



PROJECT DOCUMENTATION

Project Name:	Synapse mORMot Framework
Document Name:	Software Requirements Specifications
Document Revision:	1.17
Date:	September 9, 2012
Project Manager:	Arnaud Bouchez

Document License

THE ATTACHED DOCUMENTS DESCRIBE INFORMATION RELEASED BY SYNOPSE INFORMATIQUE UNDER A GPL 3.0 LICENSE.

Synapse SQLite3/mORMot Framework Documentation.

Copyright (C) 2008-2012 Arnaud Bouchez.

Synapse Informatique - <http://synapse.info..>

This document is free document; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

The *Synapse mORMot Framework Documentation* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this documentation. If not, see <http://www.gnu.org/licenses..>

Trademark Notice

Rather than indicating every occurrence of a trademarked name as such, this document uses the names only in an editorial fashion and to the benefit of the trademark owner with no intention of infringement of the trademark.

Prepared by:	Title:	Signature:	Date
Arnaud Bouchez	Project Manager		

Document Purpose

The *Software Requirements Specifications* document purpose is to interpret design inputs and specify software design features for the *Synapse mORMot Framework* project.

The current revision of this document is 1.17.

Related Documents

Name	Description	Rev.	Date
DI	Design Input Product Specifications	1.17	September 9, 2012
SAD	Software Architecture Design	1.17	September 9, 2012

Table of Contents

Introduction

Documentation overview	5
Purpose	5
Risk Assessment	6
Responsibilities	6
Design Input Reference Table	8

1. Client Server JSON framework

1.1. Design Input 2.1.1 (Initial release)	9
1.2. Design Input 2.1.1.1 (Initial release)	9
1.3. Design Input 2.1.1.2 (Initial release)	10
1.4. Design Input 2.1.2 (Initial release)	10
1.5. Design Input 2.1.3 (Initial release)	11

2. SQLite3 engine

2.1. Design Input 2.2.1 (Initial release)	12
2.2. Design Input 2.2.2 (Initial release)	12
2.3. Design Input 2.2.3 (Initial release)	12

3. User interface

3.1. Design Input 2.3 (Initial release)	14
3.2. Design Input 2.3.1 (Initial release)	15
3.3. Design Input 2.3.2 (Initial release)	16

Pictures Reference Table

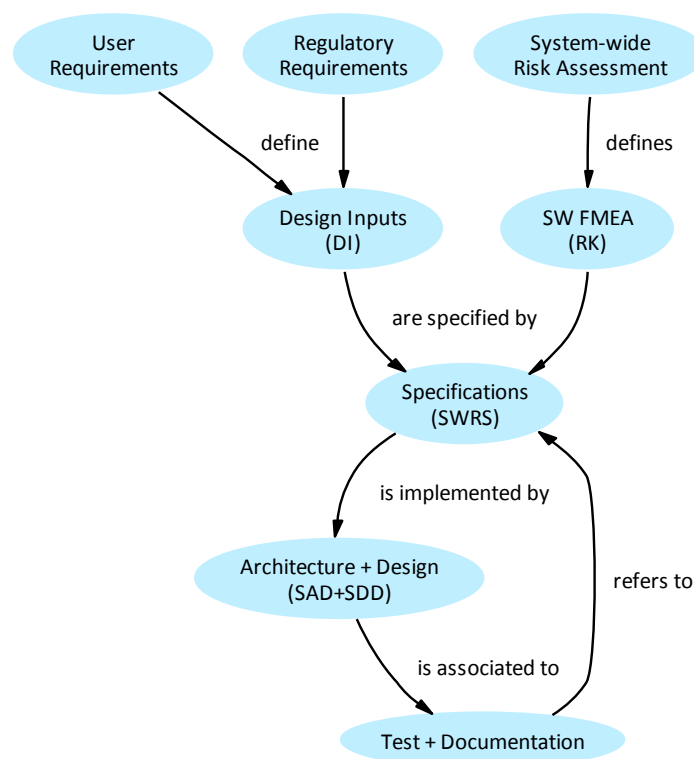
The following table is a quick-reference guide to all the Pictures referenced in this *Software Requirements Specifications* (SWRS) document.

Pictures	Page
Design Inputs, FMEA and Risk Specifications	5
User Interface generated using TMS components	14
User Interface generated using VCL components	15

Introduction

Documentation overview

The whole Software documentation process follows the typical steps of this diagram:



Design Inputs, FMEA and Risk Specifications

Purpose

This *Software Requirements Specifications (SWRS)* document applies to the first public release of the *Synopse mORMot Framework*.

It describes the software implementation of each design input as specified by the *Design Input Product Specifications (DI)* document.

The sections of this document follow the *Design Input Product Specifications (DI)* document divisions:

- Client Server JSON framework (page 9)
- SQLite3 engine (page 12)
- User interface (page 14)

For each Design Input item, the corresponding justification is specified, between parenthesis (SCR #65, e.g.).

Every *Software Requirements Specifications* (SWRS) document item is named about the corresponding *Design Input Product Specifications* (DI) document item, or, in case the initial *Design Input* is too large and must be divided into some SWRS more precise items, an unique name is proposed.

Risk Assessment

The Risk assessment indicated below was evaluated as a team work, based on the software solution proposed.

In the following *Software Requirements Specifications* (SWRS) document, a numerical Risk Assessment is given for every Design Input item, according to the *Risk Assessment Scale* table below.

A summary explanation is indicated, together with the names of those who made each evaluation.

Risk Assessment Scale

Severity: identify the severity of incorrect implementation

3 - High	Potentially affects a result or safety
2 - Med	Potentially effects one or multiple features for intended operation
1 - Low	Cosmetic or no effect to intended operation

Probability: identify the probability of incorrect or incomplete implementation

3 - High	No documentation and not familiar with the code area
2 - Med	Documentation but not familiar/familiar but no documentation
1 - Low	Documentation and familiar with the code

Occurrence: identify the reproducibility of the defect before correction

3 - High	Reproducible: failure inevitable or repeated (>25% failure rate)
2 - Med	Intermittent or recurring: occasional failures (5-25% failure rate)
1 - Low	One time: relatively few or remote likelihood of failure (<5% failure rate)

Responsibilities

- Synopse will try to correct any identified issue;
- The Open Source community will create tickets in a public Tracker web site located at <http://synopse.info/fossil..> ;
- Synopse work on the framework is distributed without any warranty, according to the chosen license terms;

- This documentation is released under the GPL (GNU General Public License) terms, without any warranty of any kind.

Design Input Reference Table

The following table is a quick-reference guide to all the software *Design Input Product Specifications* (DI) document items and their corresponding *Software Requirements Specifications* (SWRS) document items.

DI #	SWRS #	Description	Page
2.1.1	DI-2.1.1	The framework shall be Client-Server oriented	9
2.1.1.1	DI-2.1.1.1	A RESTful mechanism shall be implemented	9
2.1.1.2	DI-2.1.1.2.1 DI-2.1.1.2.2 DI-2.1.1.2.3 DI-2.1.1.2.4	Communication should be available directly in the same process memory, or remotely using Named Pipes, Windows messages or HTTP/1.1 protocols	10
2.1.2	DI-2.1.2	UTF-8 JSON format shall be used to communicate	10
2.1.3	DI-2.1.3	The framework shall use an innovative ORM (Object-relational mapping) approach, based on classes RTTI (Runtime Type Information)	11
2.2.1	DI-2.2.1	The <i>SQLite3</i> engine shall be embedded to the framework	12
2.2.2	DI-2.2.2	The framework libraries, including all its <i>SQLite3</i> related features, shall be tested using Unitary testing	12
2.2.3	DI-2.2.3	The framework shall be able to access any external database, via OleDB, ODBC or direct access for Oracle (OCI) or <i>SQLite3</i> (for external database files)	12
2.3	DI-2.3	User Interface and Report generation should be integrated	14
2.3.1	DI-2.3.1.1 DI-2.3.1.2 DI-2.3.1.3	An User Interface, with buttons and toolbars shall be easily being created from the code, with no RAD needed, using RTTI and data auto-description	15
2.3.2	DI-2.3.2	A reporting feature, with full preview and export as PDF or TXT files, shall be integrated	16

1. Client Server JSON framework

1.1. Design Input 2.1.1 (Initial release)

The framework shall be Client-Server oriented

Client-Server model of computing is a distributed application structure that partitions tasks or workloads between service providers, called servers, and service requesters, called clients.

Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server machine is a host that is running one or more server programs which share its resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await (listen for) incoming requests.

The *Synapse mORMot Framework* shall implement such a Client-Server model by a set of dedicated classes, over various communication protocols, but in a unified way. Application shall easily change the protocol used, just by adjusting the class type used in the client code. By design, the only requirement is that protocols and associated parameters are expected to match between the Client and the Server.

Severity: 1, Probability: 1, Occurrence: 3
RISK Initial release
Risk evaluation team: Arnaud Bouchez

1.2. Design Input 2.1.1.1 (Initial release)

A RESTful mechanism shall be implemented

REST-style architectures consist of clients and servers, as was stated in *SWRS # DI-2.1.1* (page 9). Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of "representations" of "resources". A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.

In the *Synapse mORMot Framework*, so called "resources" are individual records of the underlying database, or list of individual fields values extracted from these databases, by a SQL-like query statement.

Severity: 1, Probability: 1, Occurrence: 3
RISK Initial release
Risk evaluation team: Arnaud Bouchez

1.3. Design Input 2.1.1.2 (Initial release)

Communication should be available directly in the same process memory, or remotely using Named Pipes, Windows messages or HTTP/1.1 protocols

In computing and telecommunications, a protocol or communications protocol is a formal description of message formats and the rules for exchanging those messages.

The *Synopse mORMot Framework* shall support the following protocols for remote access, according to the Client-Server architecture defined in *SWRS # DI-2.1.1* (page 9):

- Direct in-process communication;
- Using GDI messages;
- Using Named pipe;
- Using HTTP/1.1 over TCP/IP.

This Design Input shall be traced to the following SWRS items:

DI-2.1.1.2.1 Client-Server Direct communication shall be available inside the same process.

DI-2.1.1.2.2 Client-Server Named Pipe communication shall be made available by some dedicated classes.

DI-2.1.1.2.3 Client-Server Windows GDI Messages communication shall be made available by some dedicated classes.

DI-2.1.1.2.4 Client-Server HTTP/1.1 over TCP/IP protocol communication shall be made available by some dedicated classes, and ready to be accessed from outside any Delphi Client (e.g. the implement should be AJAX ready).

Severity: 1, Probability: 1, Occurrence: 3

RISK Initial release

Risk evaluation team: Arnaud Bouchez

1.4. Design Input 2.1.2 (Initial release)

UTF-8 JSON format shall be used to communicate

JSON, as defined in the *Software Architecture Design (SAD)* document, is used in the *Synopse mORMot Framework* for all Client-Server communication. JSON (an acronym for JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language.

JSON shall be used in the framework for returning individual database record content, in a disposition which could make it compatible with direct JavaScript interpretation (i.e. easily creating JavaScript object from JSON content, in order to facilitate AJAX application development). From the Client to the Server, record content is also JSON-encoded, in order to be easily interpreted by the Server, which will convert the supplied field values into proper SQL content, ready to be inserted to the underlying database.

JSON should be used also within the transmission of request rows of data. It therefore provide an easy way of data formatting between the Client and the Server.

The *Synopse mORMot Framework* shall use UTF-8 encoding for the character transmission inside its JSON content. UTF-8 (8-bit Unicode Transformation Format) is a variable-length character encoding for

Unicode. UTF-8 encodes each character (code point) in 1 to 4 octets (8-bit bytes). The first 128 characters of the Unicode character set (which correspond directly to the ASCII) use a single octet with the same binary value as in ASCII. Therefore, UTF-8 can encode any Unicode character, avoiding the need to figure out and set a "code page" or otherwise indicate what character set is in use, and allowing output in multiple languages at the same time. For many languages there has been more than one single-byte encoding in usage, so even knowing the language was insufficient information to display it correctly.

Severity: 1, Probability: 1, Occurrence: 3
RISK Initial release
Risk evaluation team: Arnaud Bouchez

1.5. Design Input 2.1.3 (Initial release)

The framework shall use an innovative ORM (Object-relational mapping) approach, based on classes RTTI (Runtime Type Information)

ORM, as defined in the *Software Architecture Design (SAD)* document, is used in the *Synapse mORMot Framework* for accessing data record fields directly from Delphi Code.

Object-relational mapping (ORM, O/RM, and O/R mapping) is a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the Delphi programming language.

The published properties of classes inheriting from a new generic type named `TSQLRecord` are used to define the field properties of the data. Accessing database records (for reading or update) shall be made by using these classes properties, and some dedicated Client-side methods.

Severity: 1, Probability: 1, Occurrence: 3
RISK Initial release
Risk evaluation team: Arnaud Bouchez

2. SQLite3 engine

2.1. Design Input 2.2.1 (Initial release)

The *SQLite3* engine shall be embedded to the framework

The *SQLite3* database engine is used in the *Synopse mORMot Framework* as its kernel database engine. *SQLite3* is an ACID-compliant embedded relational database management system contained in a C programming library.

This library shall be linked statically to the *Synopse mORMot Framework*, and interact directly from the Delphi application process.

The *Synopse mORMot Framework* shall enhance the standard *SQLite3* database engine by introducing some new features stated in the *Software Architecture Design (SAD)* document, related to the Client-Server purpose or the framework - see *SWRS # DI-2.1.1* (page 9).

Severity: 1, Probability: 1, Occurrence: 3
RISK Initial release
Risk evaluation team: Arnaud Bouchez

2.2. Design Input 2.2.2 (Initial release)

The framework libraries, including all its *SQLite3* related features, shall be tested using Unitary testing

The *Synopse mORMot Framework* shall use all integrated Unitary testing features provided by a common testing framework integrated to all Synopse products. This testing shall be defined by classes, in which individual published methods define the actual testing of most framework features.

All testing shall be run at once, for example before any software release, or after any modification to the framework code, in order to avoid most regression bug.

Severity: 1, Probability: 1, Occurrence: 3
RISK Initial release
Risk evaluation team: Arnaud Bouchez

2.3. Design Input 2.2.3 (Initial release)

The framework shall be able to access any external database, via OleDB, ODBC or direct access for Oracle (OCI) or *SQLite3* (for external database files)

The following external database providers shall be made available to the framework ORM:

- Any *OleDB* provider;
- Any *ODBC* provider;
- *Oracle* database, via direct OCI client access;
- *SQLite3* database engine.

A dedicated set of classes shall implement access, together with some advanced syntactic sugar, like fast late-binding, or advanced ORM mechanism, like Virtual Tables.

Severity: 1, Probability: 1, Occurrence: 3

RISK Initial release

Risk evaluation team: Arnaud Bouchez

3. User interface

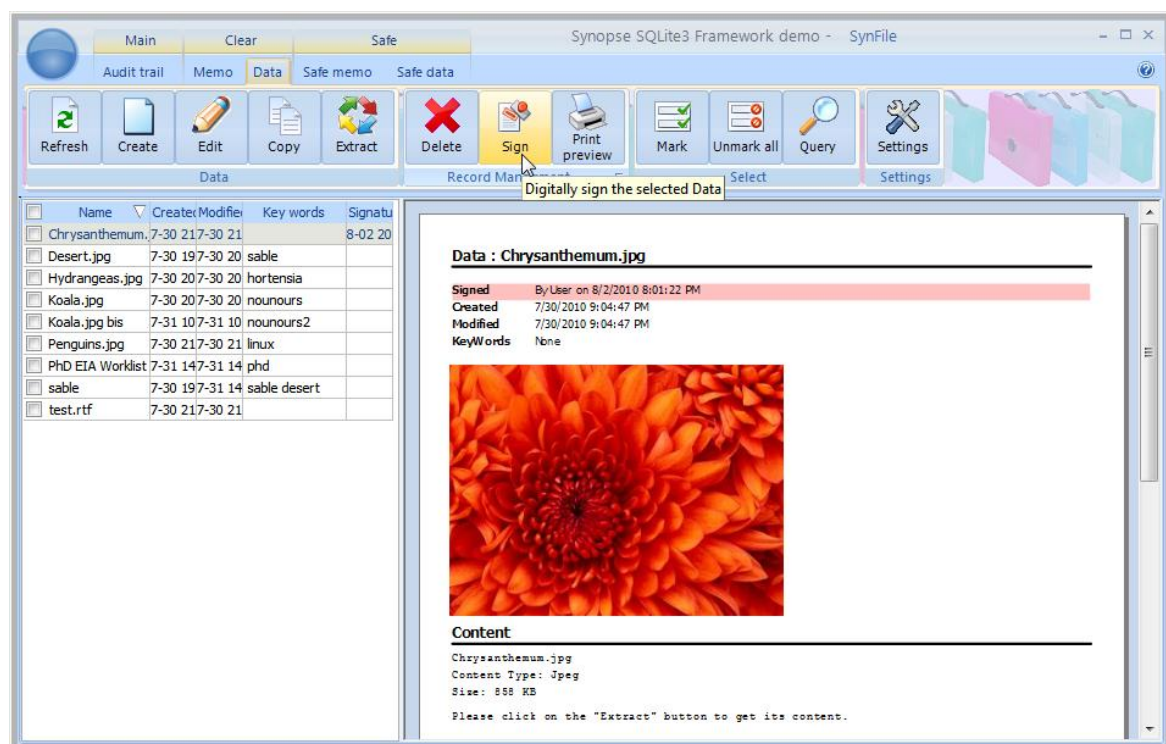
3.1. Design Input 2.3 (Initial release)

User Interface and Report generation should be integrated

The *Synapse mORMot Framework* shall provide User Interface and Report generation from code.

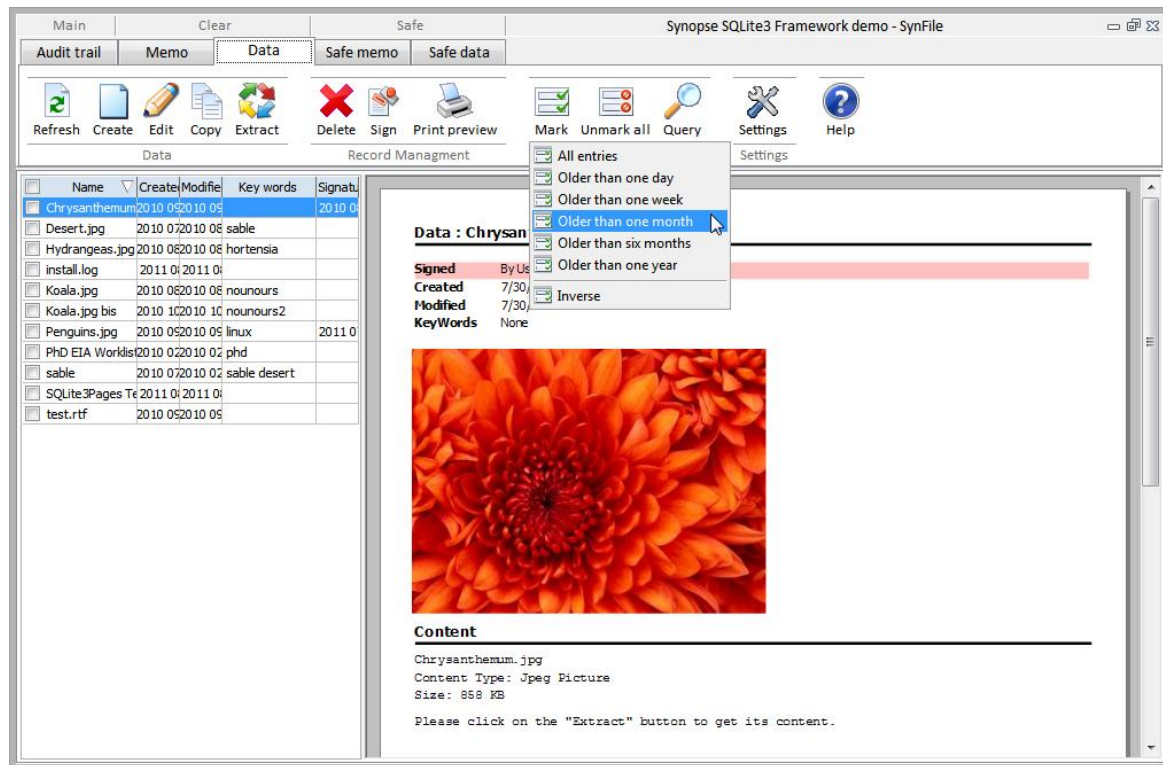
Such a ribbon-oriented interface shall be made available, in a per-table approach, and associated reports.

Here is a sample of screen content, using proprietary TMS components:



User Interface generated using TMS components

And here is the same application compiled using only VCL components, available from Delphi 6 up to XE2:



User Interface generated using VCL components

Severity: 1, Probability: 1, Occurrence: 3
RISK Initial release
Risk evaluation team: Arnaud Bouchez

3.2. Design Input 2.3.1 (Initial release)

An User Interface, with buttons and toolbars shall be easily being created from the code, with no RAD needed, using RTTI and data auto-description

The *Synopse mORMot Framework* shall provide some User-Interface dedicated units, allowing database grid display, on screen tool-bar creation (by an internal system of actions using Delphi RTTI - see the corresponding paragraph in the *Software Architecture Design* (SAD) document), and integrated reporting - see *SWRS # DI-2.3.2* (page 16).

No RAD approach is to be provided: the Client application User Interface will be designed not by putting components on the IDE screen, but directly from code.

This Design Input shall be traced to the following SWRS items:

- DI-2.3.1.1** A Database Grid shall be made available to provide data browsing in the Client Application - it shall handle easy browsing, by column resizing and sorting, on the fly customization of the cell content.
- DI-2.3.1.2** Toolbars shall be able to be created from code, using RTTI and enumerations types for defining the action.

DI-2.3.1.3 Internationalization (i18n) of the whole User Interface shall be made available by defined some external text files: Delphi resourcestring shall be translatable on the fly, custom window dialogs automatically translated before their display, and User Interface generated from RTTI should be included in this i18n mechanism.

Severity: 1, Probability: 1, Occurrence: 3

RISK Initial release

Risk evaluation team: Arnaud Bouchez

3.3. Design Input 2.3.2 (Initial release)

A reporting feature, with full preview and export as PDF or TXT files, shall be integrated

The *Synapse mORMot Framework* shall provide a reporting feature, which could be used stand-alone, or linked to its database mechanism. Reports shall not be created using a RAD approach (e.g. defining bands and fields with the mouse on the IDE), but shall be defined from code, by using some dedicated methods, adding text, tables or pictures to the report. Therefore, any kind of report shall be generated.

This reports shall be previewed on screen, and exported as PDF or TXT on request.

Severity: 1, Probability: 1, Occurrence: 3

RISK Initial release

Risk evaluation team: Arnaud Bouchez