Synchronization

The Agiloft synchronization subsystem provides facilities for automatic synchronization between Agiloft tables and corresponding records in some external system. Synchronization can be bidirectional or unidirectional in either direction.

Syncing can be used to transmit all the functionality that an external system has in common with Agiloft. For example, it allows the user to define table and field mappings, handle the communication of the data and determine which records need to be updated between each system. Each external system type also needs an external system Adaptor (ESA) to handle its unique characteristics. The ESA is responsible for implementing functions such as "List the available tables" "List the fields in a given table", "Create, Replace, Update, Delete a record". It is also possible to plug in a fully custom, third party ESA implementation and to share ESAs that you create with a particular system type.

Agiloft comes with a prebuilt ESAs for common systems and new ones can be created by customers in Java or their choice of language. If the customer develops the ESA in Java, it can be linked with the Remote Proxy that actually communicates with the Sync subsystem. If the customer develops the ESA in some other language, it communicates with the Remote Proxy using standard I/O streams.

To be synchronizable with Agiloft, an external system must have these properties:

- It must be possible to obtain some kind of modification timestamp for each record. Typically this timestamp is held in the record itself, but this is not absolutely necessary.
- Record IDs should not be changed or at least there must be a way to know any record ID as it was on the last sync. Record IDs can be "remapped" on a sync, but the old ID must be known to the ESA. This is used to track record matches.
- Records from the external system, as presented to Sync by the ESA, must be grouped by structures, which are mapped precisely to Agiloft types or Tables, such as Contact/Employee. A structure can be a table, a folder or anything else that logically groups records in the external system. Within a given structure, external records must use the same set of field mappings.

A given Agiloft table can be synced with multiple external system and it is not necessary for all fields to be mapped. Existing ESAs provide support for synchronization with the File Directory, Microsoft Exchange, QuickBooks, Excel and other systems. The synchronization process is based on comparing record timestamps in the Agiloft KB with record timestamps in the external system. For example, if a synchronization was run on Wednesday and is next run on Friday, it will look for all the records changed in one or both systems since Wednesday and will propagate those updates to the other system. At the end of a synchronization process, both systems will have matching data. Creation of new records or deletions may also be carried over to the other system if configured to do so.

Sync Terminology

Agiloft	Agiloft Inc/ Agiloft Installation
Sync	Agiloft Synchronization Sub-system

XML	Extensible Markup Language. See http://www.w3.org/xml/
external system	external system - separate from Agiloft
STDIO	Standard Input/Output streams of console
HTTPS	Secure Hypertext Transfer Protocol
URL	Universal Resource Locator
URL Encoding	Encoding as prescribed in http://www.w3schools.com/TAGS/ref_urlencode.asp
NAT	Network Address Translation
CL	Command Line
ESA	external system Adapter
ESA Proxy	Interface between the Agiloft sync subsystem and the ESA.
JDK	Java Development Kit
REST	Representation State Transfer web service. See: http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html
RPC	Remote Procedure Call

Synchronization Options

Agiloft supports several synchronization options. The choice of which method is most appropriate depends upon your particular needs.

The following table summarizes these options:

Sync Option	Description	Pros and Cons

Web Services /REST	Used for general purpose, request-response integration with remote or local systems. **Example** A website PHP script, which needs to check that there is a People table record in the Agiloft knowledgebase for some user email for a registration check.	**Pros:** The knowledgebase can be seen as an ordinary source, like a database; web services are now supported by virtually any platform. **Cons:** ■ Inflexible: the program is usually hard-coded and must be manually altered to add mappings for new tables or change existing mappings. ■ Low-level: extensive coding is required to provide reliability in case of connection failures, feedback on the Sync status, etc. ■ Moderate performance.

ESA	Best used to synchronize record copies between an Agiloft KB and an external system. When a record is modified on one system, it is automatically updated in the peer system, either immediately, or on a schedule, or by a manual sync request. To achieve this functionality, the ESA must be able to connect to the Agiloft sync subsystem, and support Create/Read/Update/Delete (CRUD) operations in the external system, which can be remote.	**Pros:** ■ The ESA is high-level. It only needs to support CRUD operations in the external system and understand that system's data model. ■ There is no need for the ESA developer to understand anything about Agiloft or even have access to the production KB. ■ Flexible. Once an ESA has been written, it can be used with other Agiloft KBs without modification. The table and field mappings can be changed by the Agiloft administrator without any code changes to the ESA. ■ Developing an ESA is relatively easy because much of the work, such as comparing modification times, conflict resolution, and error processing, is done in the sync subsystem. **Cons:** ■ The external system must actually store records to sync with and these records must have a clearly-defined ID and timestamp. ■ Moderate performance unless the method to process multiple records in batch mode is implemented.

Direct Database Integration	Agiloft stores all KB data within a database. The table and fields with their database names are exposed via the sync GUI, so it is possible to query the database directly from the remote system.	**Pros:** ■ Very flexible, can do anything related to reading data. Full data access. Can use report generation and data mining tools. ■ Very high performance. **Cons:** ■ Should be read-only. The external system should not write directly to the Agiloft database. ■ Easy implementation for simple tables and fields, but more complexity for table relations and advanced field domains, where the developer would have to understand internal Agiloft data structures that are subject to change.
Custom Scripts	Custom scripts can be run by Agiloft upon some KB-defined conditions, such as when modifying a record. These scripts are executable and there is prebuilt support for Perl and Java scripts which are on the Agiloft server. Run conditions can be quite complex and can be fully integrated with a workflow set up in the KB. **Example** Post data into Active MQ and generate a file in the shared network folder when a "Helpdesk Case" record is put into a "Scheduled" state and a staff person is assigned to work on the record.	**Pros:** ■ Fully integrated with workflow and business rules. ■ Very easy to write, easy access to record data. ■ Support for both reading and writing. **Cons:** ■ Data access is limited to the individual record provided by the business rule or workflow.

Scope

The structure with which an Agiloft table can be synchronized may be a table, a folder or anything else which logically groups records, and can be presented or derived by the ESA from the external system. All records within a single structure must have the same set of fields.

As well as records, it is also possible to synchronize relations between external records. Those are used to update Linked Fields in Agiloft.

Sync also supports Collection fields, which are something in between normal data field, holding single value and a relation to other table. Collection fields hold multiple values, which might be complex objects, with multiple fields. These objects can be mapped to a separate Agiloft table, just as related records for relations. But, unlike a relation, which treats all records independently, these records are treated as a part of their master record. If any of them is modified, a master record is considered to be modified.

Sync Configuration

In summary, the Sync Configuration Wizard creates a record in Agiloft holding:

- The external system to sync with
- The tables to be synchronized and their corresponding "Structures"
- The fields to be synchronized within a Table-Structure pair
- The relations to be synchronized
- The directions - Agiloft to XS, XS to Agiloft or both - of updating
- The command that launches the ESA
- The "identifier" fields Sync will use to identify matching records

If a Sync Configuration is modified after it has been run, all previous peering of Agiloft records with XS records is lost, and the next run will be an entirely new synchronization.

Record Peering

For every record in an Agiloft table or XS structure subject to a new sync, and for every such record created since the last sync, a corresponding record in the other system is determined, if it exists, by matching the Identifying fields - checked in the Field Mapping Wizard - from the two systems. If there's no such record, one is created. The two records are held by Sync in an Agiloft ID - XS ID pair, thus peering the Agiloft record with the external record. Logically, Sync treats the pair as a single data record, having Agiloft and external views.

If a new record is created manually in both systems, the Identification process will prevent Sync from duplicating them.

Matching

There are two algorithms for matching:

- Best match: If there are 3 identifying fields, Sync will first try matching all 3. If there is no match, it will try matching by any 2 of 3. If there is still no match, it will match by any 1 of the 3 fields.
- Exact match: If there are 3 identifying fields, Sync will first try matching all 3. If there is no match it will create a new record.

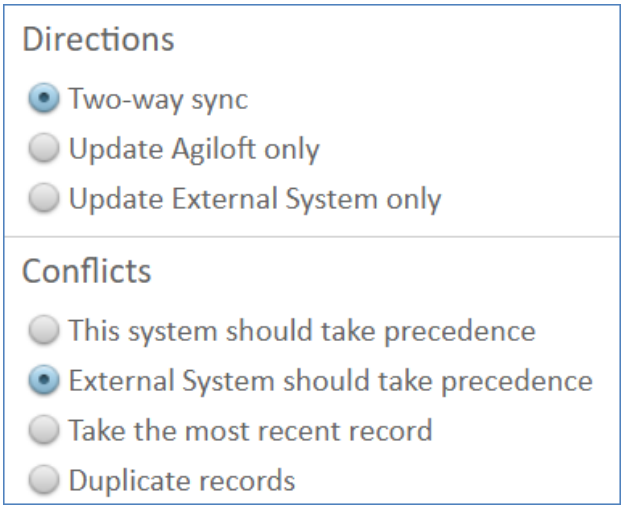
The algorithm is selected in the Field Mapping GUI - Use strict match for identification checkbox. The choice of exact matching may prevent matching if any identifying field has non-unique values.

After the peering is established, it will not be cleared until either one of these conditions is met:

- The records are deleted in both systems
- The peering information is reset by using the Reset Records Peering button on the General tab of the Sync Configuration edit GUI.

Directional Synchronization and Conflict Resolution

There are several similar options that control how the External Sync Wizard handles the direction of synchronization, and specify how the system should handle conflicts that arise when the same records are updated in both systems. For example, the General tab of the External Sync Wizard allows you to define the direction of the sync as two-way, or only updating Agiloft or the external system. Additionally, you can define how the synchronization handles record conflicts between the systems.



- Two-way sync - allows mapping between objects in both directions - from Agiloft to the external system, and the other way around. Records of the mapped tables will synchronize in both systems.
- Update Agiloft only - allows records to be updated in Agiloft with values from the external system.
- Update External System only - allows records to be updated in the external system with values from Agiloft.

When there are two conflicting files, the system will always choose to keep one and discard the other. This can be based on the timestamps in the Last Updated Field for each record, or on the system which has been specified to take precedence. However, it is possible to specify the fields which should be overwritten in the Field Mapping Tab. Conflict handling in the General tab works in this way:

- **This system should take precedence** - If both records have been changed to different values, the Agiloft record will be updated.

Example

- Last sync was performed at 07:39am
- Last modified time of Agiloft record (e.g. Agiloft_1.doc): 07:45am
- Last modified time of the external system record (e.g. Agiloft_10.doc) at Salesforce 07:47am

The sync is run at 07:52am.

In this case the record with Agiloft values (Agiloft_1.doc) will take precedence.

- **External System should take precedence** - if both records have been changed since the last sync to different values, the external record will be updated.

Example

- Last sync was performed at 07:39am
- Last modified time of Agiloft record (e.g. Agiloft_1.doc): 07:45am
- Last modified time of the external system record (e.g. Agiloft_10.doc) at Salesforce 07:47am

The sync is run at 07:52am.

In this case the record with external values (Agiloft_10.doc) will take precedence.

- **Take the most recent record** - if both records have been changed since the last sync to different values, the most recently modified record will be updated

Example

- Last sync was performed at 07:39am
- Last modified time of Agiloft record (e.g. Agiloft_1.doc): 07:45am

- Last modified time of the external system record (e.g. Agiloft_10.doc) at Salesforce 07:47am

The sync is run at 07:52am.

In this case document Agiloft_10.doc will take precedence.

- **Duplicate records** - the system creates a duplicate of each record with both external system and Agiloft values.

Example

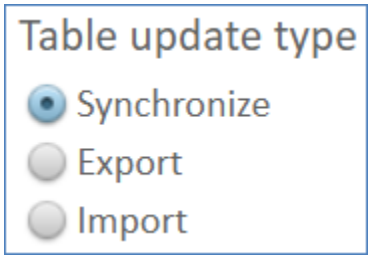
- Last sync was performed at 07:39am
- Last modified time of Agiloft record (e.g. Agiloft_1.doc): 07:45am
- Last modified time of the external system record (e.g. Agiloft_10.doc) at Salesforce 07:47am

The sync is run at 07:52am.

In this case both the Salesforce and Agiloft systems will have a duplicate of Agiloft_1 and Agiloft_10.

Table Update Type

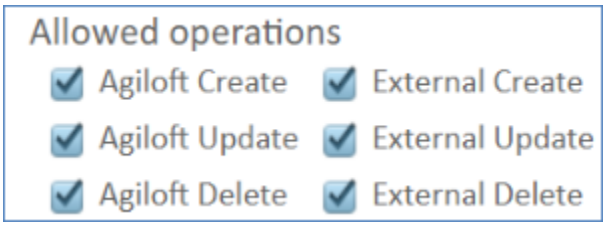
In the Field Mapping wizard, the Table Update Type options define the sync direction for that specific table. The available options here are dependent on the sync direction. For example, if sync direction = Update Agiloft only, the available Table update type options will be Synchronize and Import.



- Synchronize - compares data and resolves conflicts using the options in the General tab.
- Export - only allows data to be exported from Agiloft to the external system for this table. This will forcibly overwrite external records and always request all data from Agiloft.
- Import - only allows data to be imported from the external system to Agiloft for this table. This will forcibly overwrite Agiloft and always request all data from the external system.

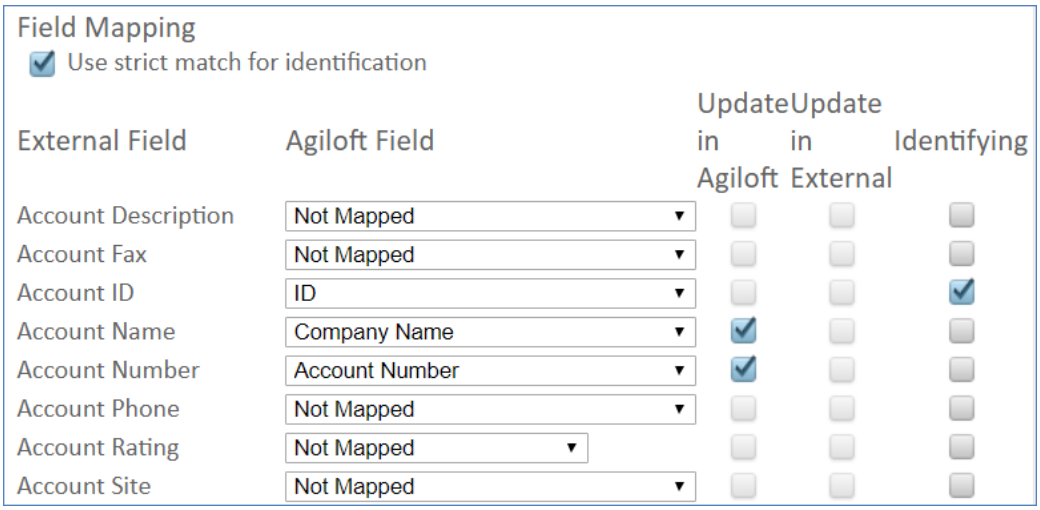
Export and Import do not attempt to synchronize the records by matching values and resolving conflicts. Instead, they simply overwrite all fields which are selected for mapping for the table.

Allowed Operations



The allowed operations restrict Create/Update/Delete operations for the tables in the internal or external systems. These options are dependent on the sync direction, and if the direction does not permit an operation it will be disabled. For example, if Sync Direction = Update External System only, Agiloft Create/Update/Delete will be greyed out here.

Field Mapping



Besides the table, each field also has an option to be updated in Agiloft or the external system. As with the Allowed Operations, these options are dependent on the sync direction. Depending on the field selections here, the sync will work in the following ways:

Example

- If Sync Direction = Two-way sync, and Allowed Operations = Agiloft Create/External Create, and Field A = Update in Agiloft - then during sync, the field will be created during the first sync, and not updated or removed if the field value changes.

- If Sync Direction = Update Agiloft only, and Allowed Operations = Agiloft Create/Update/Delete, and Field A = Update in Agiloft - then during sync, the field will be created in Agiloft during the first sync, and updated or deleted whenever it changes and re-synchronizes in the external system.

Dual Synchronization Problem

If two or more external systems are kept in sync with a given Agiloft Table - which means using two or more different Sync Configurations - and if conflicts are resolved by preserving the most recent value, the results are affected by the order in which the synchronizations are run.

Suppose an updatable field in the 3 systems XS A, XS B and Agiloft has the values and timestamps shown, and we synchronize A with Agiloft, and then B with Agiloft at 4p.m:

```
XS A p (3p.m) => p (4p.m)
XS B q (2p.m) => p (4p.m)
Agiloft r (1p.m) => p (4p.m)
```

The first update changes Agiloft because 3 p.m. > 1 p.m, and the second changes XS B because 4 p.m > 2 p.m

But if we synchronize B first, and then A, the effect is:

```
XS A p (3p.m) => q (4p.m)
XS B q (2p.m) => q (4p.m)
Agiloft r (1p.m) => q (4p.m)
```

The first update changes Agiloft because 2p.m > 1p.m, and the second changes XS A because 4p.m > 3p.m.

This is because there is only one timestamp on each value in a pair, whether it results from sync or user updating.