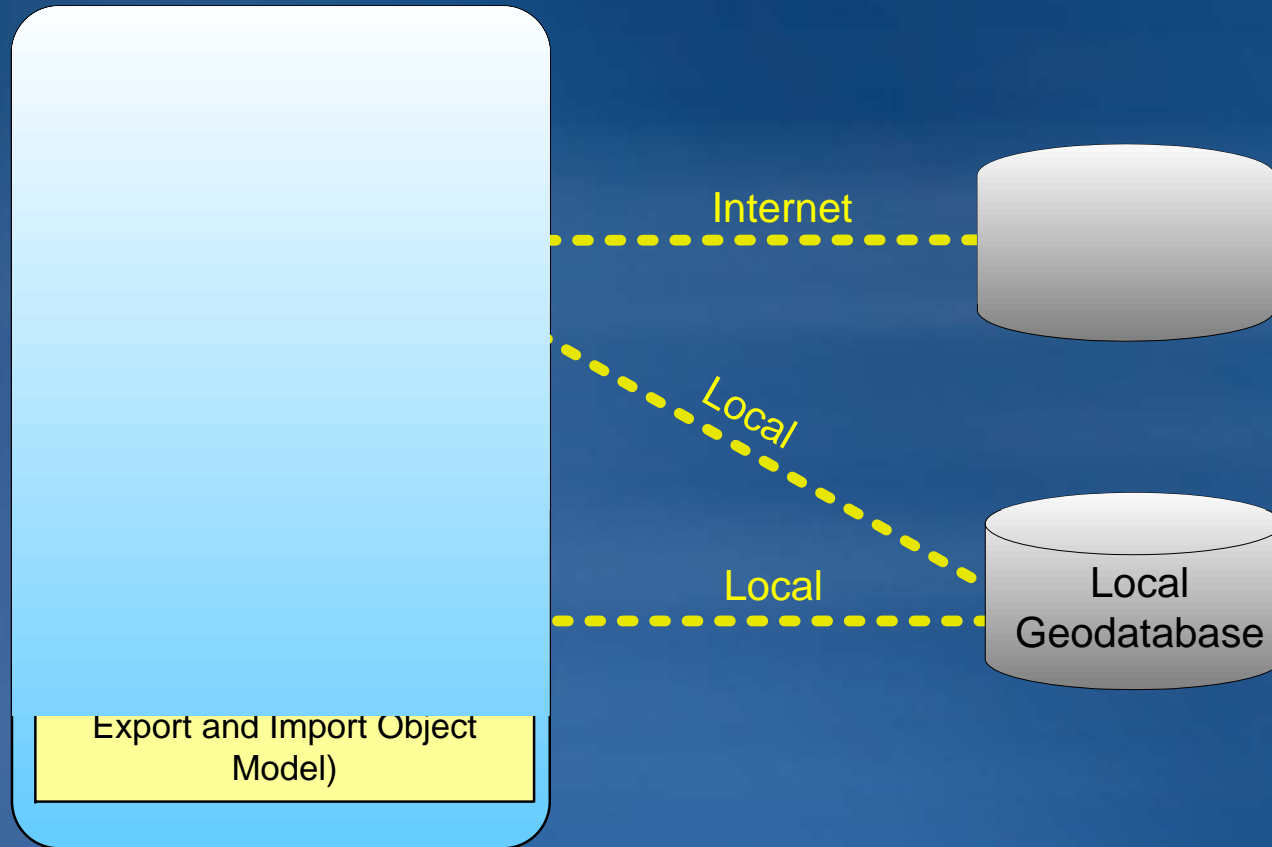
When to write code

- **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

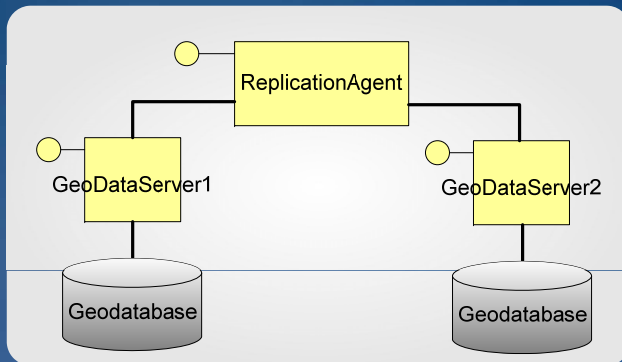
- **Disconnected**

- 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

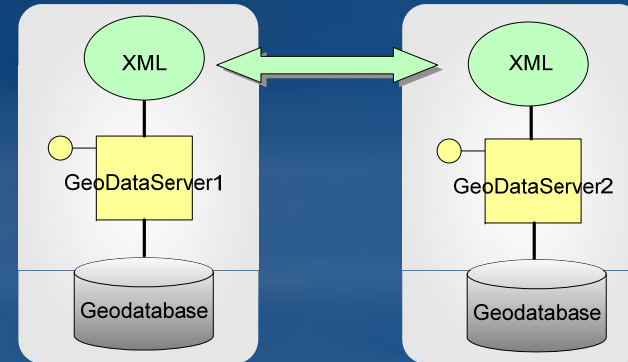
Connected

LAN

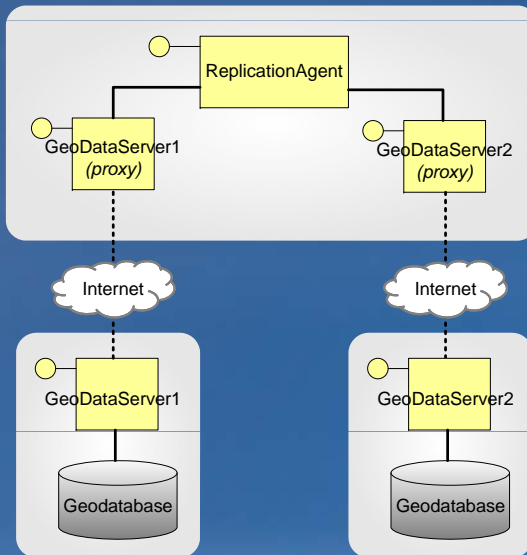


Disconnected

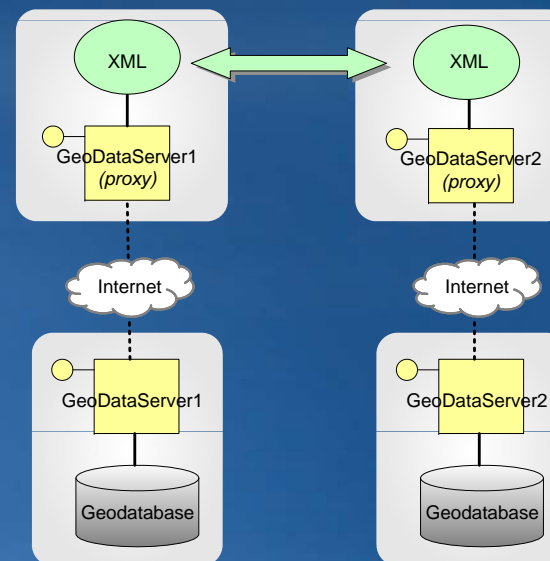
LAN



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;
    IGDSEExportOptions iGDSEOptions = new GDSEExportOptions();
    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**

WorkspaceExtension With ReplicaEvents

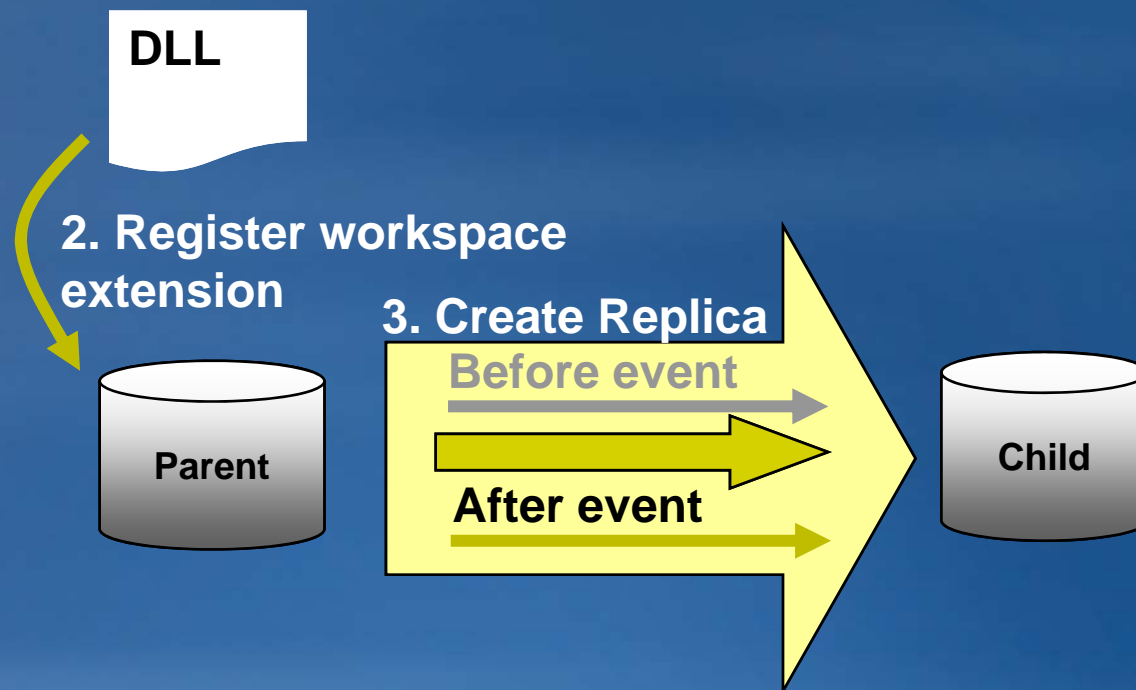
- WSExtension needs to support IWorkspaceExtensionControl, IWorkspaceExtension and IWorkspaceReplicaEvents interfaces
- WSExtension needs to be registered as COM object (regasm.exe, regsvr32.exe)
- WSExtension needs to be registered with the workspace

```
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) {...}
    }
}
```

Replica Creation Demo

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

1. Build workspace extension



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)
    {...}
}
```

Adding a Dataset to a Replica

- Simple feature classes and tables can be added to a replica once it has been created
- Data must meet replication requirements
- Feature class or table must be added to both the parent and the child replicas
- Use the RegisterReplicaDataset method from the IWorkspaceReplicasAdmin2 interface
 - This includes the filter

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

    System.Net.WebClient wc = new System.Net.WebClient();
    wc.DownloadFile(iGDSData.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);
}
```


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 may be applied as updates if already exist
- Updates may be 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",
            exOption,
            iVInit as IDataChanges,
            false); /* overwrite if exists */
    }
    catch (COMException ce)
    { ... }
}
```

Fine Grained API: Tables Differences (9.3)

```
public void ExportTableChanges(ITable iInserts, ITable iUpdates, IStringArray
iDeletes)
{
    ITablesDataChanges iTDCs = new TablesDataChangesClass();
    try {
        ITableDataChange iTDC = new TableDataChange();
        iTDC.Init("Parcel", iInserts, iUpdates, null);
        iTDC.SetDeletedIDs(null, iDeletes);
        iTDCs.Add(iTDC);
    }
    catch (COMException e) {...}
    IExportDataChanges iEDC = new DataChangesExporter();
    try {
        iEDC.ExportChanges(@"C:\TableDiffs.xml",
                           esriExportDataChangesOption.esriExportToXML,
                           iTDCs as IDataChanges,
                           false); /* overwrite if exists */
    }
    catch (COMException ce)
    { ... }
}
```

Fine Grained API: Importing Differences

Developers may need to import version/Tables differences to the replica

```
public void ImportChanges(IWorkspaceName iReplicaWorkspace)
{
    IDeltaDataChangesInit iInit = new DeltaDataChangesClass();
    try {
        iInit.Init(@"C:\diffs.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) {...}
}
```


Enhancements – User Defined GlobalIDs (9.3)

- Allows users who already manage their own unique identifiers to start using geodatabase globalid columns
- New interface (IClassSchemaEX) allows you to control the globalid values
 - Add user defined GUID values to a column of type GUID
 - Use IClassSchemaEX to convert the GUID column a to GlobalID column
 - IClassSchemaEX can also be used to change the type from GlobalID to GUID such that more changes can be made
- Not supported in a versioned environment

What's coming: 9.4 Geodatabase Replication

- **General enhancements**
 - One way replication using archiving
 - One way child to parent replication
 - Schema mapping across replicas
 - Check-out and 1 way child to parent replicas with non-versioned ArcSDE data with conflict detection
 - Create a replica to a named version on the child
 - Simple check-out and two-way replicas

What's coming: 9.4 Geodatabase Replication

- **Developer enhancements**
 - **DataChangesImporter works with non-versioned data and personal and file geodatabases**
 - **Changes can be imported directly from TablesDataChanges without having to export to a delta file**
 - **Extended update-gram format to store the original/old row**

Want to Learn More?

ESRI Training and Education Resources

- Instructor-Led Training
 - Managing Editing Workflows in a Multiuser Geodatabase
- Free Web Training Seminar
 - Introduction to Geodatabase Replication

<http://www.esri.com/training>