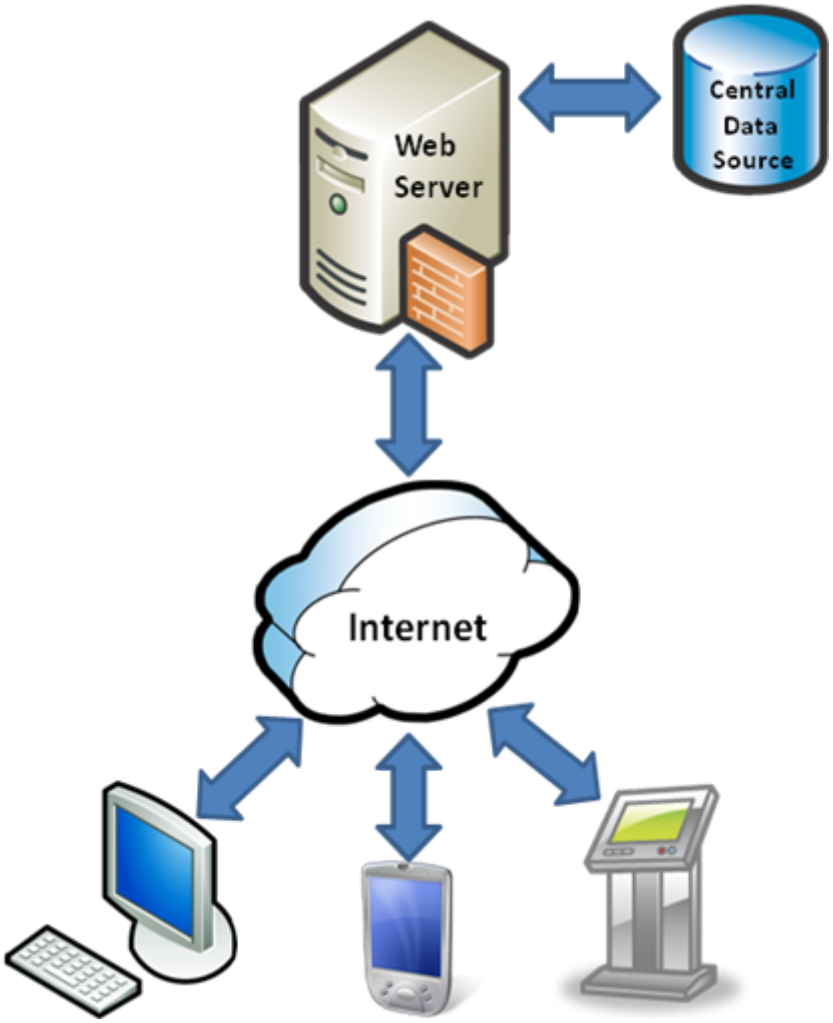
Introduction to Sync Framework Database Synchronization

Introduction

The ability to support mobile and remote workers is becoming more and more important for organizations every day. It is critical that organizations ensure users have access to the same information they have when they are in the office. In most cases, these workers will have some sort of laptop, office desktop, Smartphone, or PDA. From these devices, users may be able to access their data directly through VPN connections, Web servers, or some other connectivity method into the corporate networks as seen below.



This type of solution is fairly simple to implement. Unfortunately, for most remote workers it is less than satisfactory. Some major disadvantages of this type of solution include:

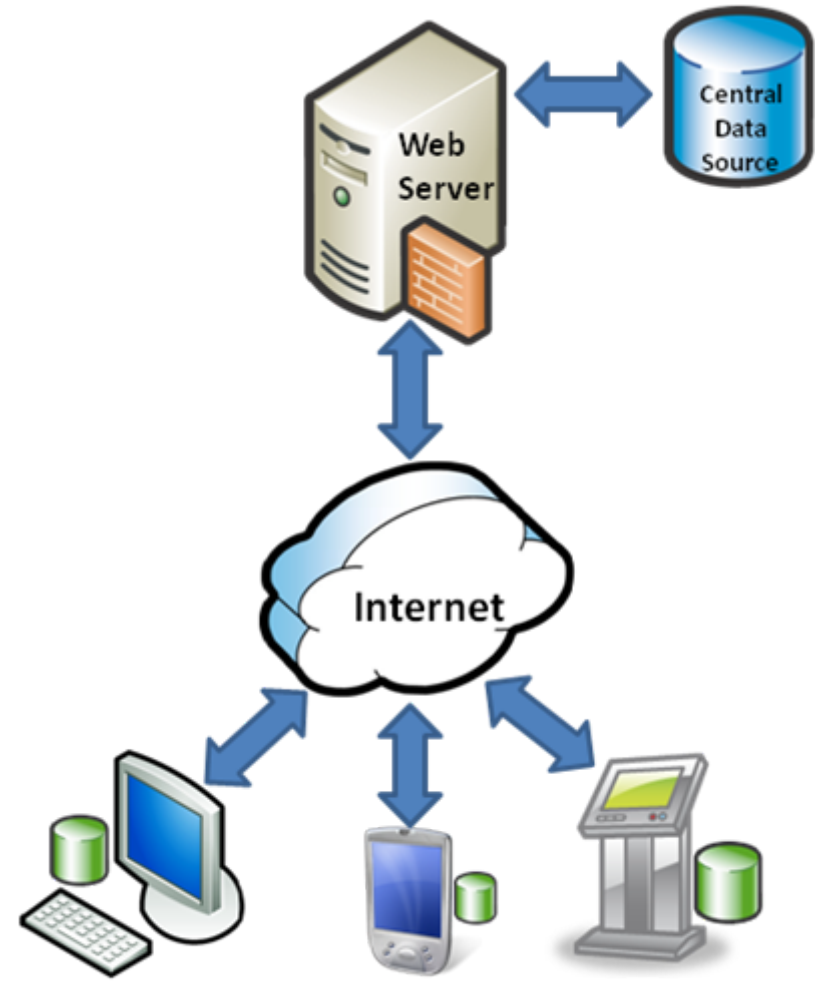
1. **Network Requirements**: In order to allow users to access their information, the remote device needs to have a constant connection to the corporate network while accessing their data. For some workers, such as those who are working from home, this may not be a problem. For others, such as sales reps who are constantly on the move, this may be more difficult. For example, if that sales rep were visiting a customer and was unable to access inventory data because of a lack of network connectivity, it would be very difficult for this user to effectively do their job.
2. **Data Access Speeds**: In a typical client/server corporate environment, users have high speed networks that allow them quick access to information. Remote workers, however, are typically connected over slow, unreliable wired or wireless

networks. With this solution, every piece of data this user needed would need to be downloaded every time it is requested because there is no way to persist the data on the device. For example, if a sales rep is required to download his product list every time he opens his application, he will quickly become frustrated with the time lag required to populate his application with information.

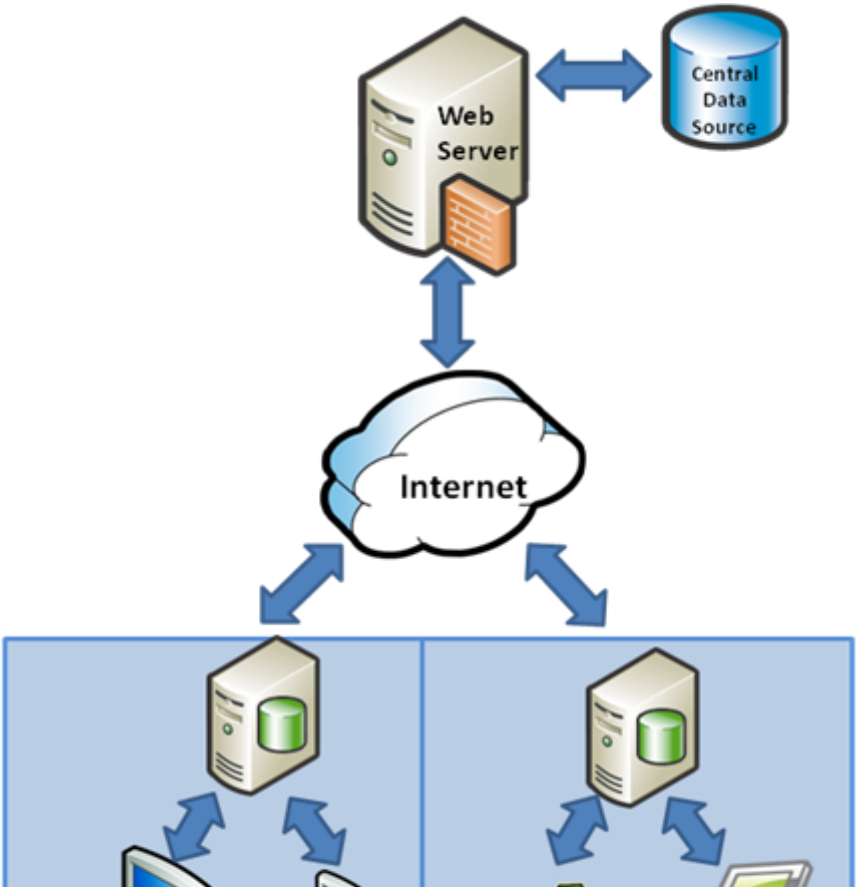
3. **Single Point of Failure**: With this type of solution, all users are reliant on a single server. If that database becomes unavailable due to planned server downtime or from server failures, all of the remote workers will be disconnected from their data.
4. **Server Scalability**: As more workers work remotely, the performance of the corporate servers will be affected, leading to a need to add additional hardware.

One alternative to this solution is to implement an Occasionally Connected Application (OCA). An OCA allows a remote worker to continue to access their data, but unlike the previous scenario where the user accessed the corporate database directly, the information the worker requires is stored locally on the user's device. In order to populate this user's local database, an OCA will typically include some data synchronization capabilities. Data synchronization consists of the ability to periodically take information that is stored in the client database (such as SQL Server Compact) and synchronize changes with a server database (such as SQL Server). The advantage of a synchronization-based solution is that users are no longer required to have a constant network connection to access their information. Since their data is stored locally they are given constant access to their data while offloading processing requirements from the central database. Furthermore, the user is no longer limited by the network speed and can now access the data at the speed of the device.

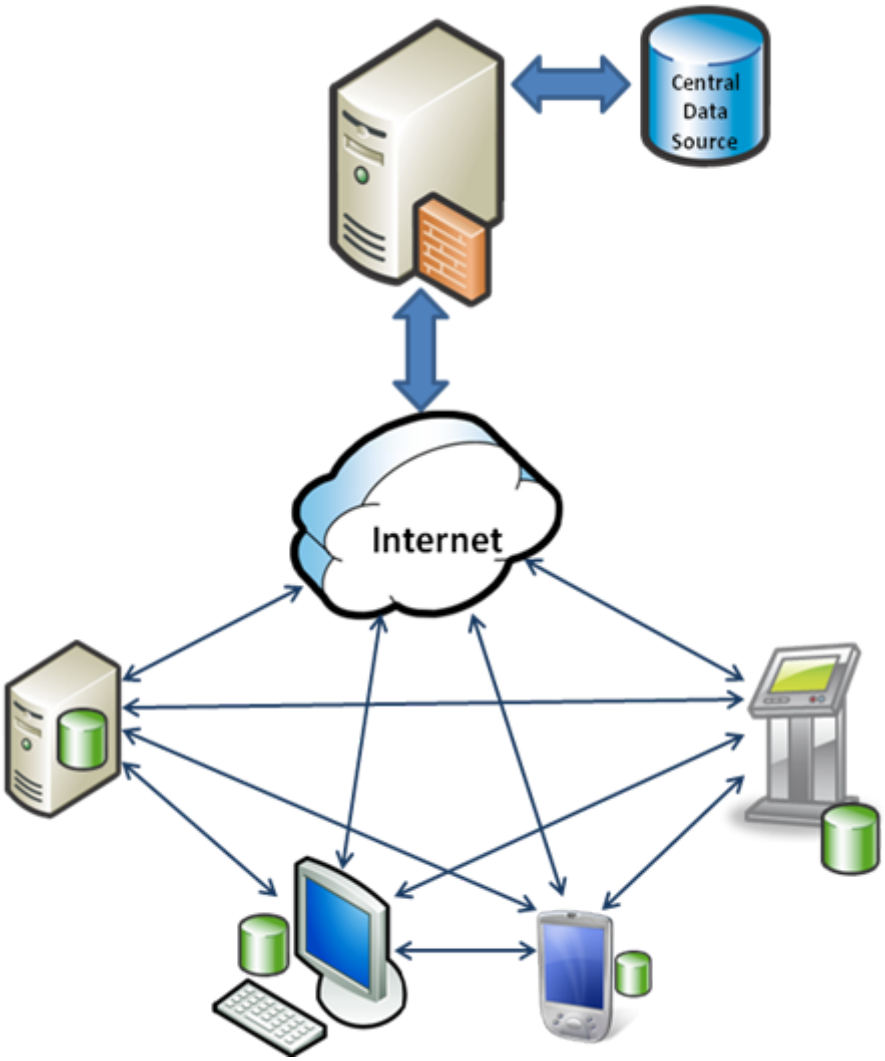
In the following two diagrams we can see examples of OCAs where data (represented by a green database) is persisted locally on the remote worker's device. The first example is of a standalone database system where information is stored directly on the user's device. The second example is of a remote office where information is stored in a workgroup database within this remote office so that multiple local workers can access the data.



Standalone Database Applications



A common extension to this type of OCA is the ability to support data collaboration between databases. As seen below, a remote database is free to exchange information with any other database. This type of solution is useful when a team of people are working in remote locations and do not have access to a central database. These workers often need to share information amongst each other but since they do not have connectivity to the central database they need to share information through some sort of peer-to-peer network.



Throughout the rest of this document we will discuss OCAs, with a special emphasis on Sync Framework database synchronization providers, a key technology that enables developers to build OCAs.

Database Synchronization and the Microsoft Sync Framework

Database synchronization providers are a part of the Microsoft® Sync Framework. Sync Framework is a comprehensive synchronization platform that enables developers to add synchronization capabilities to applications, services, and devices. Sync Framework solves the problem of how to synchronize any type of data in any store using any protocol over any topology. Fundamental to Sync Framework is the ability to support offline and collaboration of data between any types of endpoints (such as device to desktop, device to server, etc.).

Sync Framework database synchronization providers enable synchronization between ADO.NET-enabled databases. Since the database synchronization providers are part of the Sync Framework, any database that uses these providers can then also

exchange information with other data sources that are supported by Sync Framework, such as web services, file systems, or custom data stores.

The primary focus of this document will be on synchronizing information between database systems and how Sync Framework helps developers avoid many of the common issues associated with OCAs.

Challenges of Building an OCA

With an OCA, users have quick access to their data and do not require a network connection to the central database in order to access their information. With an OCA, updates are made locally and then synchronized into the central database, rather than being made directly at the central database. Although OCAs solve the key problems associated with directly accessing a central server, there are a number of challenges associated with building OCAs. The following sections will discuss these challenges and propose ways that Sync Framework can be used to avoid these challenges.

Change Tracking

In order to make data synchronization efficient, some method of change tracking is required. Change tracking is the ability to provide a list of changes (insert, updates, and deletes) that were made to the database from one point in time to another. Imagine a remote user who connects to the central database and wishes to bring their data up to date since the last time they were online. Without change tracking, this user would need to take all of the data from the central data source and then merge that data with the changes that the user made to the local database on the user's computer or device. In a mobile environment, this is extremely inefficient because of the amount of time a wireless network would take to exchange this information. It would be especially slow if large datasets were exchanged. Furthermore, if the connection drops the data must be re-downloaded, making this even less efficient.

To avoid these issues, a change-tracking system is typically used. One popular method for change tracking is through the use of rowversions and triggers. This method requires a rowversion column to be added to each table. For deletions, a separate table or a "deleted" flag column is typically required to log these removed rows that are maintained through triggers.

The major disadvantages to this solution are:

- Changes are required in the central database schema to add columns and tables that may affect current applications.
- Triggers are fired for each change made to a row, which has performance implications.
- Logic for maintaining proper rowversions and row deletions can get extremely complicated.
- Long running transactions can result in some data being missed during synchronization, resulting in data inconsistencies.

SQL Server 2008 has introduced a new alternative method for tracking changes called SQL Server 2008 Change Tracking. The concept behind change tracking is that an administrator marks certain tables to be monitored for changes. From that point SQL Server 2008 keeps tracks of any inserts, updates, or deletes that are made. When a remote "requestor" requests changes, SQL Server 2008 will provide all of the changes that have occurred since the last successful download as specified by the requestor. The Sync Framework database synchronization providers have been built to take advantage of SQL Server 2008 change tracking and provide the following advantages for an OCA environment:

- No schema changes are required to be able to track changes.
- Triggers are not required for tracking changes, which means that tracking changes has far less of an impact on the server. In certain cases, the DML overhead associated with trigger based change tracking can be 400% greater than that of SQL Server 2008 change tracking. The overhead of enabling SQL Server 2008 change tracking is similar to the overhead of maintaining a second index.
- All of the logic for tracking changes is internal to the SQL Server engine and as such reduces the complexity for setting up this type of system.
- Data consistency issues associated with long running transactions are no longer an issue.
- Includes integrated database administration feature such as Dynamic Management Views and Security.

Maintaining Change Data

Change tracking tables will typically grow quite quickly, taking up storage space and affecting the performance of queries being executed against them. As such, the next logical step is to determine when tracked changes can be removed. Basing cleanup on when users have last synchronized is difficult because a device may have not been synchronized or the user may have simply

stopped using it. These factors make it difficult for administrators to determine when all active users have received the changes which would allow them to clean up the change data.

In SQL Server 2008, customizable retention thresholds are used to maintain this change data and automatically clean up old data. Furthermore, this process runs as a background process to help offset any performance impact on the server.

Conflict Detection and Resolution
Conflicts are another issue that can arise in an OCA. Conflicts will occur when two or more databases make a change to the same piece of data and then the synchronization engine tries to apply those into a single database. For example, imagine two sales reps who try to update a customer's address to two different values. The first sales rep successfully synchronizes the update to the central database. When the next sales rep goes to synchronize the update to the main database, a conflict occurs because the current state of the row is different from what the synchronization engine was expecting. There are a variety of ways to resolve these conflicts. For example, the last change to come in may be the one that wins, or alternatively it may be based on which user has the most seniority.

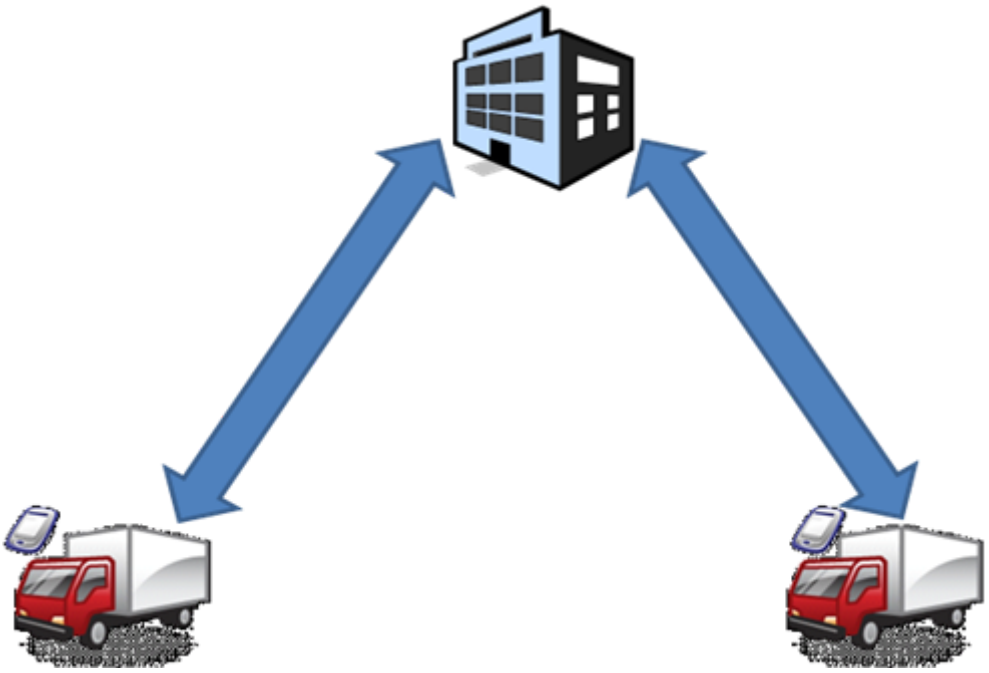
Sync Framework provides conflict detection and resolution capabilities out of the box and SQL Server 2008 improves on this experience by decreasing the complexity associated with identifying conflicts. By using this technique, the first sales rep in our example will successfully upload the change to the central server. It will be applied since there is no conflict yet. When the second sales rep goes to upload the change, a conflict will be detected because the change version in the central server does not exist in the current sales rep database. From this point, logic in the remote application determines how to handle the conflict.

Prioritizing Data Exchange
When an OCA is deployed, users will often look for ways to optimize their data exchange given limited or slow network connections. One good way to optimize data synchronization is by prioritizing data to define data that is high priority or critical. Using priority data exchange, critical changes could be synchronized immediately while leaving less important data to be synchronized at a later time. For example, imagine a user currently has access only to a slow connection. Given the limited bandwidth, this user wishes to send change from a single (highly important) table and leave changes from other tables for later in the day when she can connect through a faster network connection. With Sync Framework, synchronization can be executed from the remote application on a table-by-table basis and on an upload-only or download-only basis. This means that specific tables can be synchronized, leaving others to be synchronized at a later time while still enabling guaranteed data exchange for all data.

Background Synchronization
Nothing will frustrate a user more than when they are locked out of their application while another process (such as data synchronization) takes control. With most OCAs, users are not able to access their local database while processes like data synchronization are being executed.

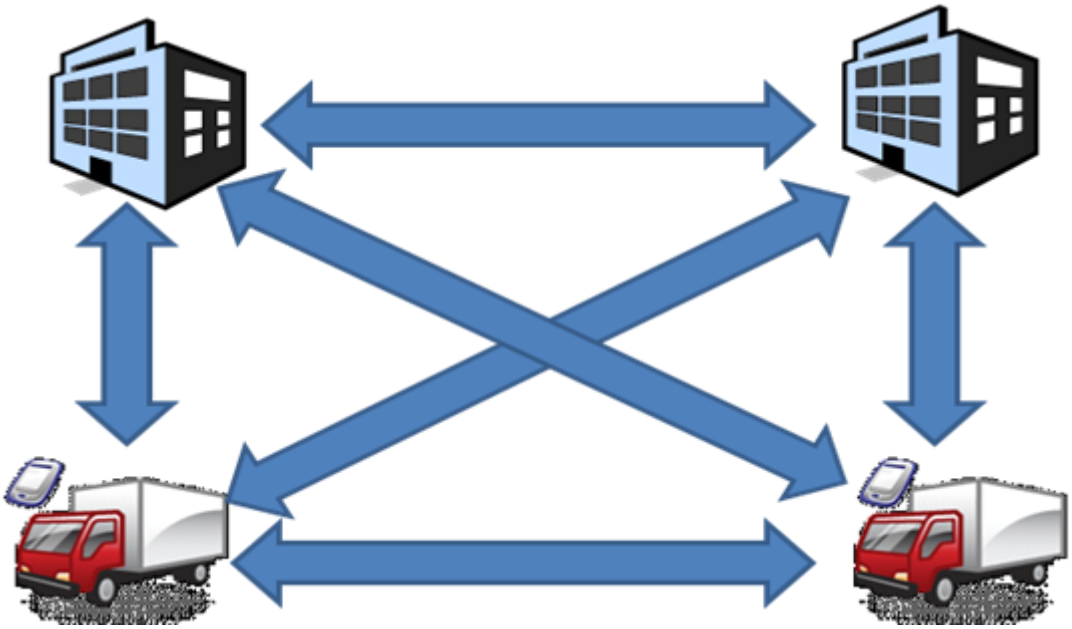
With Sync Framework, synchronization can run as a background thread. As long as the local database supports the ability to synchronize on a separate connection (as SQL Server Compact does), synchronization can be executed in the background. This allows a local user to continue to use and change their database while synchronization is being completed.

Multiple Synchronization Topologies
When first architecting an OCA, organizations will typically start with a single network topology. For example, imagine a group of delivery workers who download their route information at the start of the day from a central warehouse database and then upload their package and route completion information to the same warehouse database at the end of the day. In a traditional synchronization environment this would be considered a hub-and-spoke model as seen below.



However, what happens if a delivery worker completes his work on the other side of the city from the warehouse and the driver wishes to synchronize with a different warehouse? Or to take it one step further, why not let the delivery worker synchronize data with another delivery worker by using data collaboration. The second user could then go to a warehouse and upload both users' data. As you can imagine the logic required to orchestrate this type of synchronization can quickly become very complicated.

With Sync Framework, virtually any type of synchronization topology can be used. Now organizations are no longer limited to a single topology but can choose any one or a combination of topologies, meaning they could create combinations of offline and collaboration-based architectures.



Custom Client and Server Databases

Without a doubt, there will come a time when you need to add a new database into your synchronization environment. For example, an enterprise may have legacy systems or mainframes that need to be integrated into the existing synchronization environment. Or alternatively, a mobile device may already have a custom database that needs to be integrated into the back-end database. To make matters worse, what if you are not able to make changes to that database to allow you to track changes? With Sync Framework, providers for common databases such as SQL Server Compact and SQL Server 2008 are included out of the box. Additionally, using the Microsoft® Sync Framework, a developer can create custom providers that integrate

synchronization with virtually any place you store your data, whether that is a Web service, a USB key-chain drive, or a SIM card on a cell phone.

Security
In an OCA, there are many aspects to security that may need to be implemented. Some of these areas include:

- Database encryption
- Database authentication
- Encryption of data synchronization
- Internal authentication

Sync Framework helps to increase the security of applications that depend on synchronization. On the device side, SQL Server Compact offers the ability to both encrypt the database as well as the ability to enable user authentication. From a synchronization perspective, Sync Framework supports the ability to encrypt data as it travels between databases. On the corporate side, SQL Server 2008 as well as existing IIS security can be leveraged for user authentication as users exchange data.

Summary

Sync Framework is a comprehensive data synchronization solution that enables developers to build solutions that support synchronization of any database, on any data protocol over any network topology. With Microsoft® Sync Framework, the synchronization of information can flow virtually anywhere in or out of the organization, allowing developers to build efficient and highly scalable Occasionally Connected Applications. Using Microsoft® Sync Framework developers can extend database synchronization applications to enable collaboration of data between devices using any data source.