

Specification: CRM/ERP

Version: 1.0

Published: Jan 22, 2001

Written by: Torbjörn Eliasson, ITtecture
Claes Steen, Hogia
Lars Sigebo, Lundalogik
Dahn Nilsson, XOR
Rikard Gårdshage, SuperOffice

PREFACE: THE DATA EXCHANGE STANDARD CRM-ERP.....	4
INTRODUCTION.....	6
GOALS FOR THE SPECIFICATION	6
OVERVIEW	6
WHAT THE SPECIFICATION DOES NOT COVER	8
TRANSPORT	8
ALL KIND OF INFORMATION.....	8
SECURITY.....	8
MATCHING CUSTOMERS	8
SYNCHRONISATION	9
INTRODUCTION	9
IDENTIFIER.....	9
PROCESS.....	9
CONNECT	9
UPDATE.....	10
REJECT	11
DISCONNECT	11
DELETE	11
SEQUENCE NUMBERS	12
ERROR HANDLING AND LOGGING.....	12
CERTIFICATION	13
INTRODUCTION	13
LEVELS.....	13
TYPE	13
DIRECTION	13
CERTIFICATION MATRIX	13
EXAMPLES	13
RESTRICTIONS	14
CERTIFICATION PROCESS.....	14
XML SCHEMA.....	15
OVERVIEW	15
ADDITIONAL INFO	15
COMPLETE SCHEMA REFERENCE.....	16
XML SCHEMA DOCUMENTS	16

OTHER	16
APPENDICES	17
A. DELIVERY OF MESSAGES WITH THE FILE SYSTEM	17
OVERVIEW	17
FOLDERS	17
NAME RULES	17

PREFACE: The Data Exchange Standard CRM-ERP

This data exchange standard compromises a win-win-win situation.

The customer who uses Customer Relationship Management systems (CRM) and Enterprise Resource and Planning systems (ERP) will also be a winner. The vendor of the CRM system will be a winner as will the ERP vendor. They will all be winners with this data exchange standard.

Background: Companies often produce and store business critical information in different systems, in this cases the CRM system and the ERP system. In order to support the company's business processes, companies need overall information of their customer relations. That is information from both the CRM system and the ERP system. But those systems normally don't speak with each other, at least not voluntarily.

This fact implies several problems.

Usually the same information is entered into both the systems but from different perspectives. Retrieving what should be the same information from both systems is confusing. Most likely you would find different names, addresses and Sales figures of all your customers.

Sometimes the vendor, a consultant or the IT-department is asked to integrate the two systems. At first it might seem like a good idea, but is such a good idea in the long run? How should this integration be maintained? Every time one system is updated the integration has to be updated too. In reality there are not many companies that can afford such integration.

Solution: If you were able to track information in the financial system from the Sales system, or vice versa, you'd be happy. And if you were able to have information entered into one system, automatically copied to the other system (or vice versa), you'd be even happier.

That is what this data exchange standard is about.

Customers using CRM and ERP systems will gain from the standard, since problems like the ones above will be mostly solved.

Customer Relational Management system vendors, CRM (or Sales Force Automation vendors, SFA) will gain too. Important information from the financial system can be easily retrieved and tapped right into their systems. Now they can concentrate upon their core task: to improve their CRM/SFA-system. They do not have to make routines for manual entering or complex imports and exports.

The same gains will be there for Enterprise Resource & Planning vendors (ERP). Since their clients can use their ERP-system and easily integrate it with the customers own CRM/SFA-system; the ERP vendor can concentrate on making a better ERP system.

So, from all perspectives everyone gains.

Commitment: The framework and the concrete development of the standard have been contributed by the largest and most important CRM/SFA-systems on the Swedish market. That is Abalon, Lundalogik, and Superoffice (with former Caesar Business Systems).

Among the ERP vendors, the contribution has come from the largest and most important vendors from two segments. From the small to medium sized ERP-systems Hogia-gruppen, SPCS and XOR, have contributed. From the medium to large sized ERP-systems, Scala Business Solutions has participated.

With the experience and knowledge of integration (i.e. expertise) that these organizations have, we believe this standard solves a large part of the common needs for integration between CRM/SFA and ERP.

Everything in the standard is also made to comply with the latest development within the XML- technology and to adapt to international premises. For instance the organization Business Accounting Software Developers Association, BASDA, has been in close contact with the standards development to ensure that. The intention is to make sure this standard can be used worldwide.

From the organizations above, important initiatives and contribution has come from several people. Without their dedication this standard would not exist. They are:

- * Claes Steen, Hogia
- * Dahn Nilsson, XOR
- * Johan Linder, Superoffice
- * Karin Bergsdotter, SPCS
- * Lars Sigebo, Lundalogik
- * Lasse Bengtsson, Abalon
- * Margareth Wahlberg, Abit consulting
- * Peter Törnell, Superoffice (former Caesar Business Systems)
- * Rikard Gårdshage, Superoffice (former Caesar Business Systems)
- * Torbjörn Eliasson, ITecture
- * Erik Philipson, Scala Business Solutions

Enjoy your day and our standard!

SIE-gruppen

Erik Philipson

Board member and now Product Marketing Manager at IndustriMatematik/Abalon

Introduction

Goals for the specification

Standardized & public	The specification should be defined and maintained by an independent organisation, where a broad group of vendors are represented. It should be published for all who have an interest.
Robust	The integration should be reliable and stable.
Platform independent	No demands on special OS or development tools.
Basic	The specification should cover the basic needs so it can be accessed by a large group of applications.
Extendable	The specification should be extendable in the future.
Easy to implement for the suppliers	No special technique should be needed to implement. The need for large structural changes in the application is also avoided.
Easy to use and invisible for the users	The integration should not require the user to take any special action. He should be able to work as usual.
Flexible	The vendor should be able to only implement parts of the specification.

These are leading goals when the specification is created. It is also important to have them in mind when the specification is implemented in the application.

Overview

The specification focuses on information concerning customers, credit info, sales history and sellers. There is a specific messages for each type.

The specification is built to make it possible to establish a two-way integration. For those messages where it is applicable, i e customer and sales, it is desirable that as much

information as possible can be sent in both directions. For some messages like for credit info and sales history it is normally a one-way integration.

Nine different messages are defined.

Message	Description
SetCustomer	Contains all the customer information.
GetCustomer	Instructs the consumer to send all the information for a specific customer.
SetCredit	Contains all the information regarding the credit status.
GetCredit	Instructs the consumer to send all the credit information for a specific customer.
SetSalesHistory	Contains information of the sales history for a specific customer.
GetSalesHistory	Instructs the consumer to send the sales history for a specific time period for a specific customer.
SetSeller	Contains all the information of a seller.
GetSeller	Instructs the consumer to send all information for a specific seller.
Test	Contains response info from a test message.

The messages is described in detail in the section [XML Schemas](#). The schema file itself also contains some documentation.

What the specification does not cover

Transport

The specification is transport independent. There is nothing in the specification that connects the messages to any specific transport device. It is up to the adaptors to choose the technique that best suits their needs. As long as it handles XML documents in a proper way the specification will support it.

Due to the fact that there must be two application exchanging messages it is necessary to implement a transport device that could also be used by another vendor.

All kind of information

There has been no aim to create a specification that covers all aspects of a customer related application. The goal has been to make a standard at a reasonable level that makes it possible for all kinds of applications to adopt it. This means that if we have come across information that we have felt to be outside the main scope we have omitted it.

Security

The specification does not cover any aspects of security. The adaptor is free to use any of the public practices in this area.

Matching customers

An important issue, when integrating, is to identify and migrate records for the same identities. This is normally done in the beginning if two existing databases are integrated. This process could also be done each time one of the applications is creating a new record.

This specification does not address this issue in any way. It is up to each vendor to implement this in the way they feel appropriate.

Synchronisation

Introduction

Synchronisation is an essential part of the specification. Both customers and sellers need to be securely and uniquely identified by the applications. Therefore a method is established for this.

The method is based on a unique identification that is common to both applications. This identification is used for both customer and seller messages. After it's established it is used in all communications regarding the customer or seller.

The method is focused on synchronisation between two applications. In some cases it may be necessary to handle more applications, but that issue has not been addressed in this specification.

Identifier

The identifier is a 36 character long string. In the XML Schema it is called **<SieIdentification>**. When an application creates a new record it creates a new identifier. This application is responsible for the identifier to be unique in both applications.

The identifier could be created in any way as long it is unique. However it is recommended that GUID is used as the identifier, as this algorithm is platform independent and guarantees uniqueness. The identifier does not need to be exactly 36 characters long, but the application must be able to handle identifiers with this length.

Process

The exchange of information is based on different messages. Every message that is part of a synchronisation (SetCustomer and SetSeller) contains instructions to the receiver of the message on how to react to this message.

The **"Action"** attribute is the most important. It can hold any of five values:

Action values
Connect
Update
Reject
Disconnect
Delete

Connect

This action means that the producer of the message wants to establish a synchronisation with the consumer.

<SieIdentification> contains the identifier that the producer has created.
"Action" contains the **Connect** value.

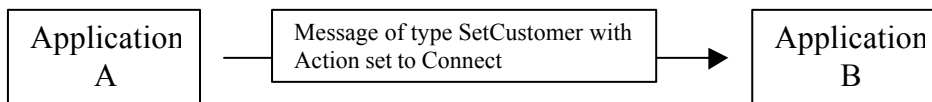
The reasons for this action could be several. The most obvious is that he has created a new customer or seller. It could also be that he has changed the status, for example from *Prospect* to *Customer*.

The message should contain all information about the object that the producer has knowledge about. Those elements that there is no knowledge about should be left out.

The receiver of the message must accept the identifier and save it together with the customer/seller. If he at this point adds additional information that is covered by this specification, i.e. customer number, he must send back a message of same type with that information. The action attribute should in this case be set to **Update**.

The consumer may want to match the incoming message with its own database to see if there is a similar record already. If there is a match it is OK to apply the incoming message to this record, as long as the identifier is used. If this occurs the receiver must send back an Update message with any additional information that he has knowledge about.

If the consumer gets information that he does not accept, he returns an message of the same type containing that information. The action attribute should be set to **Reject**. See more about this in the section *Reject*.



Example of Connect message

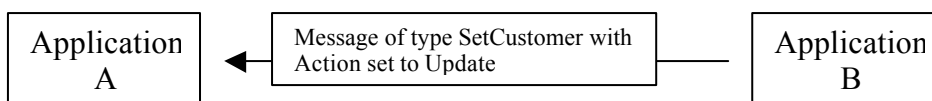
Update

This means that the producer of the message wants to update some information in the other application.

<SieIdentification> contains the identifier set in an earlier Connect message.
"Action" contains the **Update** value.

The message should only contain that information that has been changed. All other elements should be left out. If the producer is not able to recognize changes on field level its acceptable to send the entire object.

If the consumer gets information that he does not accept, he returns an message of the same type containing that information. The action attribute should be set to **Reject**. See more about this in the section *Reject*.



Example of Update message

Special note: The statement that only changed information should be sent in a Update message has one exception. This is the Address element that is part of the DeliveryAddress and InvoiceAddress elements. For this element all known information must be sent every time.

The reasons for this is that address information is handled very different in the applications on the market. Some applications support different fields for information in the element, some only support free text fields for address information. A synchronisation could be hard to establish if not all information is sent every time some address field is changed.

Reject

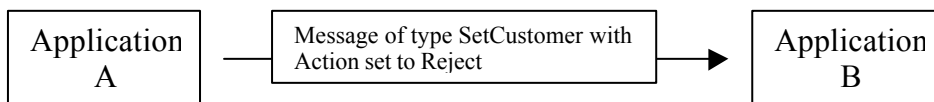
This means that the receiver of a message has not accepted all the information in a message. He therefore responds with a message of the same type as the original. This response message should only contain those elements that have been rejected.

<SieIdentification> contains the identifier set in an earlier Connect message.

"Action" contains the **Reject** value.

This action is only applicable for a Connect and Update message.

A message with this action would never return another **Reject** message.



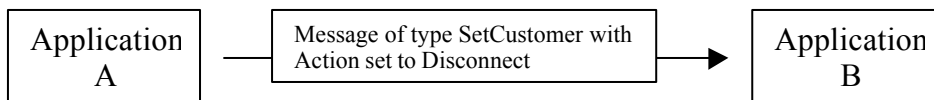
Example of Reject message

Disconnect

This means that the producer has disconnected the synchronisation for a specific record. The receiver should delete the identifier from his application.

<SieIdentification> contains the identifier set in an earlier Connect message.

"Action" contains the **Disconnect** value.



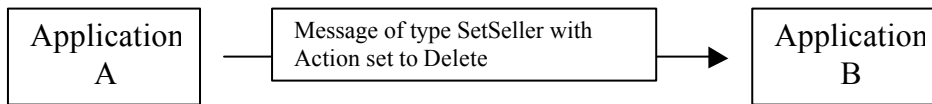
Example of Disconnect message

Delete

This means that the producer has deleted the record in his application. The receiver of the message reacts to this in an appropriate way. It is not required that the record is deleted, but it is required that the identifier is deleted from the record.

<SieIdentification> contains the identifier set in an earlier Connect message.

“**Action**” contains the **Delete** value.



Example of Delete message

Sequence numbers

The specification offers a basic method for ensuring that all messages are delivered, and in the correct order. At the top level there is an attribute called **<sequenceNumber>**. This could be used for this purpose.

The sequence number is an enumerator that is set by the producer. He must guarantee that it is in order and unique. It should start at one (1).

The receiver could with this sequence number read all the messages in exactly the right order and raise an error if messages are missing.

The specification requires that the sequence number is set by the producer but does not require the receiver to implement the checking process. Note that the sequence number is set on the envelope for the message. The message could then contain several different types of messages. All of them will be associated with the same sequence number.

Error handling and logging

If the consumer of a message identifies some general errors that not are related to the content of the message, the consumer should return a message of the type **Error**. This gives the producer a chance to react to the problem.

Is also recommended that some kind of logging functionality is supported in each application.

Certification

Introduction

The SIE Group provides a basic certification for the vendors who implements the specification. The purpose of this is to establish a basic level of quality assurance for the end users. The general rules for certification are published by SIE Group.

A vendor that implements the CRM/ERP specification should support it completely. This means that all the required messages on the level that certification is made on, and all required elements in each supported message, should be handled. The vendor should also make an effort to support as many of the optional elements.

Levels

Type

There are four different types of information: Customer, Credit, Sales history and Seller. The customer type is the base information.

Direction

It is possible to be certified on different levels of direction. The vendor could choose to implement a two-way integration or only a one-way.

Certification matrix

Type of information	Produce	Consume
Customer		
Credit		
Sales history		
Seller		

When a vendor applies for certification he defines which options he has built in support for in the application. The test message that he provides must include elements for all those types.

Examples

A typical ERP vendor could have certification like this:

Type of information	Produce	Consume
Customer	✓	✓
Credit	✓	
Sales history	✓	
Seller	✓	✓

A typical CRM vendor could be certified like this:

Type of information	Produce	Consume
Customer	✓	✓
Credit		✓
Sales history		✓
Seller	✓	✓

A vendor with no needs for a two-way integration could be certified like this:

Type of information	Produce	Consume
Customer		✓
Credit		✓
Sales history		✓
Seller	✓	✓

Restrictions

It is a requirement that the customer type is supported. It is not possible to implement support for any of the other types without also supporting this message.

If a vendor is certified to produce a message this means that he must support both the Set... and Get... message for that type. For example, if an application can produce the SetCustomer message it must also be able to read the GetCustomer message.

Regardless of certification the application has to support the Test message.

Certification process

The SIE Group will provide a special section on the Internet to accomplish certification. The address for this will be published as soon it's ready. This page will define all the rules for the certification and also guides you in the process.

An effective way to ensure that the messages is correct is to validate them in some XML parser that supports XML Schema.

XML Schema

All the messages are described in the same schema file, *SIE_CRMERP_v1.0*. From this it is possible to create all XML instance documents.

Overview

A message always contains the root element called *SIE_CRMERP*. This element could contain any of the nine different elements listed below.

Message	Contains the elements
SetCustomer	Identification Customer Action
GetCustomer	Identification
SetCredit	Identification Credit
GetCredit	Identification
SetSalesHistory	Identification SalesHistory
GetSalesHistory	Identification
SetSeller	Identification Seller Action
GetSeller	Identification
Test	sequenceNumber origSequenceNumber timeReceived softwareManufacturer

Additional info

The information in this specification is in the perspective that so many applications as possible should be able to implement it. Some information has therefore intentionally been left out. But for those applications that would like to build synchronisations with other applications that require extended information a special element has been added.

This element is called *Additional*. With this it's possible to add any valid XML information into a standard message without disturbing the normal procedure. It is the producer that is responsible for this element and that the receiver has the possibility to understand the information. This element is declared as a *ANY* element in the schema. Go to the W3C documentation of XML Schema for more information of this type.

Complete schema reference

[Schema reference in HTML format](#)

[Schema reference in Word format](#)

XML Schema documents

[XML Schema in XSD format](#)

[XML Schema in XML format](#)

Other

[Example message](#)

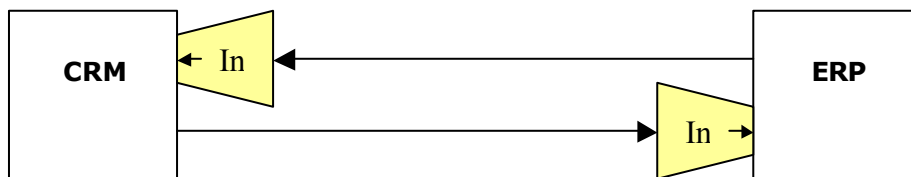
Appendices

A. Delivery of messages with the file system

The CRM/ERP specification is completely transport independent. It does not require any special way to deliver the messages between two applications. This appendix is just included to provide a basic platform for any transport mechanism and specific file based implementation. Eventually the implementation will be done with more advanced technique.

Overview

Each application engaged in the synchronisation needs to manage two separate folders. One for the incoming messages and one for the outgoing messages. The sender of a message creates the message and saves in the other applications IN folder. The receiver of the message is responsible for reading all messages that arrives to the IN folder.



Folders

Each application publish a specific folder for incoming messages. The folder could be placed anywhere but it's recommended that it's placed in the same directory as the application.

Name rules

- The In folder is named *SIE_In*.
- The file is named with the prefix SIE followed by the sequence number assigned inside the message. *Ex: SIE123456789.xml*.
- The extension of the file should be *.xml*.