Data Management

Microsoft Dynamics CRM 4.0

Sharing Data across Microsoft Dynamics CRM Deployments

White Paper

Date: March 2010

Microsoft Dynamics

Acknowledgements

Initiated and sponsored by the Microsoft Dynamics CRM *Engineering for Enterprise* (MS CRM E^2) Team, this documentation was developed with support from across the organization and in direct collaboration with the following:

Key Contributors

Brian Bakke (*Microsoft PFE*)
Grant Geiszler (*Microsoft PFE*)
Roger Gilchrist *(Microsoft)*

Technical Reviewers

Shawn Dieken (*Microsoft PFE*)
Jim Steger (*Sonoma Partners*)

The MS CRM E^2 Team recognizes their efforts in helping to ensure delivery of an accurate and comprehensive technical resource in support of the broader CRM community.

MS CRM E^2 Contributors

Amir Jafri, Program Manager
Jim Toland, Content Project Manager

Feedback

Please send comments or suggestions about this document to the MS CRM E^2 Team feedback alias (entfeed@microsoft.com).

Table of Contents

Overview

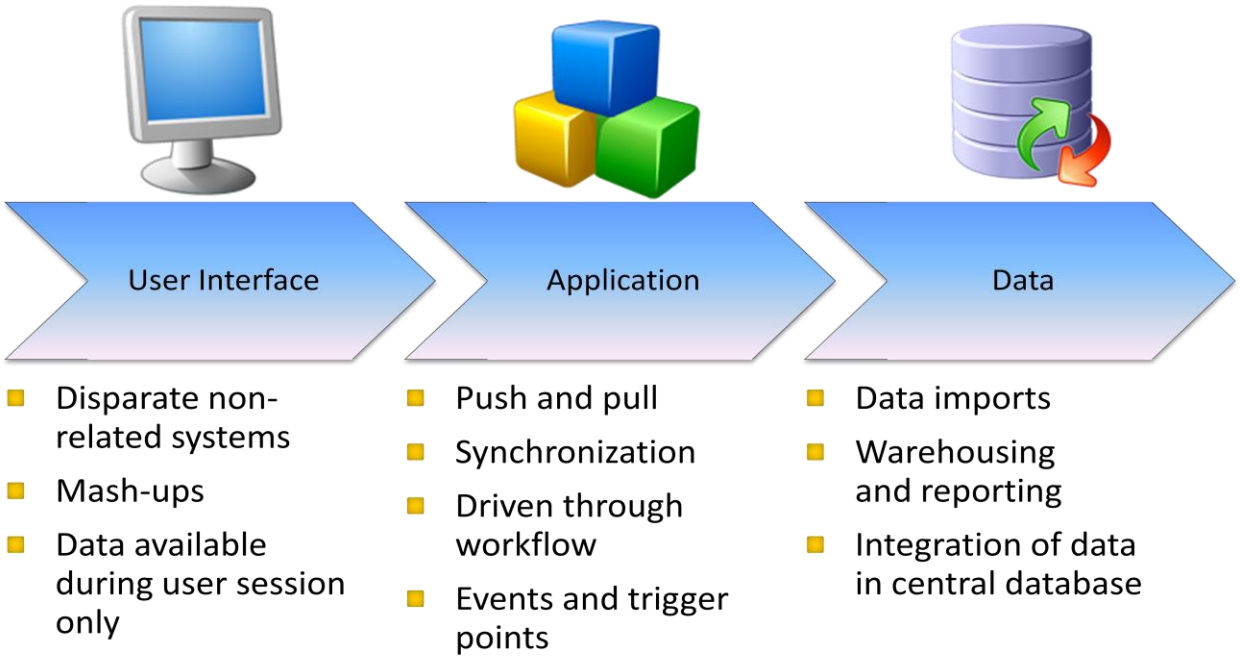
The ability to share data across Users, Teams, and Business Units within a single organization is provided by Microsoft Dynamics CRM 4.0 "out-of-the-box." Administrators can provide users with differing levels of access to the data stored in CRM based on the roles, privileges, teams and Business Unit settings associated with specific users.

While using a single CRM instance is probably the simplest way to accommodate the data sharing needs within an organization, using multiple instances of Dynamics CRM may be preferable in a number of scenarios.

Consider the following reasons for using multiple instances of Microsoft Dynamics CRM:

- *Organizational Scale*, to accommodate large volumes of users or data, or heavy processing
- *Geographical Distribution*, to allow users to work against local rather than remote instances of CRM
- *Divisional or partner segregation*, to limit the data that is distributed and the levels of access that each instance has
- *Legal requirements*, to ensure that data is retained in a specific local geography

This paper is focused on sharing data across multiple CRM organizations, which can occur at three distinct levels: Front-End / User Interface Level, Application Level, or Data Level.



The paper provides an overview of each level of integration and details about the techniques that are used for integration at the Application and Data levels, which include Web Services, Web Services-based solutions, Filtered Views, and SQL Server Replication.

Important: This paper is not intended to serve as a solution guide, but rather to assist Microsoft Dynamics CRM architects with designing and implementing solutions to meet the needs of specific business scenarios.

Customer Scenarios

CRM systems are typically part of a larger solution ecosystem. While it may be simplest to provide users with access to all business related information within a single system, it is typically impractical to account for all business needs within a single system. As a result, integration becomes a core component of these solutions.

Customers with data sharing requirements typically need to address the following types of scenarios:

- Migrating data from a legacy system to the CRM system
- Updating the CRM system with data from external sources at regular intervals
- Synchronizing a subset of data in the CRM system with an external system
- Updating data in the CRM system based on events or actions invoked by external systems
- Delivering functionality or specific processes offered by an external system that specializes or is certified for that capability

A summary of the reasons that customers might require an integrated solution, together with some details of the associated rationale, are provided in the following table:

Need	Rationale
Access enterprise capabilities	▪ Provide seamless access to users ▪ Streamline process ▪ Consistency of process; reuse existing process rather than recreating a duplicate
Enable a single view of customer	▪ Ensure staff have all the relevant information about customers at point of interaction
Share insight	▪ Drive use of customer and market analysis to the point of interaction ▪ Don't simply use analysis in the back office ▪ Use interactions to improve analysis data
Reduce costs and time-to-market	▪ Reuse existing capability ▪ Costs and time taken to rebuild and replace existing systems prohibitive

Integrating multiple systems is often a better solution that creating an all encompassing implementation to serve multiple business processes. Taking the integration approach allows the solutions to use best-of-breed components that may be built on different technologies, or to follow different data paradigms and deployment models (Self-Hosted, Partner-Hosted, or Cloud). The Microsoft Dynamics CRM 4.0 platform itself leverages this approach to accommodate reporting, with reports hosted and rendered by SQL Server Reporting Services (SSRS) and Workflow executed by the Windows Workflow Foundation engine.

Integration does not necessarily require data to be transferred between systems. What is important is that the integrated solution meets the specific needs of the business. As a result, an integrated solution might make data available seamlessly to the end user or ensure that certain actions are taken under specific circumstances. In many cases, User Interface Level integration is sufficient to meet the business needs, but in others, integration at the Application or Data level may be required.

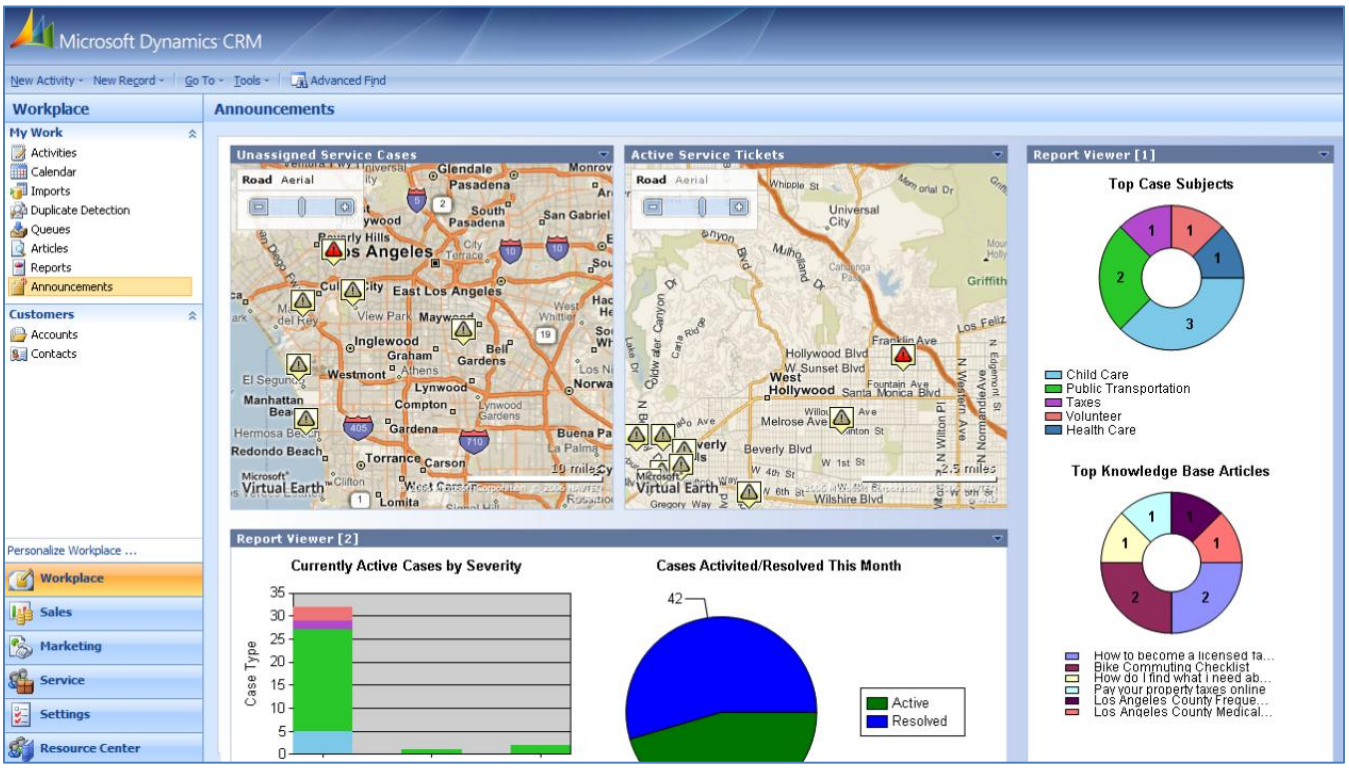
Levels of Integration

Depending on customer requirements, there are many levels at which interactions can occur between different systems within an organization.

User Interface Level

Integration at the User Interface (UI) level is typically used to connect disparate, non-related (or loosely related) systems. UI level integration is often used to provide direct access to information that is not part of core processes and can be achieved via Dashboards, Mash-ups, IFrames, or other UI constructs. The data surfaced in the solution resides on an external system and is only available during the user session.

An example of such integration between Dynamics CRM and the Microsoft Virtual Earth mapping application is shown in the following graphic:



The Dynamics CRM platform exposes URL addressable UI elements, such as Forms and Views, which enable an application to directly link to a CRM page simply by using a URL with no significant development effort. This technique provides a simple integration with the CRM look and feel and the CRM UI can be embedded directly into other applications in an IFrame or browser control.

While UI level integration techniques are not covered in detail in this paper, some common characteristics of such solutions are below:

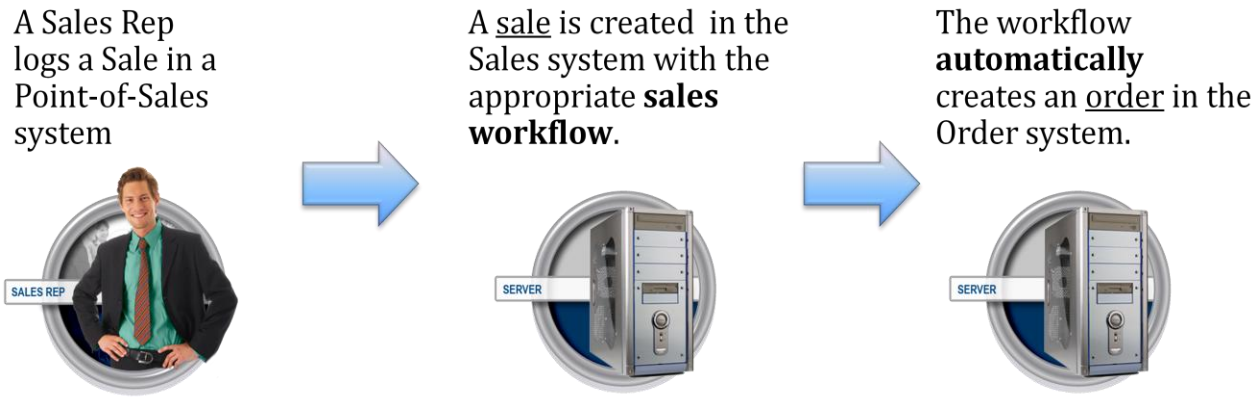
- Simple URL-based integration
- No need to replicate functionality or data
- Can link to external resources e.g. Bing Maps, SharePoint, other Line-of-Business applications, etc
- Support different security models across external systems
- The data is not resident on the local system and hence:
  - The system is not able to trigger workflows or plug-ins based on CRM events
  - The user cannot directly search data in advanced find/report wizard
  - The user cannot create CRM relationships between external data and other entities

UI level integration creates a unified interaction surface for the user with minimal work. It is most effective for cases in which a user simply needs to view the data from an external system. There are limitations around authentication and security with this approach since the external system may have a different security model.

Application Level

Application level integration is typically implemented as part of a business process or workflow. This level of integration is usually driven by events, triggers, or workflows, and CRM Web Services are used to interact with the external systems. All the data is stored in the local system and is updated though interactions with external systems.

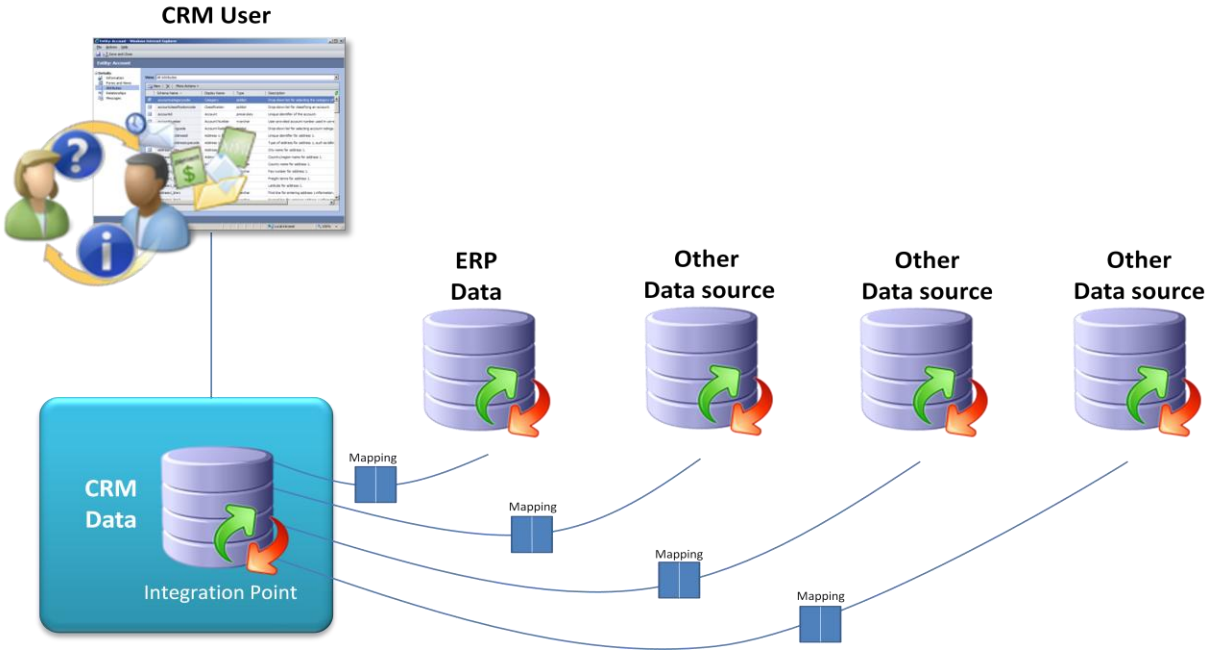
A typical example of the need for integration at the application level would be a business scenario that includes both Sales and Order management systems. Application level integration would ensure that recording a sale in the Sales system initiates an event that issues a request to the Order system to create an entry for the order.



CRM systems most commonly integrate at the application level. Using CRM Web Services reduces the dependency between the systems and ensures the functionality of the integration in future versions of the product. In addition, this level of integration does not rely on the UI layout (unlike UI level integration), so the developer can select only the data required from the external system. This approach also allows for adding interactions about which a user may not need to be aware, e.g. passing some requests through quality control checks.

Data Level

Data level integration is based on the transfer of data between independent systems. This synchronization can be a one-time effort to transfer data from legacy systems (data migration scenario) or a solution that synchronously or asynchronously transfers data between systems.



Data level integration is commonly used for scenarios in which data resides in one or more central databases and external systems pull data from those systems. Each external system requires independent access to the data, and the schema of the shared data may vary greatly across those systems.

Integration Techniques

The most common method of interacting with CRM data is via CRM Web Services. There are also several data synchronization solutions built using CRM Web Services and tools like BizTalk. Dynamics CRM also provides read-only SQL Server Filtered views which can be used for Business Intelligence (BI) and Reporting solutions. SQL Server Replication is also an alternative to data synchronization solutions for certain scenarios.

The following sections provide details about each of these techniques:

- CRM Web Services and Web Services based solutions
- Filtered Views
- SQL Server Replication

CRM Web Services and Web Services based solutions

CRM Web Services are the recommended method for interacting with Dynamics CRM data and integrating with external systems (other ERP, CRM systems, dashboards, etc). Accessing CRM using the web services layer ensures that user authentication and authorization, data integrity, referential integrity and plug-ins and workflow rules are respected. The web services layer also abstracts the underlying database schema which allows solutions to work on subsequent versions of Dynamics CRM even if the underlying schema is modified.

Customer Scenarios

Consider the following customer scenarios, in which each customer has two tenants.

Scenario 1. One tenant manages Orders and the other tenant maintains the list of Customers and keeps a count of their orders in the system. When a new order is created, an action is triggered via CRM workflow or Plug-ins and a web service call is made to the Customer system to update the order count for that customer.

Scenario 2. One tenant manages the Sales process and the other tenant manages the Marketing process. Both tenants require access to the list of Accounts and need to be able to update the Account details from either tenant. The customer uses a custom Web Services based solution to synchronize Account data between the two tenants. Any changes made at one tenant are propagated to the other.

Solution Details

Dynamics CRM offers a web service API that enables other applications to make data requests of CRM. This allows for simple integration for either data retrieval or making changes in the local CRM system by an external system. CRM data can also be presented directly in the User Interface of an external system by invoking the CRM SDK.

Note: For additional information about the Dynamics CRM SDK and samples that describe how to interact with the CRM system, see the Microsoft Dynamics CRM Developer Center at: http://msdn.microsoft.com/en-us/dynamics/crm/default.aspx

Web Services based synchronization solutions are the best option for synchronization of data that may need to be updated in multiple locations. These types of solutions are typically built by using custom code with BizTalk or SSIS, and they may require significant effort to deploy and maintain. Third-party vendors also offer powerful packaged solutions for Microsoft Dynamics Integration and Migration which requires minimal development.

CRM Web Services are well suited for event-based integration. Web services, in conjunction with CRM Plug-ins and Workflows, address the majority of application level integration scenarios.

A key characteristic of Web Services based solutions is that they operate under the premise that the data source and destinations have different schemas, security models, and validation rules. These solutions extract data from the CRM system, transform it to map to the target system, and make a call to the target system to verify the input data against the target system. This process requires a lot of processing, which is necessary to ensure system integrity. Tuning the Dynamics CRM system may be necessary to achieve a high data level of performance if the solution is synchronizing a large amount of data in a limited amount of time.

Note: For additional information about the optimization techniques that should be applied to ensure optimal performance, on Microsoft Downloads, see the white paper *Optimizing and Maintaining Microsoft Dynamics CRM 4.0 (v2*) at: http://www.microsoft.com/downloads/details.aspx?FamilyID=ba826cee-eddf-4d6e-842d-27fd654ed893&displayLang=en

Filtered Views

Microsoft Dynamics CRM includes SQL database filtered views that are used for business data access. Filtered views are fully compliant with the Microsoft Dynamics CRM security model and are best suited as a data source for Reporting and Business Intelligence (BI) applications. However, they can be used by any solution that requires read-only access to CRM data. Accessing data by using filtered views is much more efficient that by using Web Services, and filtered views support standard SQL Server reporting tools.

When an application or user accesses a filtered view, the Microsoft Dynamics CRM security role and access rights associated with that specific application or user determines which data will be accessible. Data in filtered views is restricted at three levels: the organization, the business unit, and the owner. Filtered views exist for all Microsoft Dynamics CRM business objects (entities) including custom entities.

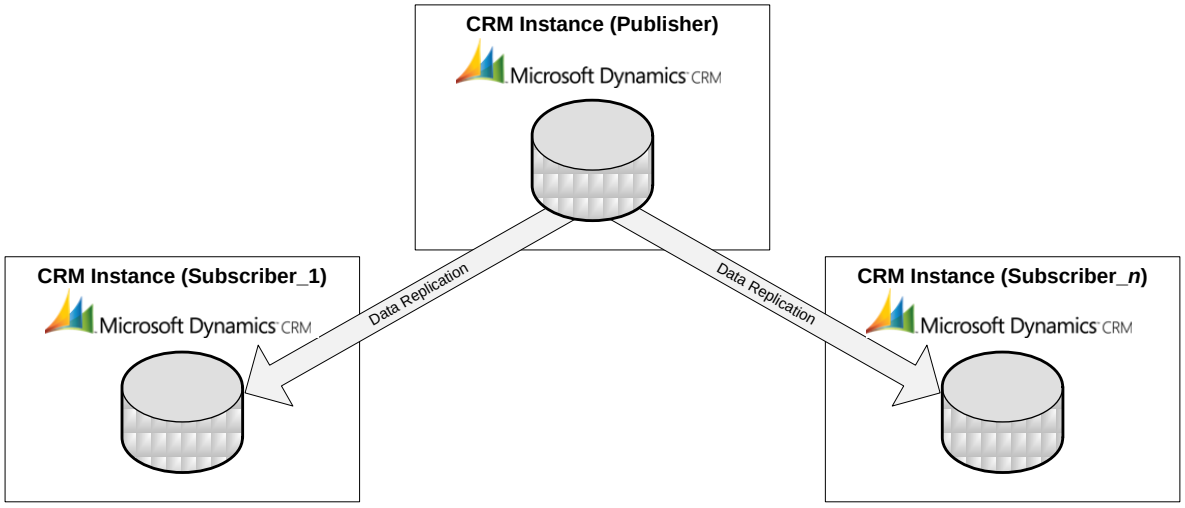
Filtered views also provide the ability to easily pull Microsoft Dynamics CRM data into Microsoft Office applications, such as Excel and Access.

Note: Microsoft Dynamics CRM Online does not provide data access via filtered views.

SQL Server Replication

SQL Server Replication can be used for scenarios in which a CRM system needs to share read-only copies of data with other CRM systems. This technique is applicable only for scenarios in which entities have a static data schema and a limited number of relationships; it is not recommended for sharing transactional data. This approach also requires that the schema of the shared data be the same on the Publisher and Subscriber CRM systems. It is important to note that if the schema of the data needs to be changed (via CRM customization), the replication relationship has to be suspended, customizations applied on both the Publisher and Subscriber instances, and replication relationship reestablished.

Unlike the Web Services solution, this technique does not transform the data, but instead relies on the Publisher CRM instance to ensure data integrity and apply all post processing (using workflows or plug-ins), and simply replicates the data to the Subscriber system. This solution typically has significantly superior performance over web services based solutions.



The solution described in this section uses one-way transactional replication. Dynamics CRM does not encapsulate SQL requests belonging to a business transaction into a SQL transaction; hence SQL Server Peer-to-Peer, Merge Replication or any data-level multi-master replication solution should not be used since it may leave the CRM system in an inconsistent state.

Customer Scenarios

Consider the following customer scenarios:

Scenario 1. A customer has 3 tenants. One tenant manages the Customer accounts across the company, while the other two tenants are used for Sales processing and Order management. Both tenants need Customer account information to associate customers with the sales teams and orders, and to contact the customer if needed. A SQL Replication solution can be deployed that shares only the Customer account information with the two tenants.

Scenario 2. A software company's headquarters has a CRM tenant that maintains a list of customers worldwide. Across the world, there are multiple CRM tenants that handle the company's incident management process in their respective geographies. Each tenant requires access to the Customer list but also needs to be able to work independently, even without access to the network at headquarters. A SQL Replication solution is ideal in this scenario, performing very effectively over Wide Area Networks (WANs) with inconsistent connectivity.

Solution Details

This solution is predicated on a CRM instance writing data to its tables, and that data being replicated over to another instance with the same schema. A CRM organization is designated as the Publisher of the entities and makes the entities available to other organizations by publishing them using SQL Server Replication. Multiple organizations may then subscribe to the published entities and the related data will be replicated to their CRM databases. The replicated data would be read-only at the Subscribing organizations and any writes (which can be blocked) will be overwritten by the Publisher Organization.

A replication-based solution has significantly better performance than a web services-based solution because it does not perform any translation between the data schema of the CRM deployments since the deployments are kept in sync by replication. Another key advantage is the out-of-box queuing that is provided by SQL Server, which allows the systems to work independently if any one of them is unavailable and which synchronizes all changes when those systems do become available. A custom web services-based solution would have to handle that logic. This solution is suited for scenarios in which performance is a premium and only read-only copies of data are required to be shared.

Note: Details and a walkthrough of how to configure this type of solution in a Virtual Machine environment is provided in the accompanying white paper *Sharing Data across Microsoft Dynamics CRM Deployments – Virtual Machine Build Guide*.

Types of Sharing

Customers can choose to share an entire entity (Full Share) or a subset of the attributes associated with an entity (Partial Share).

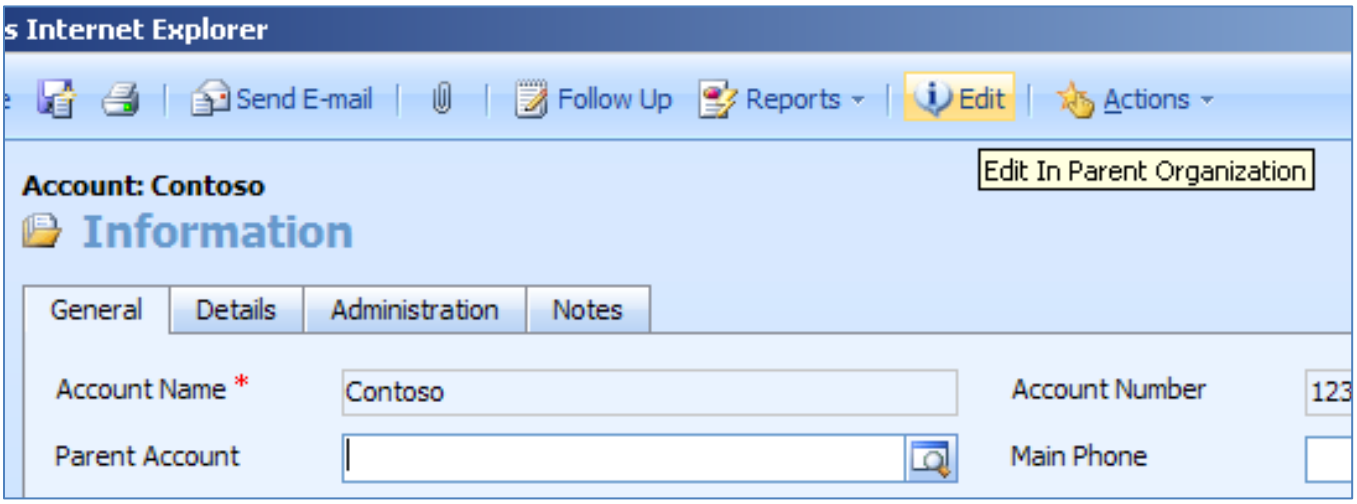
The Pros and Cons associated with each type of sharing are listed in the following table.

	Full Share	Partial Share
Pros	▪ Easier to setup and maintain. ▪ Works best with custom entities since some system entities contain reference to other entities in the entity record itself which makes it more complicated to replicate. ▪ Deletions can be supported more easily because the entire record can be deleted at the CRM Publisher and Subscribers.	▪ Only publishes the fields that need to be shared. ▪ Subscribing organizations may add to the schema of the entity as needed as long as the shared fields are not removed.
Cons	▪ The schema of the entity being replicated is fixed and the Subscribing organizations cannot customize the entity and add any attributes to it.	▪ Management is more involved. ▪ Deletion may not be supported since additional fields added to the entity should not be deleted by the parent,

Solution Impact

User Interface

To prevent users from attempting to write to the read-only fields in the database, a replication solution such as the one described in this section would involve some level of UI customization. One approach would be to mark the shared fields as read-only (using CRM customization tools) at the subscribing deployments. If the users of the subscribing deployments need to update the record at the Publisher, the form can be customized either by showing the URL of the entity at the Publisher or by adding a button that opens the appropriate URL.



Security

Security in Dynamics CRM is managed within the organization database. After user is authenticated via Active Directory or other means, the user's authorization is verified by using CRM database records that contain information about the Teams and Business Units to which a user belongs, as well as the entities to which a user has access (and the level of that access - read only, update, delete, etc.). This data is typically not replicated across CRM deployments. To account for this, the solution must ensure that new records created in subscribing entities are correctly assigned to users in the local tenant. In the associated white paper *Sharing Data across Microsoft Dynamics CRM Deployments – Virtual Machine Build Guide*, all new records are simply assigned to the CRM Administrator by using database triggers. However, this behavior would be adapted as necessary to accommodate the customer's requirements.

It is important to note that after a piece of data is shared by the Publisher, the Subscribing tenants can assign that data according to their business needs. In addition, Subscribing tenants may add business logic to enforce stricter control on access to the data.

Workflows and Plug-ins

A key factor affecting event based behavior (such as workflows and plug-ins) is that shared data on the subscribing systems is not changed via CRM web services, and hence is not linked to the Dynamics CRM event pipeline. If a deployment includes a workflow that sends an e-mail when a CRM Order is created, that workflow will not be triggered if the Order is created directly in the database. This highlights the fact that the replication solution described in this section might result in more complex, potentially cross-tenant, workflows and plug-ins.

Operation Behavior

Major aspects of the operation's behavior are described in the following table.

Event	Description
Create	Shared entities can only be created at Publisher tenant. After the entities are created, the data is replicated to the Subscribers.
Retrieve	All data is retrieved from the local database.
Update	Updates to shared data are only allowed at the Publisher. In case of Partial Share, fields not shared from the Publisher can be updated on the Subscribers. Consider the following example. The Publisher shares the ID, Name, and Gender attributes of the Player entity with Subscribers, and then Subscriber_1 updates the Player entity by adding a Description attribute. In this scenario, updates to the Player ID, Name, or Gender must occur at the Publisher, while updates to Description can occur at the Subscriber because the field is not shared.
Delete	Handling deletes is challenging given the cascading nature of deletes and the added data created by the subscriber in the Partial Share scenario. Options include: ▪ Preventing deletes of shared entities; mark them as disabled instead. ▪ Intercepting the delete event at the Publisher tenant and issue a delete on each child via web services. ▪ Allowing Subscribers to delete shared entities, but the delete has no effect on the data stored at the Publisher.

Important: The best approach for a specific customer depends on that customer's business requirements.

Caveats

Be sure to consider the following caveats:

- Implementing this solution requires a Database Administrator and CRM Developer/Administrator with a thorough understanding of SQL Server Replication and the Dynamics CRM entity model. This is necessary to ensure that the correct set of data is replicated, all references are updated accordingly, security and authentication issues are considered, and the effect on CRM customizations is accounted for.
- Customization of CRM entities results in changes to the entity schemas, and replication must be suspended during this process. Note that the Publication may also need to be updated if the shared data schema has been modified.
- CRM benefits from holding data directly within its database
    - Advanced Find, Views and tools like the Reports Wizard can be used since all the data resides in Dynamics CRM
    - Workflows and Plug-ins can be triggered against data (as previously discussed in this paper)
    - Relationships between data can be established easily
- Managing Backup and Restore, and Disaster Recovery of this environment is also fairly complex. The procedures have to be managed in a way that in case of a failure, the data can be restored from a backup (at either the Publisher or Subscriber or both) and the replication relationship recreated and resynchronized.

Supportability

Commercially reasonable support will be provided for the SQL replication based solution described within this document. Commercially reasonable support is defined as all reasonable support efforts by Microsoft Customer Service and Support that do not require Microsoft Dynamics CRM code fixes. This support extends to Microsoft provided code and samples, but not to custom developed code which is the responsibility of the ISV or developer.

Summary

This paper has discussed the different levels at which sharing data between multiple CRM organizations can occur, as well as details about the techniques associated integration at the Application and Data Levels.

It is important to note, however, that the paper does not prescribe particular solutions for specific scenarios. No single solution is the right solution for all customers. Each solution needs to be mapped to the business requirements of a specific customer.

A pitfall in many projects can be the tendency to over-integrate and synchronize more data than is necessary, which then has to be managed over the data and business process lifecycle. As a result, attempt to address the requirements first at the User Interface Level, progressing on to the Application Level (and then Data Level) only if necessary to meet the business need.

Also keep in mind that while CRM Web Services, Filtered Views, and SQL Replication all are important techniques for sharing data between CRM deployments, each requires a varying level of CRM and Database knowledge. A solution built using SQL Replication may offer the highest throughput of these three techniques, but it also works on a layer below the CRM business logic. As a result, this type of solution in particular should be carefully designed by a CRM Architect with a thorough understanding of SQL Server and Replication technologies.

Appendix A: Additional Resources

Consider the following additional resources:

The Microsoft Dynamics CRM Developer Center
http://msdn.microsoft.com/en-us/dynamics/crm/default.aspx

Filtered Views
http://msdn.microsoft.com/en-us/library/cc308184.aspx

Updated Dynamics CRM 4.0 Adapter Now Available – Includes Support for BizTalk Server 2009
http://blogs.msdn.com/crm/archive/2009/09/10/updated-dynamics-crm-4-0-adapter-now-available-includes-support-for-biztalk-server-2009.aspx

Appendix B: Overview of SQL Replication

Replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases to maintain consistency. Using replication, you can distribute data to different locations and to remote or mobile users over local and wide area networks, dial-up connections, wireless connections, and the Internet.

Transactional replication is typically used in server-to-server scenarios that require high throughput, including: improving scalability and availability; data warehousing and reporting; integrating data from multiple sites; integrating heterogeneous data; and offloading batch processing. Merge replication is primarily designed for mobile applications or distributed server applications that have possible data conflicts. Common scenarios include: exchanging data with mobile users; consumer point of sale (POS) applications; and integration of data from multiple sites. Snapshot replication is used to provide the initial data set for transactional and merge replication; it can also be used when complete refreshes of data are appropriate. With these three types of replication, SQL Server provides a powerful and flexible system for synchronizing data across your enterprise.

In addition to replication, in SQL Server 2008, you can synchronize databases by using Microsoft Sync Framework and Sync Services for ADO.NET. Sync Services for ADO.NET provides an intuitive and flexible API that you can use to build applications that target offline and collaboration scenarios.

Note: For additional information about SQL Server Replication, on MSDN, see the article *SQL Server Replication* at:
http://msdn.microsoft.com/en-us/library/ms151198.aspx

For an overview of Sync Services for ADO.NET, on MSDN, see the following resources:

- *Microsoft Sync Framework*:
  http://msdn.microsoft.com/en-us/library/cc281959.aspx
- *Microsoft Sync Framework Developer Center*:
  http://msdn.microsoft.com/en-us/sync/default.aspx