itineris
excellence in utilities

your way to perform

Integration Design Document (IDD)

UMAX® - Cityworks® Storeroom

12/10/2015

0. General

0.1. Table of contents

0.2. Version history

Date	Version	Author	Modification description
05/22/2015	001	Nickolas Heirbaut	Moved Cityworks Storeroom interfaces to separate Integration Design Document
05/22/2015	002	Nickolas Heirbaut	Added reference to mapping document

0.3. Related documents

Service nr.	Name	Version	Description
IMD001	IMD001 - Integration mapping - UMAX-Cityworks - Assets	001	Mapping document (assets)

0.4. Input

Nr.	Name	Version	Description
001	Appendix D Baltimore Interface Descriptions.pdf	001	RFP - Appendix D
002	UMAX-Cityworks-ESRI Integration_ITINERIS RESPONSE 7-08-14.docx	001	BAFO - Scope document
003	Technical Workshop MM 20150114 Cityworks - FINAL.pdf	001	Analysis workshop - Meeting notes
004	CIS Workshop Action Items.xlsx	001	Workshop Action Items
006	20150409 WS Meeting Minutes - Interfaces - Cityworks Storeroom.pdf	001	Design workshop - Meeting notes

0.5. Appendix

Nr.	Name	Version	Description

0.6. Open topics

No.	Open topic	Status

0.7. Approval and signature

Function	Name	Date	Signature
Business Key-User			
Business Project Manager			
Itineris Project Manager			
Itineris Solution Architect			

1. Introduction

1.1. Management summary

This document provides the design for the integration between UMAX® and Cityworks®. It describes the high-level business process flow, as well as the data flows and services that are herein involved.

The integration between UMAX and Cityworks enables the following data exchanges:

- Limited synchronization of asset from Cityworks Storeroom (inventory) to UMAX (e.g. newly purchased meters or ERTs)

Cityworks®

- Cityworks Storeroom is the inventory management application used to store assets such as meters, registers, ERT's, etc.

UMAX®

- UMAX uses the synchronized assets to determine which asset should be issued by the meter shop. It's also being used to perform validations on the serial number of a meter for incoming service order results (e.g. meter install, meter replacement, etc.).

1.2. Audience

This document serves as a reference for project managers, analysts, consultants, IT staff members, architects, integration specialists, designers and developers to provide insight in the design of the solution.

1.3. Preconditions and assumptions

No.	Description
001	Only assets relevant for UMAX processes are being synchronized
002	A file can contain both new & updated assets.
003	Cityworks storeroom will keep track of serial numbers per asset

1.4. Requirements

The requirements extracted from the RFP and workshops.

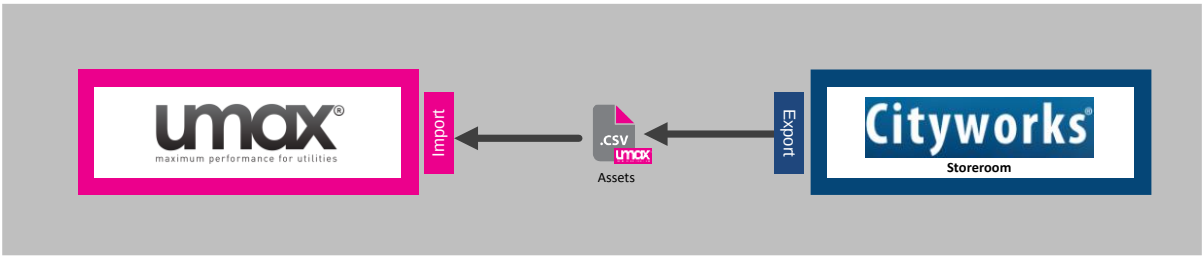
No.	Origin	Requirement
1	RFP	None
1.1	Analysis	UMAX must be able to import meter and ERT information

2. Architecture

2.1. High level architecture

The picture shown below provides a high level overview of the systems involved and the way they communicate with each other. Both systems exchange data file-based by means of an XML or CSV file.

For data exchanged from Cityworks Storeroom to UMAX, a CSV file is created and stored in a designated folder by Cityworks. The file is fetched by UMAX in which it will be read and processed.



2.2. Applications

The applications shown in the table below are involved in this integration.

Application	Short description
UMAX	Customer Interaction System / Billing system
Cityworks Storeroom	Inventory solution

2.3. Interfaces

Interface ID	Interface name	Remarks
F-CWS-UMX-01	Import assets	Creation or update of a meter or ERT in UMAX

3. Functional processes

In this section the high level process is illustrated. Subsequently the different service groups are viewed in relation with the process and the different services within these groups are described.

3.1. High level processes

Phase	Description
1. Assets synchronization	The process to retrieve meters & ERTs from Cityworks storeroom

The terminology used within the different applications can be mapped as follows:

UMAX	Cityworks
Asset	Material
Installation object	Material

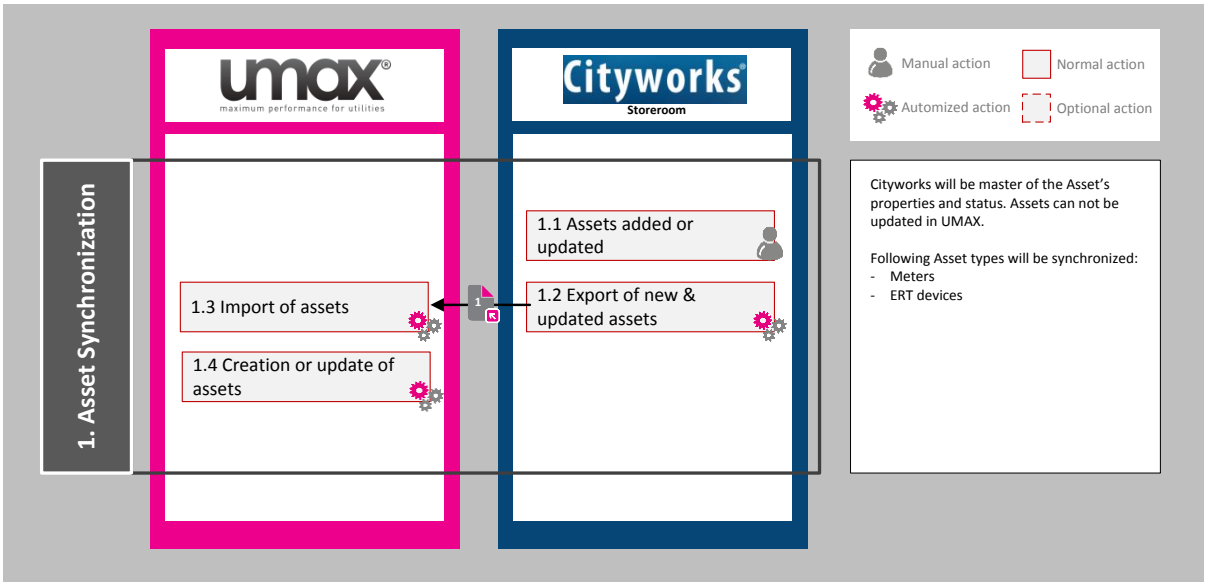
3.2. Asset synchronization

The main purpose of this process is to synchronize assets between UMAX and Cityworks Storeroom. Assets are only synchronized in one-way, from Cityworks Storeroom to UMAX. Updates to assets in UMAX are not communicated to Cityworks Storeroom.

Cityworks Storeroom is considered to be the master of the assets.

3.2.1. Service process flow

The synchronization process when an asset in Cityworks is created or updated.



The following steps can be distinguished:

Step	Description
AS-1.1	**Assets added or updated** New assets are uploaded in Cityworks Storeroom or existing assets are being updated (e.g. correction of the serial number). This can be either manual or via bulk import.
AS-1.2	**Export of new & updated assets** Cityworks exports any new or updated asset using a nightly batch job
AS-1.3	**Import of assets** UMAX imports the list with new and updated assets using a nightly batch job
AS-1.4	**Creation or update of asset** The asset is created or updated in the UMAX inventory

3.2.2. Services involved

ID	Interface name	Steps	Description
F-CWS-UMX-01	Create and update UMAX asset	AS-1.2 AS-1.3	Synchronization of assets

3.2.3. F-CWS-UMX-01 - Create and update UMAX asset

Cityworks Storeroom exports all newly added or changed assets periodically using a batch job since the last export date. Updates to existing and already synchronized assets are managed in Cityworks Storeroom, not UMAX. New & updated assets will be included in one single file.

3.2.3.1. Communication paradigm

Paradigm item	Description
Integration approach	File transfer via file share
Exchange pattern	Export / import
Sequencing	Asynchronous
Message content	New and updated assets

3.2.3.2. Data sources and targets

Source and target systems for all services involved are:

Type	Application	Role
Source	Cityworks Storeroom	Data provider
Target	UMAX	Data requestor

3.2.3.3. Interface conditions

3.2.3.3.1. Cityworks Storeroom to file share

Use case item	Description
Actor/initiator	Cityworks Storeroom
Trigger	Scheduled batch
Preconditions	Assets exists for which the creation or changed date is more

	recent than the last export date
Base flow of events	An automated process runs periodically (e.g. every night) to export all meters & ERTs in the inventory that have been changed or created since the last export date. A CSV file is created and stored on a designated file share.
Alternative flows	*n/a*
Post conditions	*n/a*
Protocol	SMB

3.2.3.3.2. File share to Cityworks

Use case item	Description
Actor/initiator	UMAX
Trigger	Scheduled batch (polling)
Preconditions	File exists on file share
Base flow of events	UMAX monitors a file share periodically to detect the presence of a new CSV file that contains the meters & ERTs. The CSV file is imported into UMAX which triggers the creation of an asset or the update of an existing asset. After successful import, the file is moved to an archiving folder. When import fails, the file is moved to an error folder.
Alternative flows	*n/a*
Post conditions	No files exists on the file share; All assets in the CSV file are created or updated in UMAX
Protocol	SMB

3.2.4. Scenario's

The scenario's supported by the service.

Scenario	Description	Supported
001	New asset created in Cityworks Storeroom	Yes
002	Existing asset updated in Cityworks Storeroom	Yes

4. Non-functional requirements

4.1. Requirements per service

Characteristic	F-CWS-UMX-01			
Interface ID	F-CWS-UMX-01			
Interface name	Create and update UMAX asset			
Integration approach	File transfer			
Exchange pattern	Export / import			
Sequencing	Asynchronous			
Source / sender	Cityworks Storeroom			
Target / receiver	UMAX			
Message size	Max. 1.000 records or 100 MB			
Average message volume (messages/interval)	1 / day			
Expected growth (2 years)	No growth expected			
Peak message volume (messages/interval)	1 / day			
Peak hour	Outside business hours			
Expected end-to-end answering time	N/A			
Maximum end-to-end answering time	N/A			
Possibility to re-send (idempotence)	Yes			
Multi-threaded	No			
Maximum number combined operations	1			
Message buffering	No			
Protocol	SMB/TCP/IP			
Security	NTLM			
Encoding	UTF-8 (incl. BOM)			

4.2. Operational behavior

Topic	Description
Load balancing	Multiple import files can be processed in parallel
Fail-over	Supported in UMAX in a multi-AOS environment. When an AOS is unavailable, another AOS will be assigned to process the service
Scalability	Adding extra AOS services
Implementation technology	File transfer, import/export
Monitoring schedules	Alerting mechanism in UMAX in place for detecting failures during import or export. An e-mail notification can be sent to functional maintenance.
Backend start-up procedures	n/a
Backend shutdown procedures	n/a
Audit requirements	Currently no specific requirements with regards to audit.
Backup and restore	Back-up schedule is expected to be in place for data in UMAX & Cityworks Back-up schedule is expected to be in place for file share to protect files exported by one system but not yet imported by the other.

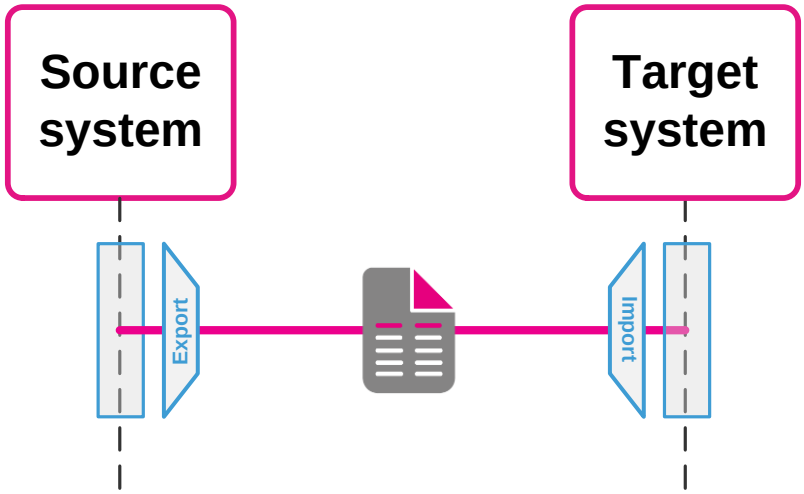
5. General agreements interfaces

5.1. File-based export / import

5.1.1. Communication paradigm

Source system produces a file containing information that the target system needs to consume. The files are produced and consumed at regular intervals according to the nature of the business. The format of the file is determined upfront.

The export or import of a file can be either triggered manually via user interaction, or via an automated batch process.



5.1.2. Logging

When an import or export operation is executed as a batch process, any message that is encountered during processing is stored in the batch log file. Alerts can be sent when the batch job succeeds, fails or finishes. Alerts are visible in the AX client, or they can be sent to an e-mail address.

When an import or export operation is executed as a user action in the AX client, any message that is encountered during processing is shown in the Info Log.

5.1.3. Security

Security is expected to be enforced by the operating system that hosts the file locations which are used for import & export. File locations are expected to be Windows file shares which are accessible by a Universal Naming Convention (UNC) path (\\servername\path) or local file system (LFS) path (e.g. D:\temp\input) using the Microsoft SMB protocol and the NTLM authentication protocol.

Files are not protected; file content is not encrypted.

5.1.4. Encoding

File encoding for CSV is UTF-8, including BOM

5.1.5. Message structure

Not generic message structure

5.1.6. Message format

Comma separated values format (MS-DOS):

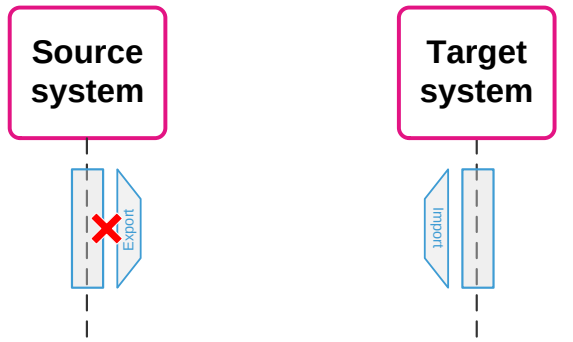
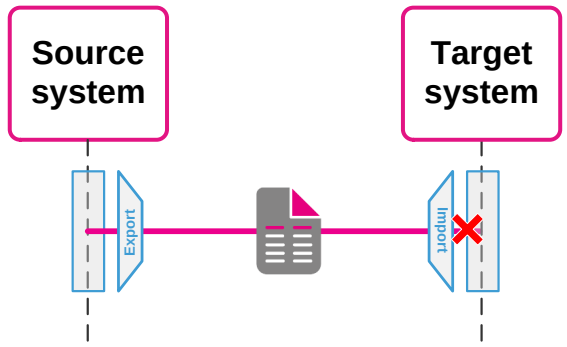
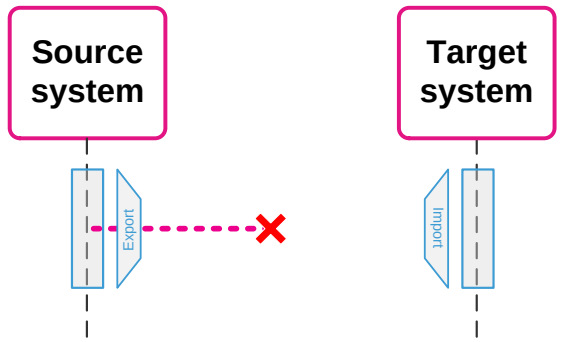
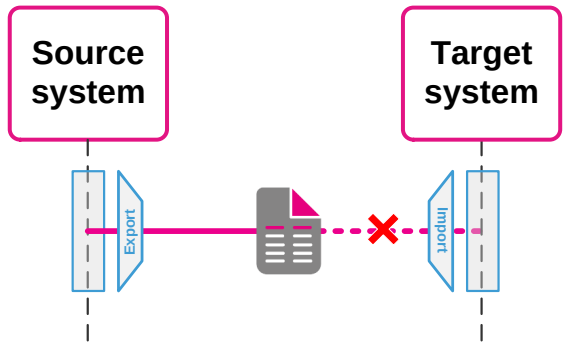
- Each record will be contained on one line of the file, terminated by a **carriage return and line feed pair (CR/LF; 0D0A),** with each field separated by a **semicolon**.
- The first record in the file will contain a **column name header** in each of the fields.
- There is **no semicolon following the final field** in the record.
- In case field values have embedded commas, embedded double-quote characters, intentional leading/trailing space characters, or other reserved characters, fields will always be enclosed within **double-quote characters**, whether necessary or not.
- Leading and trailing spaces or tabs adjacent to semicolon (not within double-quotes) will be trimmed.
- Any embedded double-quote characters must be represented by a pair of double-quote characters.

5.1.7. Data types

Data type	Description	Example
String	Text value	Single Meter with ERT
Number	Numeric	5488
Date	Date	2014-04-30

5.1.8. Error handling

In the message exchange process using an export/import paradigm, the source and target system work independently from each other. Errors are handled by the system that is performing the processing, which can be either import or export.

Errors handled by source system	Errors handled by target system
Error in source system during export	*Error in target system during import*
	
Failure to write to file share	*Failure to read from file share*
	

5.1.8.1. Technical errors

5.1.8.1.1. Guaranteed delivery

When a failure occurs (e.g. file system unavailable, database unavailable, functional error, etc.), the system processing the file is responsible for recovering from this situation. The system might try to attempt the processing for a predefined number of times in a given interval.

The source system does not inform the target system of any failure, this process in handled manually. Operators are responsible for manually recovering from the failure.

When the retry threshold is reached during export, the source system will notify operators about the failure by raising an alert (e.g. e-mail notification).

When the retry threshold is reached during import, the target system will move the file from the input folder to a designated error folder and notify operators about the failure by raising an alert (e.g. e-mail notification).

As long as the export fails and no file is present at the designated input folder, the target system assumes that no files are available for import.

5.1.8.1.2. Ordered delivery

The source system ensures that messages are exported in ordered sequence. When a file fails to be exported, no other files are being exported by the source system to the designated file share until the failed export is resolved. This allows ordered delivery of the messages.

The target system ensures that messages are imported in ordered sequence. The date/timestamp in the filename is used for ordering. The file with the oldest date/timestamp in the filename is imported first. When the import of a file fails, subsequent files are not being imported until the failed import is resolved.

5.1.8.1.3. Idempotence

With idempotence it is meant that the target system should be able to safely import the same message several times without changing the result beyond the initial import.

Each file gets a unique filename. This unique filename is used by the target system to detect if the received file is processed already before. If this is the case the target system ignores the file without processing it again.

This way the message is delivered guaranteed and will be processed only once.

On record level, idempotence is not guaranteed. The source system takes care for preventing that duplicate records occur in one single file, or over multiple files.

5.1.8.2. Functional errors

For functional errors a specific action will be required each time, depending on the supported functionality. In most cased the functional errors will lead to a manual action or work order that has to be executed. To make sure that this action is executed, the required person(s) has/have to be notified.

The origin of a functional error can be one of the following:

- A check or validation has a negative result
- When executing a functional process in the target system an exception is generated

5.1.9. Naming conventions

5.1.9.1. Naming system

In the naming system for a service the following construction will be used:

`<Type>-<Source>-<Target>-<Sequence number>`

Integration type	The type of integration F: File-based integration
Source / target	The source that provides the message or the target that consumes the message CWS: Cityworks Storeroom UMX: UMAX

Sequence number	Sequence number of 2 digits (01, 02, …)

5.1.9.2. File name

Structure:

```
<Type>-<Source>-<Target>-<Sequence number>_<YYYYMMDD>_<HHMMSS>_<GUID>.<FileExt>
```

Example:

```
S-CWS-UMX-01-20151225_144563_6b3f0727-7141-42f3-936a-5e268ddca4fa.csv
```