



Automating Production of Cross Media Content for Multi-channel Distribution

www.AXMEDIS.org

DE5.1.3.1

AXMEDIS Framework Guidelines

Version: 2.2

Date: 10/08/2006

Responsible: L. Ortimini, S. Chellini of DSI (revised and approved by coordinator)

Project Number: IST-2-511299

Project Title: AXMEDIS

Deliverable Type: public

Visible to User Groups: Yes

Visible to Affiliated: Yes

Visible to Public: Yes

Deliverable Number: DE5.1.3.1

Contractual Date of Delivery: M22

Actual Date of Delivery: 10/08/2006

Work-Package contributing to the Deliverable: WP5

Task contributing to the Deliverable: all tasks of WP5

Nature of the Deliverable: report

Author(s): DSI, FHGIGD, EXITECH, FUPF, and almost all partners

Abstract:

This document contains the collection of all major guidelines of AXMEDIS and in some cases refers to other AXMEDIS document. Relevant guidelines are those on: components validation, AXMEDIS plug in production, regression testing, performances and optimisation, demonstrator video production, etc.

Keyword List: AXMEDIS, framework, validation, testing, performance, optimisation, etc.

AXMEDIS Copyright Notice

The following terms (including future possible amendments) set out the rights and obligations licensee will be requested to accept on entering into possession of any official AXMEDIS document either by downloading it from the web site or by any other means.

Any relevant AXMEDIS document includes this license. PLEASE READ THE FOLLOWING TERMS CAREFULLY AS THEY HAVE TO BE ACCEPTED PRIOR TO READING/USE OF THE DOCUMENT.

1. DEFINITIONS

- i. "**Acceptance Date**" is the date on which these terms and conditions for entering into possession of the document have been accepted.
- ii. "**Copyright**" stands for any content, document or portion of it that is covered by the copyright disclaimer in a Document.
- iii. "**Licensor**" is AXMEDIS Consortium as a de-facto consortium of the EC project and any of its derivations in terms of companies and/or associations, see www.axmedis.org
- iv. "**Document**" means the information contained in any electronic file, which has been published by the Licensor's as AXMEDIS official document and listed in the web site mentioned above or available by any other means.
- v. "**Works**" means any works created by the licensee, which reproduce a Document or any of its part.

2. LICENCE

1. The Licensor grants a non-exclusive royalty free licence to reproduce and use the Documents subject to present terms and conditions (the **Licence**) for the parts that are own and proprietary property the of AXMEDIS consortium or its members.
2. In consideration of the Licensor granting the Licence, licensee agrees to adhere to the following terms and conditions.

3. TERM AND TERMINATION

1. Granted Licence shall commence on Acceptance Date.
2. Granted Licence will terminate automatically if licensee fails to comply with any of the terms and conditions of this Licence.
3. Termination of this Licence does not affect either party's accrued rights and obligations as at the date of termination.
4. Upon termination of this Licence for whatever reason, licensee shall cease to make any use of the accessed Copyright.
5. All provisions of this Licence, which are necessary for the interpretation or enforcement of a party's rights or obligations, shall survive termination of this Licence and shall continue in full force and effect.
6. Notwithstanding License termination, confidentiality clauses related to any content, document or part of it as stated in the document itself will remain in force for a period of 5 years after license issue date or the period stated in the document whichever is the longer.

4. USE

1. Licensee shall not breach or denigrate the integrity of the Copyright Notice and in particular shall not:
 - i. remove this Copyright Notice on a Document or any of its reproduction in any form in which those may be achieved;
 - ii. change or remove the title of a Document;
 - iii. use all or any part of a Document as part of a specification or standard not emanating from the Licensor without the prior written consent of the Licensor; or
 - iv. do or permit others to do any act or omission in relation to a Document which is contrary to the rights and obligations as stated in the present license and agreed with the Licensor

5. COPYRIGHT NOTICES

1. All Works shall bear a clear notice asserting the Licensor's Copyright. The notice shall use the wording employed by the Licensor in its own copyright notice unless the Licensor otherwise instructs licensees.

6. WARRANTY

1. The Licensor warrants the licensee that the present licence is issued on the basis of full Copyright ownership or re-licensing agreements granting the Licensor full licensing and enforcement power.
2. For the avoidance of doubt the licensee should be aware that although the Copyright in the documents is given under warranty this warranty does not extend to the content of any document which may contain

references or specifications or technologies that are covered by patents (also of third parties) or that refer to other standards. AXMEDIS is not responsible and does not guarantee that the information contained in the document is fully proprietary of AXMEDIS consortium and/or partners.

3. Licensee hereby undertakes to the Licensor that he will, without prejudice to any other right of action which the Licensor may have, at all times keep the Licensor fully and effectively indemnified against all and any liability (which liability shall include, without limitation, all losses, costs, claims, expenses, demands, actions, damages, legal and other professional fees and expenses on a full indemnity basis) which the Licensor may suffer or incur as a result of, or by reason of, any breach or non-fulfilment of any of his obligations in respect of this Licence.

7. INFRINGEMENT

1. Licensee undertakes to notify promptly the Licensor of any threatened or actual infringement of the Copyright which comes to licensee notice and shall, at the Licensor's request and expense, do all such things as are reasonably necessary to defend and enforce the Licensor's rights in the Copyright.

8. GOVERNING LAW AND JURISDICTION

1. This Licence shall be subject to, and construed and interpreted in accordance with Italian law.
2. The parties irrevocably submit to the exclusive jurisdiction of the Italian Courts.

Please note that:

- You can become affiliated with AXMEDIS. This will give you the access to a huge amount of knowledge, information and source code related to the AXMEDIS Framework. If you are interested please contact P. Nesi at nesi@dsi.unifi.it. Once affiliated with AXMEDIS you will have the possibility of using the AXMEDIS specification and technology for your business.
- You can contribute to the improvement of AXMEDIS documents and specification by sending the contribution to P. Nesi at nesi@dsi.unifi.it
- You can attend AXMEDIS meetings that are open to public, for additional information see WWW.axmedis.org or contact P. Nesi at nesi@dsi.unifi.it

Table of Contents

1	EXECUTIVE SUMMARY AND REPORT SCOPE	8
2	INTRODUCTION ON AXMEDIS FRAMEWORK (DSI).....	9
3	GUIDELINES ON AXMEDIS FRAMEWORK IMPLEMENTATION (DSI)	12
3.1	DECISIONS ON DESIGN, IMPLEMENTATION AND SPECIFICATION TOOLS (ALL)	12
3.2	IDENTIFICATION OF A FRAMEWORK FOR CROSS PLATFORM C++ APPLICATIONS DEVELOPMENT TO BE USED AS BASE OF AXMEDIS FRAMEWORK	12
3.2.1	Requirements for the C++ Framework.....	13
3.2.2	WxWidget	13
3.3	SUPPORTED PLATFORMS (ALL PARTNERS).....	14
3.4	DEVELOPMENT FRAMEWORKS.....	15
3.5	USER INTERFACES (ALL PARTNERS)	15
3.6	COMMUNICATION (ALL PARTNERS).....	16
3.7	INFORMATION EXCHANGE (EXITECH, ALL PARTNERS).....	16
3.8	SPECIFIC NOTES ON DEVELOPMENT OF AXMEDIS TOOLS FOR PDA (DSI: VALLOTTI, TISCALI; NATERI) .	16
3.9	SPECIFIC NOTES ON DEVELOPMENT OF AXMEDIS TOOLS FOR MOBILES (DSI: VALLOTTI, ROGAI, TISCALI; NATERI) .	17
4	GUIDELINES ON AXMEDIS FRAMEWORK SOURCE CODE (DSI)	18
4.1	DOC GENERATION.....	21
4.2	ADOPTION OF LIBRARIES.....	21
4.2.1	Licensing terms (All Partners).....	22
5	GUIDELINES ON AXMEDIS FRAMEWORK MULTILINGUAL ON TOOLS (DSI).....	23
6	GUIDELINES FOR ADDING EXTERNAL LIBRARIES (DSI)	23
6.1	AXMEDIS PLUG IN PURPOSES AND USAGES	23
6.2	AXMEDIS PLUG IN MODEL.....	23
6.3	AXMEDIS PLUG IN DEVELOPMENT.....	24
6.4	AXMEDIS PLUG IN CONTENT PROCESSING AND PROTECTION TOOLS.....	27
6.4.1	Content Processing.....	27
6.4.2	Protection Tools	27
6.5	AXMEDIS PLUG-IN EXAMPLES	28
6.5.1	AXMEDIS Plug-in examples: Developing a Content Processing Plug-In.....	28
6.5.2	AXMEDIS Plug-in examples: Developing a Protection Tool Plug-In.....	32
6.6	AXMEDIS PLUG IN CONTENT PROCESSING AND PROTECTION TOOLS.....	35
6.6.1	Adaptation Tools and Algorithms for Text	35
6.6.2	Adaptation Tools and Algorithms for Video.....	35
6.6.3	Adaptation Tools and Algorithms for Images.....	35
6.6.4	Adaptation Tools and Algorithms for Audio	35
6.6.5	Adaptation Tools and Algorithms for Multimedia.....	36
6.6.6	Adaptation Tools and Algorithms for Metadata	36
6.6.7	Adaptation Tools and Algorithms for DRM information	36
6.6.8	Adaptation Tools and Algorithms for Ringtones	37
6.6.9	Descriptor extractor as fingerprint for Text files	37
6.6.10	Descriptor extractor as fingerprint for Audio files.....	37
6.6.11	Descriptor extractor as fingerprint for Video Files	38
6.6.12	Fingerprint Estimation for Text files.....	38
6.6.13	Fingerprint Estimation for Audio files	38
6.6.14	Fingerprint Estimation for Video files	39
6.6.15	Fingerprint Estimation for Metadata	39
6.6.16	Fingerprint Estimation for Generic Files.....	39

6.6.17	External Protection Libraries	39
7	GUIDELINES ON AXMEDIS COMPONENT VALIDATION AND ACCEPTANCE (FUPF)	40
7.1	COMPONENT VALIDATION	40
7.2	COMPONENT SUBMISSION	41
7.3	REVIEW AND ACCEPTANCE REPORT FORM	41
7.4	VERIFICATION REPORT FORM	42
7.5	COMPONENT ACCEPTANCE	42
7.6	START UP OF COMPONENT VALIDATION AND ACCEPTANCE	42
7.7	PERIODIC VERIFICATION	43
7.8	ACCEPTANCE TESTING	43
8	GUIDELINES ON AXMEDIS FRAMEWORK VALIDATION (EXITECH)	45
8.1	SOURCE CODE REPOSITORY MODEL	45
8.1.1	Premises	45
8.1.2	Repository Structure	45
8.1.3	General Guidelines.....	49
8.1.3.1	Directory include (optional: only for C/C++)	49
8.1.3.2	Directory source (mandatory)	49
8.1.3.3	Directory doc/specification	49
8.1.3.4	Directory doc/code (mandatory)	50
8.1.3.5	Directory doc/test (mandatory)	50
8.1.3.6	Directory doc/configuration-deployment (optional: only for modules requiring deployment).....	50
8.1.3.7	Directory doc/other (optional).....	50
8.1.3.8	Directory lib (mandatory).....	50
8.1.3.9	Directory bin (mandatory).....	51
8.1.3.10	Directory project (optional: only for modules requiring building)	51
8.1.4	C/C++ Application.....	51
8.1.4.1	Directory include	51
8.1.4.2	Directory source	51
8.1.4.3	Directory doc/specification	51
8.1.4.4	Directory doc/test	51
8.1.4.5	Directory doc/configuration-deployment.....	51
8.1.4.6	Directory doc/other.....	51
8.1.4.7	Directory lib.....	51
8.1.4.8	Directory bin.....	51
8.1.5	C/C++ Dynamic Library	51
8.1.5.1	Directory include	51
8.1.5.2	Directory source	51
8.1.5.3	Directory doc/specification	51
8.1.5.4	Directory doc/code	52
8.1.5.5	Directory doc/test	52
8.1.5.6	Directory doc/configuration-deployment.....	52
8.1.5.7	Directory doc/other.....	52
8.1.5.8	Directory lib.....	52
8.1.5.9	Directory bin.....	52
8.1.6	C/C++ Static Library	52
8.1.6.1	Directory include	52
8.1.6.2	Directory source	52
8.1.6.3	Directory doc/specification	52
8.1.6.4	Directory doc/test	52
8.1.6.5	Directory doc/configuration-deployment.....	52
8.1.6.6	Directory doc/other.....	52
8.1.6.7	Directory lib.....	52
8.1.6.8	Directory bin.....	52
8.1.7	C/C++ WebService	52
8.1.7.1	Directory include	52
8.1.7.2	Directory source	52
8.1.7.3	Directory doc/specification	53
8.1.7.4	Directory doc/code	53
8.1.7.5	Directory doc/test	53
8.1.7.6	Directory doc/configuration-deployment.....	53
8.1.7.7	Directory doc/other.....	53

8.1.7.8	Directory lib.....	53
8.1.7.9	Directory bin.....	53
8.1.8	Java Application.....	53
8.1.8.1	Directory include.....	53
8.1.8.2	Directory source.....	53
8.1.8.3	Directory doc/specification.....	53
8.1.8.4	Directory doc/code.....	53
8.1.8.5	Directory doc/test.....	53
8.1.8.6	Directory doc/configuration-deployment.....	53
8.1.8.7	Directory doc/other.....	53
8.1.8.8	Directory lib.....	53
8.1.8.9	Directory bin.....	54
8.1.9	Java Library (JAR).....	54
8.1.9.1	Directory include.....	54
8.1.9.2	Directory source.....	54
8.1.9.3	Directory doc/specification.....	54
8.1.9.4	Directory doc/code.....	54
8.1.9.5	Directory doc/test.....	54
8.1.9.6	Directory doc/configuration-deployment.....	54
8.1.9.7	Directory doc/other.....	54
8.1.9.8	Directory lib.....	54
8.1.9.9	Directory bin.....	54
8.1.10	Java WebService.....	54
8.1.10.1	Directory include.....	54
8.1.10.2	Directory source.....	54
8.1.10.3	Directory doc/specification.....	54
8.1.10.4	Directory doc/code.....	54
8.1.10.5	Directory doc/test.....	54
8.1.10.6	Directory doc/configuration-deployment.....	55
8.1.10.7	Directory doc/other.....	55
8.1.10.8	Directory lib.....	55
8.1.10.9	Directory bin.....	55
8.2	GUIDELINES FOR THE SUBMISSION PROCESS.....	55
8.3	GUIDELINES FOR CHECKING-OUT, UPDATING COMMITTING.....	56
8.4	GUIDELINES FOR ADDING NEW EXTERNAL LIBRARIES TO THE AXFW (ALL).....	60
8.4.1	Addition of external libraries.....	60
8.4.2	Authorisation of external libraries use.....	60
8.5	SOURCE CODE REPOSITORY PRESENT STATUS.....	60
8.5.1	WebService Formal Validation (Revision 1312 of the repository).....	60
8.5.2	Application Formal Validation (Revision 1312 of the repository).....	61
8.5.3	Framework Formal Validation (Revision 1312 of the repository).....	63
9	GUIDELINES ON AXMEDIS FRAMEWORK INTEGRATION AND MAINTENANCE (FUPF).....	65
9.1	LINKED MODULES.....	65
9.2	INTEGRATION TESTING HOW TO.....	65
9.3	INTEGRATION REVIEW REPORT FORM.....	65
9.4	VERIFICATION REPORT FORM.....	66
10	GUIDELINES ON AXMEDIS FRAMEWORK REGRESSION TESTING (FUPF).....	67
10.1	UNIT TEST TOOLS.....	67
10.2	REGRESSION TESTING GUIDELINES.....	67
10.3	TESTING USING CPPUNIT.....	68
11	GUIDELINES ON PERFORMANCE ASSESSMENT AND OPTIMISATION (DSI).....	69
11.1	REFERENCE VALUES.....	69
11.2	LOAD TESTS MEANING FOR DATABASES AND WEBSERVICES.....	70
11.3	AXMEDIS AXCS (DSI).....	73
11.3.1	AXCS Registration and Certification database (DSI).....	73
11.3.1.1	Table size / # of tuple.....	73
11.3.1.2	Search Time / # of tuple.....	74
11.3.1.3	Single and Multiple access tests comparison.....	75
11.3.2	AXCS – ObjectsID (DSI).....	77

11.3.2.1	Database size / # of tuple	77
11.3.2.2	Search Time / # of tuple.....	77
11.3.2.3	Single and Multiple access tests comparison	78
11.3.3	AXCS – Accounting (DSI).....	80
11.3.3.1	Database size / # of tuple	80
11.3.3.2	Search Time / # of tuple.....	80
11.3.3.3	Single and Multiple access tests comparison	81
11.3.4	Users Registration Web Service (DSI).....	83
11.3.4.1	Time Tests\.....	83
11.3.4.2	Capacity Tests	83
11.3.4.3	Concurrency Tests.....	85
11.3.5	Objects Registration Web Service (DSI).....	85
11.3.5.1	Time Tests\.....	85
11.3.5.2	Capacity Tests	86
11.3.5.3	Concurrency Tests.....	87
11.3.6	Reporting Web Service (DSI).....	88
11.3.6.1	Time Tests\.....	88
11.3.6.2	Capacity Tests	88
11.3.6.3	Concurrency Tests.....	90
11.3.7	Statistics Web Service (DSI)	90
11.3.7.1	Time Tests\.....	90
11.3.7.2	Capacity Tests	91
11.3.7.3	Concurrency Tests.....	92
11.3.8	AXMEDIS AXCV and AXS Web Services (FUPF)	93
11.3.8.1	Time Tests	93
11.3.8.2	Capacity Tests	93
11.3.8.3	Concurrency Tests.....	95
11.4	AXMEDIS PMS (FUPF)	95
11.4.1	AXMEDIS PMS Server License database (FUPF).....	95
11.4.1.1	Database size / # of tuple	95
11.4.1.2	Search Time / # of tuple.....	96
11.4.1.3	Single and Multiple access tests comparison	97
11.4.2	AXMEDIS PMS Server Web Service (FUPF)	98
11.4.2.1	Time Tests	98
11.4.2.2	Capacity Tests	99
11.4.2.3	Concurrency Tests.....	101
11.5	AXMEDIS DATABASE (EXITECH, FUPF)	101
11.5.1	Loader Web Service [EXITECH].....	101
11.5.1.1	Response Time for Number of connection.....	101
11.5.2	Saver Web Service [EXITECH].....	102
11.5.2.1	Response Time for Number of connection.....	102
11.5.3	Quey Support Web Service for Metadata [EXITECH].....	103
11.5.3.1	Response Time for Number of connection.....	103
11.5.4	Query Support Web Service for PAR [FUPF]	104
11.5.4.1	Time Tests	104
11.5.4.2	Capacity Tests.....	105
11.5.4.3	Concurrency Tests.....	107
11.6	AXMEDIS CONTENT PROCESSING GRID (DSI).....	107
11.6.1	AXMEDIS Rule Scheduler.....	107
11.6.2	AXMEDIS Rule Executor, or Engine	108
11.7	AXMEDIS CONTENT PROCESSING ALGORITHMS (DSI, FHGIGD).....	108
11.8	AXMEDIS P2P SERVER FOR QUERIES (EXITECH).....	108
11.9	AXMEDIS P2P AXEPTOOL AND AXMEDIA TOOLS (DSI, HEXAGLOBE)	108
11.9.1	AXMEDIS P2P Tracker (DSI, HEXAGLOBE)	108
12	GUIDELINES FOR VIDEO DEMONSTRATION ACQUISITION (DSI: NICOLA MITOLO).....	110
12.1	HOW TO RECORD WITH BBFLASHBACK RECORDER	110
12.2	STRUCTURE OF VIDEO PRESENTATIONS.....	114

1 Executive Summary and Report Scope

This document concerns:

- ♣ Implementation guidelines
This section describe and specify all postulates regarding AXMEDIS Framework creation, including coding language, project tools and behavior, wide used libraries and environmental issues.
- ♣ Guidelines for code development
This section describe how an AXMEDIS code module should look in terms of code organization
- ♣ Guidelines on AXMEDIS Framework Multilingual on Tools
This section describe guidelines for localization of tools in AXMEDIS Framework
- ♣ Guidelines on adding new external libraries
This section describes how to develop plug-ins to add new Content Process and Protection Tools functionalities to AXMEDIS Framework
- ♣ AXMEDIS component validation and acceptance,
This section describes guidelines to validate an AXMEDIS module and acceptance test each module should pass to assure its quality level.
- ♣ AXMEDIS framework validation
This section describes validation methods for the whole AXMEDIS Framework
- ♣ AXMEDIS framework integration and maintenance,
This section describes behaviors to test integration and compatibility among AXMEDIS Modules
- ♣ Regression testing guidelines
This section describes behaviors to perform regression testing activities on AXMEDIS Modules
- ♣ Assessment optimisation of components
This section involves guidelines for realization of stress tests to be used in the optimization of AXMEDIS Modules.
- ♣ Guidelines for Video demonstrator acquisition
This section describes how to use specific tools to realize video demonstrators

2 Introduction on AXMEDIS Framework (DSI)

AXMEDIS is an ambitious Integrated Project of Research and Development partially founded by the European Commission in IST FP6 and including about 20 partners such as University of Florence, HP, EPFL, FHGIGD, ACIT, AFI, TISCALI, University Pompeo Fabra, University of Leeds, CPR, EXITECH, XIM, University of Reading, etc. The duty of AXMEDIS is to work on research activities, develop new tools and products and trial them as effective demonstrators.



AXMEDIS goals

AXMEDIS is creating and developing the AXMEDIS Framework, an open solution exploiting a set of new technologies and tools, which can be used by your solutions and applications for:

- reduction of costs and increasing efficiency for content production, protection, management and distribution; better pricing and value-for-money for industry products and services, containing costs to set up sustainable business ventures in the digital cross media content:
 - integrating your Content Management Systems, CMSs, with the distribution systems by automating the communication and update of content and information between the two systems;
 - automating content gathering and ingestion processes from local or remote CMSs and file systems;
 - automating composition, allowing parallel processing, exploiting GRID technology, and optimization techniques for content ingestion, production, protection and formatting;
 - managing the workflow at level of the content factory and among different content factories sharing the same content production objectives;
 - automating the whole process allowing content production on demand;
- support for the whole value chain: composition, packaging, integration, aggregation, synchronization, formatting, adaptation, transcoding, indexing, integration in the same objects protected and non protected components, definition of relationships with other resources, metadata integration and remapping/transcoding, protection, license production and verification;
- convergence of the media, interoperability of content supporting the multichannel distribution, support content distribution:
 - on different channels such as satellite data broadcast, Internet, cellular network, wireless, traditional supports as DVDs, internet, mobiles networks, local and wireless networks;
 - including Peer-to-Peer (P2P) in both B2B (Business-to-Business) and B2C (Business-to-Consumer) levels;
 - on different devices such as PC, PDA, i-TV, STB, etc.;

- with different transaction models on the same channels and content with flexibility in the business and transaction models;
- adoption of new methods and tools for innovative, flexible and interoperable Digital Rights Management (DRM), including
 - exploitation of MPEG-21 REL (Right Expression Language) and overcoming its limitations with specific extensions,
 - supporting different business and transactions models and their integration,
 - supporting the integration/interoperation of different DRM models such as MPEG-21 REL and ODRL OMA (Open Mobile Alliance);
- harmonization of B2B and B2C areas for DRM, bringing the DRM model in the B2B area, supporting production and protection models in the whole value chain;
- increment of content accessibility with a P2P platform at B2B level, which can integrate content management systems and workflows.

AXMEDIS implements the AXMEDIS Framework for all, and especially for small and large industries sharing a common interest in the exploitation of new technologies and solutions. The AXMEDIS Framework can be used to setup and build a set of complete applications and services in the area of content production, protection and distribution. With the flexibility of AXMEDIS dynamic Plug-In technology, you can customize your applications and processes according to your needs.

The AXMEDIS digital content and content components is an **open format** capable of integrating any kind of cross media format (video, images, animations, games, learning objects, multimedia, audiovisual, document, audio, etc.) in any digital format, any kind of metadata including identification, classification, categorization, indexing, descriptors, annotation, relationships and play activities and protection aspects.

The AXMEDIS format permits the combination of content components and their secure distribution in respect of the copyright laws, supporting a large variety of DRM rules and models according to concepts of interoperability among DRM models (mainly, but not only, based on MPEG-21, with both binary and XML low level formats). AXMEDIS is open to any DRM model and solution.

Within the AXMEDIS content any type of cross media content can be included from simple multimedia files to games or software components, for leisure and entertainment, infotainment, and also for managing protected governmental content, healthcare information, business of value information, etc.

The AXMEDIS framework is an environment for integrating and validating the new enabling technologies and new knowledge invented with WP4, WP5, and WP8. In this section the guidelines for creating software components for the AXMEDIS framework is reported. These software components are created by exploiting and including research algorithms as described in WP4. The guidelines and most of the components are accessible to the whole community together with the components developed by partners, and accessible in the AXMEDIS Framework: <http://www.axmedis.org> .

This document describes the **AXMEDIS open architecture and framework**. It is open since:

- all the AXMEDIS specification is public and its specific use is royalty free. Any company or third party can use the document to create an AXMEDIS compatible solution;
- all the source code of AXMEDIS Framework is accessible by getting affiliated with AXMEDIS. The affiliation fee is low and affordable for all;
- all the algorithms of the same type will be interchangeable, any innovation in the format, in the process, in the workflow, in the business model, in the DRMs, can be added into the framework without restructuring;
- The structure of the AXMEDIS framework is well finalised in terms of interfaces and protocols among its components. They are mainly algorithms and software modules for managing content: composition, formatting protection, query, etc.;
- the affiliation to AXMEDIS can be obtained also providing work or results to the community. So that you can have the access to the AXMEDIS Framework in change of your contribution in improving and extending it;

DE5.1.3.1 – AXMEDIS Framework Guidelines

- the AXMEDIS plug-in technology is public, and the source code for creating new plug-in is public without needs to be affiliated;
- in AXMEDIS the focus is on interoperability and openness of content model, including multichannel distribution;
- in AXMEDIS the focus is on interoperability of DRM model, including multichannel distribution.
- Other distribution channels can be added according to the above solutions: direct channels such as those towards PCs or PDAs or, mediate via Channel Distributors such as those towards i-TVs;
- The same channel structure can be duplicated without any problem. You may have more Channel Distributors for i-TV, pay per view, etc. They can be set up for localization of content and services, for language and cultural differences, for providing content towards different technologies for providing different content;
- No limitation about the number of clients;
- No limitation about the number of transactions;

More technical information on AXMEDIS architecture and framework and about how to access at the AXMEDIS framework getting affiliated to AXMEDIS are available on <http://www.axmedis.org> .

3 Guidelines on AXMEDIS Framework implementation (DSI)

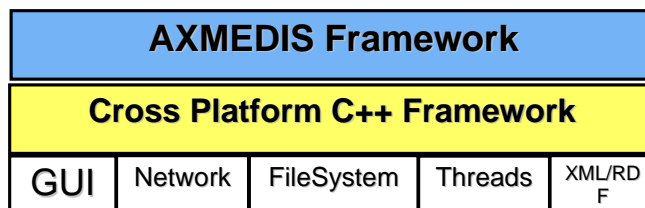
The AXMEDIS framework is an environment for integrating and validating the new enabling technologies and new knowledge invented with WP4, WP5, and WP8. In this section the guidelines for creating software components for the AXMEDIS framework is reported. These software components are created by exploiting and including research algorithms as described in WP4. The guidelines and most of the components are accessible to the whole community together with the components developed by partners, and accessible in the AXMEDIS Framework: <http://www.axmedis.org> .

3.1 Decisions on Design, Implementation and Specification Tools (ALL)

- UML: a tool will be selected, VISIO 2003 professional is one of the most largely used in AXMEDIS, other possibilities are UML free tools, etc. They have to be capable of processing UML, verification of consistency and managing packages, action diagrams, IDL, etc....
- Model for API is based on IDL, which in turn is based on UML formalisation
- All the tools including the AXMEDIS Object Manager have to be developed in C++
- AXCS and AXMEDIS object database have to provide for the database access some ODBC or JDBC, this has still to be decided [DSI Chellini-Martini] At the present time, AXCS database accesses are performed by means of JDBC technology
- Technology for AXMEDIS PMS has been fixed to be C++ for Servers
- Communications with AXCS and PMS and clients and tools is in SSL
- Database technology is scalable: SQL, MYSQL, DB2, Postgress, etc.
- AXEPTool has to be based on non proprietary protocol such as HTTP
 - The same DB used in the AXMEDIS database has to be used in the AXEPTool
- AXMEDIS content processing and editing tools have to be realized in C++, firstly for WINDOWS but taking into account of porting them on MAC/LINUX. So that a GUI abstraction is needed such as Mozilla or WXwin, a scripting tool for the GUI can be taken into consideration..
- AXMEDIS Protection tools have to be realized in C++, firstly for WINDOWS but taking into account of porting them on MAC/LINUX. So that a GUI abstraction is needed such as Mozilla or WXwin, a scripting tool for the GUI can be taken into consideration..

3.2 Identification of a Framework for Cross Platform C++ Applications Development to be used as base of AXMEDIS Framework

In this section a Framework for Cross Platform C++ Applications Development to be used as base of AXMEDIS Framework is identified.



The work performed is sketched in the following steps:

1. Identification of requirements for the C++ Framework
2. Selection of candidates
3. Evaluation of candidates

3.2.1 Requirements for the C++ Framework

The requirements identified for the C++ Framework to be used as the basis of AXMEDIS Framework are the following.

The **C++ Framework** has to:

- 1 be multiplatform supporting at least: MS Windows (98?, 2000, XP), MACOS X, Linux
- 2 NOT be under GPL license
- 3 NOT be under commercial license (NOT MANDATORY)
- 4 be rather wide spread
- 5 be usable from a C++ Application
- 6 allow the realization of standard user interfaces as well as MDI, drag&drop support, tree view, details view, clip board,...
- 7 allow run-time generation of user interfaces
- 8 allow the realization of custom views, where the application can control the visualization and the interaction with the view.
- 9 allow cross platform access to basic resources like: File system, Network (TCP & UDP), Threads, XML Parsing, RDF.
- 10 render multimedia information (video, audio, images) (RELEVANT)
- 11 allow the realization of applications with skin (like Windows Media Player) (OPTIONAL) or not prevent its realization on some platforms (MANDATORY).

3.2.2 WxWidget

For the candidates will be evaluated:

- platforms supported
- licence
- diffusion
- GUI capabilities
 - basic user interfaces
 - advanced user interfaces
 - support for automatic run-time generation of user interfaces
 - custom user interfaces
 - support for multimedia
 - skin support
- Platform abstraction
 - file system
 - network
 - threads
- Tools
 - xml support
 - rdf support

wxWidgets

General	
Platforms	Win32, MAC OSX, Linux, OS2, palmOS, winCE
Licence	LGPL
Diffusion	Good
GUI capabilities	
Basic GUI	Good
Advanced GUI	Good
Custom GUI	Good
Automatic GUI	Feasible but not directly supported (XRC is used for XML description of GUI)

Multimedia Support	Native support for images (jpg, png, tiff), contributions are present for some multimedia support
Skin support	Feasible but not directly supported
Platform abstraction	
File system	Ok, it supports also zip files
Network	Ok (not UDP)
Thread	Ok
XML	Marginal, a non validating parser is present
RDF	No
Notes	
A library for XML parsing may be used (XERCES)	

In the above evaluation of candidates, are marked in red/orange/yellow the features that are problematic. The features marked in red strongly discourage the adoption of the framework, the ones marker in orange suggest to avoid to use it.

It results that wxWidgets, wxPython, and GTK+ are the frameworks that could be adopted. However it seems that wxWidgets gives more coverage especially since it can be used also in PDAs.

3.3 Supported platforms (All Partners)

The components produced by the research activities will have to be compliant with the languages and platform identified in the specification and in particular:

- Content production tools, editors, publication tools, engines, protection tool editor, etc.:
 - Language: C++
 - OS: Windows XP and 2000, and viable as a second choice also for MACOS X
 - GUI: WxWidget (the new version of the WxWindows)
 - XML Parser: XERCES
 - WebServices/SOAP: gSoap
 - TCP/IP library: that of WxWidget
 - STL can be used but a particular attention has to be given on using only functionalities and data structures that are supported in the STL for PDA (Pocket PC 2003)
 - Avoiding the usage of functionalities that are Microsoft specific
- Protection Manager Support
 - Client: all features as above in C++
 - Domain Home, Domain Factory and Server: Java and C++
- AXEPTool:
 - Publication and loading engines: as the editor area above
 - C++ for the tools
 - P2P virtual database: Java if needed
 - Graphic User Interface: WxWidgets
- AXMEDIS Database and Query Support and AXCS:
 - Database technology: MySQL or Postgres
 - Technology for coding logic: Java
 - DBC: JDBC for the access
 - Operating system: Windows and Linux.
 - User Interface: JSP or PHP, probably better the JSP formalization in classes and graphic design
 - Web Server: Apace, TOMCAT
- Scripting language:
 - Java Script (ECMA Script)
- Workflow Manager:

- based on OpenFlow or BizTalk
- Plug in of the workflow: the same as the content production area
- The usage of AXMEDIS Error Manager for all the tools in C++ that use an AXOM
- The usage of AXMEDIS Configuration Manager for all the tools in C++ that use an AXOM
- Fingerprint and Metadata/Descriptors extractors:
 - the same as the content production area

Other information is collected into the AXFW specification document.

3.4 Development frameworks

Development should be done using C/C++ for all code-sharing components, whereas all internal components may be developed using other languages with related tools.

Since the source code of all the tools and components will be accessible to all partners, the choice of a preferred development framework will not limit the possibility to use a different one for regular development and to build for the preferred development framework before upload or on less regular basis.

The following are the preferred development platforms for the project partners under windows:

Partner	Preferred Development Framework
DSI	MS Visual Studio .NET 2003
DIPITA	gcc/CygWin
COMVERSE	MS Visual Studio .NET 2003, MS Visual Studio 6 (Service Pack 5)
EPFL	MS Visual Studio .NET 2003
EUTELSAT	gcc/CygWin
FHGIGD	MS Visual Studio .NET 2003 (MS VS 6.0 and gcc+eclipse as secondary frameworks)
ILABS	MS Visual Studio .NET 2003
HP	
TISCALI	
FUPF	MS Visual Studio 6.0 or gcc (linux tools)
XIM	gcc or MS Visual Studio .NET 2003
CRS4	gcc
SEJER	gcc/CygWin
UNIVLEEDS	MS Visual Studio .NET 2003
IRC	MS Visual Studio .NET 2003
EXITECH	MS Visual Studio .NET 2003 or gcc

Thus the most preferred development framework is MS Visual Studio .NET 2003.

For Java development the choice of a uniform virtual machine will allow to reduce the possibility of integration problems. The chosen JVM is Sun 1.4.2. The migration to a new JVM (1.5.0) will be made all together.

The use of a specific Integrated Development Environment for Java development is not set, but it is mandatory the adoption of ANT for having an easier integration (some IDE use ANT as integrated building system). For testing the use of JUNIT is appreciated. For development of WebServices Java WSDP 1.5 has to be used.

3.5 User Interfaces (All Partners)

For the user interface in C++ applications wxWidgets should be used.

3.6 Communication (All Partners)

For the communications among modules in different processes/machines WebServices is the preferred technology to be used. However in some cases other solutions like XML-RPC or custom protocols could be used, for example for the interaction with existing tools or for performance reason.

For the interaction with modules in the same process static or dynamic libraries should be used. COM/ActiveX technology could be used to interact with existing components.

3.7 Information exchange (EXITECH, All Partners)

This section contains guidelines on the way in which information will be exchanged, through documents posted on the WEB AXMEDIS portal and by synchronizing source code and test cases updates

The main mode for information exchange between the AXMEDIS members is the use of document repository on the WEB PORTAL. Refer to the portal specification for the operative procedures.

The documents editing has to follow the rules defined by the AXMEDIS consortium. The templates to be followed for the document editing have to be posted on the WEB site into the management activity in a dedicated folder. New templates have to be created any time a new type of document has to be produced (i.e. slide template, deliverable template, reports and management reports, UML diagrams, etc.). The document responsible and the activity coordinators have to verify if the received/posted content comply with the template.

In case a new document type is needed, the template can be proposed to the project coordinator, who, if accept it, will post it on the portal.

It is strongly advised against sending documents in attach to the reflector

Also for the code editing a “template” can be created. The programming style template can be based on some standard guidelines as:

- C++ Coding Standard defined at <http://www.possibility.com/Cpp/CppCodingStandard.html>
- Mozilla Coding Style Guide at <http://www.mozilla.org/hacking/mozilla-style-guide.html>
- Code Conventions for the Java™ Programming Language defined by SUN at <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>
- Apache coding standards at <http://jakarta.apache.org/turbine/common/code-standards.html>

The standard guideline can be adapted for the project needs.

3.8 Specific Notes on Development of AXMEDIS Tools for PDA (DSI: Vallotti, TISCALI: Nateri)

Pocket PC 2003 with Windows CE version 4.20 (WCE420) has been chosen as main testing platform. It is not the last version of Microsoft mobile operative system but it is quite new therefore it is widespread and provides a good set of basic functionalities. Moreover, it is compatible with new version of Microsoft OS for mobiles.

Microsoft eMbedded Visual C++ 4.0 (eVC4) has been chosen as reference development environment since it provides a quite new C++ compiler and it can be enriched with a wide set of Standard Development Kit for different OSs and hardware platforms. The following libraries have to be used (and eventually ported) for PDA developing:

- STLPort version 5.0.1 or greater;
- Xerces-C++ version 2.7.0;
- OpenSSL version 0.9.8a or greater with WCECompat version 1.2;
- Libcurl version 7.15.0.

Some workarounds have to be provided in order to fill the gap due to some missed functionalities in the Windows CE operative system. For example, Windows CE does not support the concept of “current directory”. For that reason, utility function will be provided by the AXMEDIS framework in order to enable installation of AXMEDIS tools on PDAs.

3.9 Specific Notes on Development of AXMEDIS Tools for Mobiles (DSI: Vallotti, Rogai, TISCALI: Nateri)

Since C++ development on mobile is not as flexible as on other platforms, it is feasible to adopt other solution to provide AXMEDIS tools for mobiles. In particular, the main effort has be done in creating a Java-based core library to enable playing of streamed content on J2ME virtual machines. From this point of view, some restriction have to be defined for content which can be used on mobiles. Since mobile platforms have a footprint lower than PDA ones, it is suitable realizing a very simple version of basic libraries such as AXOM and PMS Client. In particular, the mobile version of the AXOM should be able to manage simple AXMEDIS objects, protected in simple manner and it should not provide advanced functionalities such as referred objects, dynamic commands, etc. On the other hand, PMS Client should provide only “online” functionalities, i.e. it should not provide license and protection information caching or context related features.

4 Guidelines on AXMEDIS Framework source code (DSI)

For Java source code use guidelines <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>

For C++ source code:

Source files

- Header files have to have **.h** extension.
- Implementation source files have to have **.cpp** extension.
- All file names have to be lowercase (e.g. *axobject.h*).
- A header file should normally contain the definition of one class, more than one class can be defined in one .h file if they are strongly dependent.
- When including files from directories use forward slash “/” and not back slash “\” (e.g. *#include “wx/dc.h”*)
- Don’t use tabs to indent code, use 4 spaces.
- Use 76 columns for coding

Naming conventions

- class names should begin with capital letter (e.g. *AxObject*)
- class attributes and methods should begin with lower case (e.g. *model*, *getModel()*)
- local variables and functions should begin with lower case;
- macro, enum values and constants should be in all capital letters with “_” as word separator (e.g. *READ_ONLY*)
- namespaces should be in all lower case letters
- typedefs names should end with Type suffix (e.g. *typedef ParamValueType<int> ParamIntType*)
- for accessors use get/set prefixes (e.g. *getModel()*, *setValue(x)*)
- for classes belonging to the AXMEDIS Framework that are exposed to the framework user use the “Ax” prefix (e.g. *AxObject*), for classes that are not exposed this is not mandatory.

Code formatting

- for brace placement follow one of the following rule, but use the same for the whole module

- Place brace under and inline with keywords:

```
if (condition)           while (condition)
{                         {
    ...                   }
}                         ...

```

- Traditional Unix policy of placing the initial brace on the same line as the keyword and the trailing brace inline on its own line with the keyword:

```
if (condition) {         while (condition) {
    ...                   }
}                         ...

```

- in case a line is too long split it leaving the operator at the end of the previous line and align the starting of the continuing line with the expression start, examples:

```
if (aaaaaaaaaa &&
    bbbbbbbbbb &&
    ccccccccc &&
    ddddddddd &&
    eeeeeeeee)
{
    ...
}

aFunction(aaaaaaaaa,
```

```
bbbbbbbbbb,  
cccccccccc,  
dddddddddd);
```

Class Header Template

```
/**  
 * A one line description of the class.  
 *  
 * #include "XX.h" <BR>  
 * -llib  
 *  
 * A longer description.  
 *  
 * @see something  
 */  
  
#ifndef XX_h  
#define XX_h  
  
// SYSTEM INCLUDES  
//  
  
// PROJECT INCLUDES  
//  
  
// LOCAL INCLUDES  
//  
  
// FORWARD REFERENCES  
//  
  
class XX  
{  
public:  
    // LIFECYCLE  
  
    /**  
     * Default constructor.  
     */  
    XX(void);  
  
    /**  
     * Copy constructor.  
     *  
     * @param from The value to copy to this object.  
     */  
    XX(const XX& from);  
  
    /**  
     * Destructor.  
     */  
    ~XX(void);  
  
    // OPERATORS  
  
    /**  
     * Assignment operator.
```

```

*
* @param from The value to assign to this object.
*
* @return A reference to this object.
*/
XX& operator=(XX& from);

// OPERATIONS
// ACCESS
// INQUIRY

protected:
private:
};

// INLINE METHODS
//

// EXTERNAL REFERENCES
//

#endif // XX_h_

```

Class Implementation Template

```

// SYSTEM INCLUDES
//

// PROJECT INCLUDES
//

// LOCAL INCLUDES
//

#include "XX.h" // class implemented

//////////////////////////////////// PUBLIC //////////////////////////////////////

//===== LIFECYCLE =====

XX::XX()
{
} // XX

XX::XX(const XX&)
{
} // XX

XX::~~XX()
{
} // ~XX

//===== OPERATORS =====

XX&
XX::operator=(XX&);

```

```

{
    return *this;
} // =

//===== OPERATIONS =====
//===== ACCESS =====
//===== INQUIRY =====
////////////////////// PROTECTED ////////////////////////

////////////////////////////////////// PRIVATE ////////////////////////////////////////
    
```

for what not explicitly stated try to follow the guidelines provided in:
<http://www.possibility.com/Cpp/CppCodingStandard.html>

4.1 DOC generation

Documentation of source code (C++ and Java) will be integrated in the code using JavaDoc comments style.

4.2 Adoption of Libraries

Libraries used in the project are:

Library and version	Area	Licence
wxWidget – 2.4.2	GUI	LGPL
FL C++ Lib contribution to wxWidgets Lib	GUI	LGPL
STC C++ based on Scintilla editor, contribution to the wxWidgets Lib	GUI	LGPL
wxImagick	Image processing	LGPL
Imagick	Image processing	LGPL
openssl	Protection	LGPL
cryptlib	Protection	GPL and standard commercial license
xerces-C++ - 2.6.0	XML parser	Apache Licence 2.0
gSoap	Web services	LGPL
CURL – 7.12.13	Crawler	BSD
easysoap	Crawler	LGPL
expat	Crawler	LGPL
libxml2	Crawler	LGPL
ODBC	Crawler	LGPL
PEAR Library	Database	LGPL
Xerces	Database	Apache Licence 2.0
Xalan	Database	Apache Licence 1.1
XPath	Database	
OpenFlow – 1.1	Workflow	GPL 2.0
Zope – 2.7.3	Workflow	ZPL 2.0
Phyton -2.3	Workflow	
XmlrpcLib	Workflow	
Cexpat	Workflow	
Microsoft ASP	Workflow	Microsoft licence
Microsoft .NET	Workflow	Microsoft licence
DirectX SDK	Players	
splay	Players	LGPL

faac	Players	LGPL
im1_dmif_mp4	Players	ISO
im1_dmif_trif	Players	ISO
im1_dmif_remote	Players	ISO
im1_dmifclientfilter	Players	ISO
DOCFRAC	Fingerprints	LGPL
GNU_ghostscript	Fingerprints	GPL
XPDF	Fingerprints	GPL
HTMLDOC	Fingerprints	GPL
WordNet (English, Italian, Spanish, French, German)	Fingerprints	Free for English, proprietary for other languages
TreeTagger	Fingerprints	Free for research. Proprietary for commercial use.
CLAM (0.7)	Fingerprints	GPL
Torch3	Fingerprints	BSD
LibSVM – 2.71	Fingerprints	LGPL
Libsndfile – 1.0.11	Fingerprints	LGPL
BeeCrypt	Fingerprints	LGPL
Botan	Fingerprints	BSD-style
CryptLib (?)	Fingerprints	GPL and standard commercial license
RtAudio – 3.0	Fingerprints	BSD-style open source
PortAudio – 18	Fingerprints	BSD-style open source
Libsndfile – 1.0.11	Fingerprints	LGPL
FFTW – 3.0.1	Fingerprints	GPL and Non-free license (see http://web.mit.edu/tlo/www/)
FFMPEG	Fingerprints	LGPL
FOBS	Fingerprints	LGPL
SpiderMonkey JavaScript Engine ver. 1.5 by Mozilla	Engine	LGPL
SoundTouch	Adaptation	LGPL

4.2.1 Licensing terms (All Partners)

- license for libraries and access to code modality as stated in the CA for each module produced and reused

5 Guidelines on AXMEDIS Framework Multilingual on Tools (DSI)

The multilingual requirements on AXMEDIS tools is not one of the mandatory requirements in this phase. The tools developed should take into account that in the life of the product tool it would be possible and need to have multilingual tools.

6 Guidelines for Adding External Libraries (DSI)

The AXMEDIS Tools allows including the functionality of external libraries through an advanced plug-in interface. Through this plug-in interface different external libraries are already included. The included plug-ins can be grouped in different categories:

- **Content adaptation** tools and algorithms are the algorithms that allow automatic content processing. The input and the output content type are the same. E.g. typical applications are scenarios where content is adapted according to the needs of the receiver.
- **Content description** tools and algorithms automatically extract low or high level descriptors from content. AXMEDIS supports the overall range of the content descriptors through is general description.
- **Content fingerprinting** tools and algorithms are a subset of low level descriptors. Their purpose is the identification and authentication of content.
- **External processing** algorithms are external algorithms not belonging to the above group that are so far considered to be integrated within the AXMEDIS framework.

6.1 AXMEDIS Plug in purposes and usages

AXMEDIS plug-ins aim to enable the enrichment of AXMEDIS framework functionalities. In fact, using the plug-in technology, the AXOM, the basic library in the AXMEDIS framework, can be enriched with new functionalities in order to manipulate and protect AXMEDIS Objects. The two main application fields currently supported are the creation of new Protection Tools and Content Processing Functions.

Plug-ins are loaded, independently by their category, by the Plug-in Manager. It organizes them on the basis of their category allowing specific modules easily retrieving them. These modules (such as Content Processing and Protection Processor) provide specific functions to invoke functionalities provided by the loaded plug-ins.

Within the AXMEDIS Tools different workflow processed and content processing operations are supported. To increase the flexibility and therefore the potential of the AXMEDIS framework, external libraries can be integrated into the different workflow processed and content processing operations. So far, the following use cases are supported:

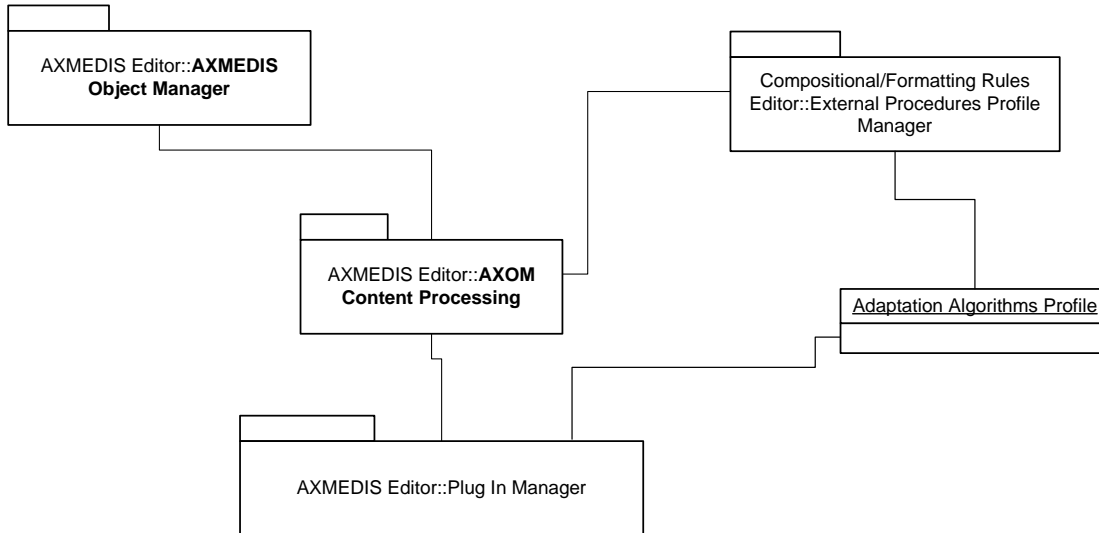
1. **Content Crawling**: calling external databases and transferring content into an AXMEDIS database
2. **AXMEDIS Object Editing**: using the AXMEDIS editor
3. **Automatic Content Processing**: using the AXCPengine
4. **Receiving Content from the AXEPTool**: content that is received via the AXEPTool is automatically authenticated

6.2 AXMEDIS plug in model

Within the AXMEDIS Framework, the AXMEDIS Plug-in Manager supports the installation/registration of plug-ins, loading of plug-ins. AXMEDIS Editor Plug-in Manager is able to manage different kinds of plug-ins, including:

- Data-manipulation plug-ins
- Metadata management and production
- Workflow plug-ins
- Protection plug-ins

As shown in the next figure, the AXMEDIS Plug-In Manager is accessed via the AXOM. Thus, the newly integrated functionalities are available within the different AXMEDIS Tools. For the management of the plug-ins, a description of the functionality has to be provided. This is the so-called algorithm profile.



The Plug-in Manager provides standard interface for the integration of the algorithms, which includes a plug-in profile. The plug-in profile describes the available functionality. Each plug-in is described by an XML file which contains the following information:

- the category of the plug-in, e.g. content processing;
- the unique identifier of the plug-ins, e.g. an URI like a XML namespace;
- the signature of the plug-in evaluated by an AXCS;
- data specific for the kind of plug-in (see below);
- the signature, estimated by the AXCS, of the whole XML file.

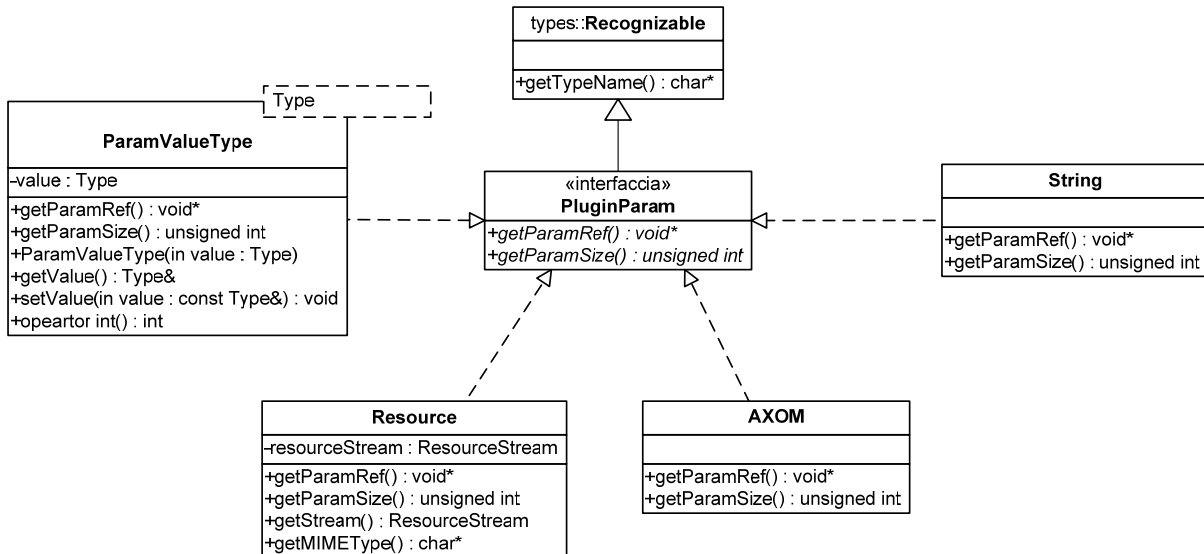
The profile provides a placeholder for custom information which can vary on the basis of plug-in category. The definition of general plug-in profile schema is reported in section 29 of DE3.1.2.2.4. The category-dependant descriptions of the plug-ins are reported in section 30 of DE3.1.2.2.4 (Content Processing) and in section 20 of DE3.1.2.2.3 (Protection Tool).

6.3 AXMEDIS Plug in development

In order to develop a plug-in, a developer has simply to create dynamic libraries which respect given interfaces (see section 6.5). These dynamic libraries are container of classes which implement given interface (which interface has to be implemented depend on the category of the plug-in). A library simply acts as factory for the implemented classes. Once created the dynamic libraries classes the developer can easily creates the profile of the plug-in on the basis of the implemented functionalities. The profiles have to respect the schemas described in the specification.

As stated in the description schema, for each parameter type a corresponding class is defined. This defined class has to be used in the programming language. Those classes derive from a common base class (**PluginParam**), which provides a basic interface for all possible parameter types. Moreover, PluginParam

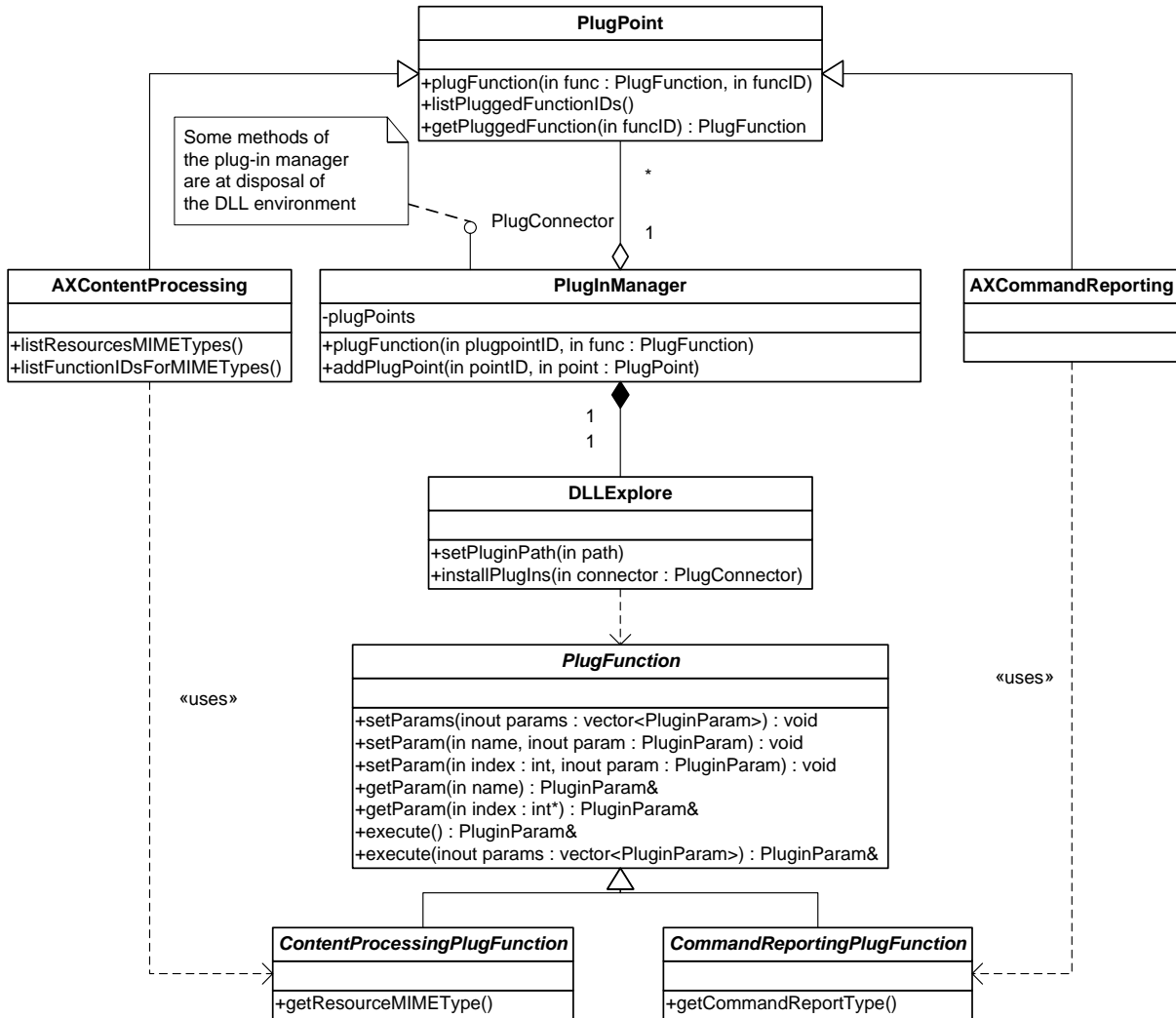
derives from Recognizable which is an interface which allows to perform type-related operations on the objects (e.g. something like reflection in Java and C#), for more details see DE-3-1-2, Part A in the Document Model Support section.



The associations among class and parameter types is depicted below:

- **ParamValueType** – it is a template class which will be used, through a set of *typedef* definition, to wrap the basic data type, that is: UNIT16, INT16, UINT32, INT32, CHAR, BOOLEAN, FLOAT, DOUBLE.
This class exposes a constructor and a cast operator which allow to freely use its instances in the same way of base type.
- **String** – it represents a parameter of type STRING
- **Resource** – it represents a RESOURCE. It exposes two meaningful functions: **getMimeType**, **getStream**. The former allow to know the MIME type of the resource contained in it, the latter returns a stream which allow to get the resource wherever it is physically located.
- **AXOM** – it represents a parameter of type AXOM.

The base class exposes two generic function **getParamRef** and **getParamSize** which return a reference to the real parameter (as void pointer) and the size in byte of the parameter itself respectively.



In the above diagram the plug-in manager structure is depicted. That diagram provides base classes which allows to construct more complicated architecture. In particular, fundamental element of that approach are the following:

- **PluginManager** – it is the main class of the module which links the plug-in (and their functionalities) to those component which will use them. It contains a list of **PlugPoint**, each of them is registered to a specific category of plug-in and function. Every time **PluginManager** loads a plug-in and its functions, it alerts the related **PlugPoint**.
- **DLLExplorer** – it is an utility class which is used by the **PluginManager** to discover all the plug-in installed on a device. Every time it found a dynamic library with its associated profile it invokes the **PluginManager**.
- **PlugFunction** – it is a base class for all those classes which have to be used as plug-in function. The **execute** method is pure virtual and it have to be defined in the derived classes, it is the real functional part of those classes.
- **PlugPoint** – it is a base class for all those classes which want to use plug-in functions. As depicted in the figure, PlugPoint is the base class the content processing module derives from (as well as the command and reporting module)

Plug-in certification functionalities are based on the services provided by the Protection Processor which needs to access to profile of the plug-in.

6.4 AXMEDIS Plug in Content Processing and Protection Tools

6.4.1 Content Processing

Content Processing Plug-in have to export the following functions:

```
extern "C" AxCPFunction* GetPluginFunction(std::string funcName)
extern "C" void releasePluginFunction(AxCPFunction* function)
```

`GetPluginFunction` allows creating Content Processing Function instances by providing the function name (`funcName`).

`releasePluginFunction` releases the given function instance. This function has been introduced for the function instances are created by the dynamic library and they have to be deleted by it.

`AxCPFunction` is the base class for Content Processing Function classes. It provides some default methods for setting parameters and requires to implement the execution method of the function itself. Therefore the only method which has to be implemented in order to develop a Content Processing Function is `execute`.

An example of content processing plug-in can be found in the following folders in the repository:

<https://cvs.axmedis.org/repos/Framework/source/adaptation/image/>

<https://cvs.axmedis.org/repos/Framework/bin/adaptation/image/>

The first folder contains the source code of the dynamic library. The second one contains the XML profile of the plug-in.

6.4.2 Protection Tools

Protection Tools used by the Protection Processor are distributed as AXMEDIS Plug-ins. Each plug-ins contains a set of Protection Tools. Moreover, the manifest of the plug-in contains a specific description which reports the identifiers of the provided tools and their main features.

In particular, an AXMEDIS Plug-in containing some protection tools has to export the following functions:

```
extern "C" Alg* createIPMPTool(const std::string& toolID, bool encoding)
extern "C" void releaseIPMPTool(Alg* tool)
```

`createIPMPTool` allows creating Protection Tool instances by providing the tool identifier (`toolID`). When the `encoding` parameter is `true`, the function has to return the encoding version of the tool itself. Otherwise, the function has to return the decoding part of the required tool. That is, a tool identifier identifies the couple of encoding and decoding algorithms.

`releaseIPMPTool` releases the given `tool` instance. This function has been introduced for the tool instances are created by the dynamic library and they have to be deleted by it.

`Alg` is an abstract class which is the base class of all Protection Tools. That is, each Protection Tool has to expose the interface provided by `Alg`. In particular, a given Protection Tool has to implement its own specific `BlockProcess` function which, given a block of bytes, has to produce the clear-text version of this block w.r.t. the implemented algorithm. In this way, different Protection Tools can be treated in a common manner.

An example of protection tool plug-in can be found in the following folder in the repository:

<https://cvs.axmedis.org/repos/Framework/doc/test/axom/ppplugin1/>

The folder contains the XML profile of the plug-in and the source code for the related dynamic library.

6.5 AXMEDIS Plug-in examples

This section describes the development of plug-ins for the integration of existing tools. Besides calling the functionality that is available in external libraries, system functionalities (executables) can be called as well using the system-call function available in C/C++.

6.5.1 AXMEDIS Plug-in examples: Developing a Content Processing Plug-In

This example for the integration of a content processing plug-in is based on the integration of functionality available in the ImageMagick image processing library. The corresponding source code can be found in the following folders in the repository:

<https://cvs.axmedis.org/repos/Framework/source/adaptation/image/>
<https://cvs.axmedis.org/repos/Framework/bin/adaptation/image/>

The first folder contains the source code of the dynamic library. The second one contains the XML profile of the plug-in.

As `AxCPFunction` is the base class for Content Processing Function classes, for each processing functionality a class have to be defined, which is derived from `AxCPFunction`.

In the following example, the constructor allocates memory for the input parameters for mapping (content processing function) parameter to the corresponding class parameters. .

The function `execute()` contains the content processing functionality or calls the corresponding functionality.

The following code extract shows the functionality for the function that manipulates the contrast of an image by calling the corresponding ImageMagick functionality.

```
//##### Contrast #####

class Contrast: public AxCPFunction
{
public:
    Contrast()
    {
        mapParameters["InputResource"] = new
AxCPPParameterResource(NULL, NULL, "");
        mapParameters["AMOUNT"] = new AxCPPParameterINT32();
        mapParameters["OutputResource"] = new
AxCPPParameterResource(NULL, NULL, "");
    }
    AxCPPParameter* execute()
    {
        AxCPPParameter * result = new AxCPPParameterSTRING();
        AxCPPParameterResource* parInputResource =
dynamic_cast<AxCPPParameterResource*>(mapParameters["InputResource"]);
        AxCPPParameterResource* parOutputResource =
dynamic_cast<AxCPPParameterResource*>(mapParameters["OutputResource"]);
        AxCPPParameterINT32* paramAmount =
dynamic_cast<AxCPPParameterINT32*>(mapParameters["AMOUNT"]);
        std::istream& inputStream = parInputResource->getIstream();
        std::ostream& outputStream = parOutputResource->getOstream();
        unsigned int amount = paramAmount->getValue();
        std::string mimeType=parInputResource->getMimeType();
    }
};
```

```

        AxImage image;
        if (image.loadFrom(inputStream, mimeType))
        {
            image.Contrast(amount);

            image.writeTo(outputStream, mimeType);

            parOutputResource->setMimeType(mimeType);

            result->setStrValue("SUCCESS");
        }
        else
            result->setStrValue("ERROR - cannot load file");
        return result;
    }
};

```

As described in the previous section the following DLL - functions have to be implemented:

```

extern "C" AxCPFunction* GetPluginFunction(std::string funcName)
extern "C" void releasePluginFunction(AxCPFunction* function)

```

The function `createIPMPTool` creates the required objects for the content processing functionality, The string parameter `toolID` identifies the selected algorithm (if multiple algorithms are available).

The following code extract exemplifies the implementation with the functions `Conversion`, `Import`, `Resize`, `Contrast`, and `Test`. It starts with the function header and some other functionalities.

```

extern "C" __declspec(dllexport) AxCPFunction * GetPluginFunction(string
functionName)
{
    AxCPFunction* Inst;
    if (functionName=="Conversion")
        Inst = new Conversion();
    else if(functionName=="Import")
        Inst = new Import();
    else if(functionName=="Resize")
        Inst = new Resize();

```

This is code extract responsible for the management of the contrast function.

```

        else if(functionName=="Contrast")
            Inst = new Contrast();

```

The plug-in must as well verify that a correct parameter for the requested functionality is given.

```

        else if(functionName=="Test")
            Inst = new Test();
        else
            throw std::runtime_error(" REQUESTED FUNCTION IS NOT INTO PLUGIN ");

        return Inst;
    }
}

```

For completeness, the corresponding objects have to be freed. This is done by calling the function `releasePluginFunction`.

```
extern "C" __declspec(dllexport) void releasePluginFunction(AxCPFFunction *
pluginFunction)
{
    delete pluginFunction;
}
```

To finalize the plug-in development, an XML-description of the available functionality has to be described.

The corresponding XML file for the above described functionality and an additional dummy function is given below. It starts with the general header for each content processing plug-ing.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)-->
<pin:Plugin xmlns:fd="http://www.axmedis.org/plugin-function-schema"
xmlns:pin="http://www.axmedis.org/plugin-schema"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.axmedis.org/plugin-function-schema plugin-
function-schema.xsd http://www.axmedis.org/plugin-schema plugin-schema.xsd">
  <pin:GeneralDescriptor>
    <pin:Category>ContentProcessing</pin:Category>
    <pin:Identifier>ImageProcessing</pin:Identifier>
    <pin:Library>Adaptation</pin:Library>
    <pin:Version>1.001</pin:Version>
    <pin:Vendor>Axmedis</pin:Vendor>
    <pin:MainLibrary>imageprocessingplugin.dll</pin:MainLibrary>
    <pin:Description>Plugin for image processing</pin:Description>
  </pin:GeneralDescriptor>
  <pin:ComponentsSignature>
    <dsig:SignedInfo>
      <dsig:CanonicalizationMethod
Algorithm=" " ></dsig:CanonicalizationMethod>
      <dsig:SignatureMethod Algorithm=" " ></dsig:SignatureMethod>
      <dsig:Reference>
        <dsig:DigestMethod Algorithm=" " ></dsig:DigestMethod>
        <dsig:DigestValue>AA==</dsig:DigestValue>
      </dsig:Reference>
    </dsig:SignedInfo>
    <dsig:SignatureValue
xmlns="http://www.w3.org/2000/09/xmldsig#" >AA==</dsig:SignatureValue>
  </pin:ComponentsSignature>
  <fd:FunctionList>
```

Below, the description of the available functionality is given.

```
    <fd:Function>
      <fd:Name>Contrast</fd:Name>
      <fd:Version>1.0</fd:Version>
      <fd:FunctionDescription>Change image
contrast</fd:FunctionDescription>
      <fd:ParameterList>
        <fd:Param>
          <fd:Name>InputResource</fd:Name>
          <fd:ParamType>RESOURCE</fd:ParamType>
```

```

        <fd:In/>
        <fd:Mandatory/>
        <fd:ParamDescription>The Resource to be
manipulated</fd:ParamDescription>
        <fd:Constraints>
            <fd:Resource>

                <fd:ResourceType>image</fd:ResourceType>

                <fd:ResourceFormat>jpeg</fd:ResourceFormat>

                <fd:ResourceFormat>gif</fd:ResourceFormat>

                <fd:ResourceFormat>png</fd:ResourceFormat>
            </fd:Resource>
        </fd:Constraints>
    </fd:Param>
    <fd:Param>
        <fd:Name>AMOUNT</fd:Name>
        <fd:ParamType>INT32</fd:ParamType>
        <fd:In/>
        <fd:Mandatory/>
        <fd:ParamDescription>The contrast
amount</fd:ParamDescription>
    </fd:Param>
    <fd:Param>
        <fd:Name>OutputResource</fd:Name>
        <fd:ParamType>RESOURCE</fd:ParamType>
        <fd:Out/>
        <fd:Mandatory/>
        <fd:ParamDescription>Where the manipulated
resource will be stored</fd:ParamDescription>
    </fd:Param>
</fd:ParameterList>
<fd:Result>
    <fd:Name>Result</fd:Name>
    <fd:ResultType>STRING</fd:ResultType>
    <fd:ResultDescription>The result of conversion, SUCCESS
if ok, ERROR followed by a message in case of error</fd:ResultDescription>
</fd:Result>
</fd:Function>

```

The missing plug-in XML - description for the final description is shown below:

```

</fd:FunctionList>
    <pin:Signature>
        <dsig:SignedInfo>
            <dsig:CanonicalizationMethod
Algorithm=""></dsig:CanonicalizationMethod>
            <dsig:SignatureMethod Algorithm=""></dsig:SignatureMethod>
            <dsig:Reference>
                <dsig:DigestMethod Algorithm=""></dsig:DigestMethod>
                <dsig:DigestValue>AA==</dsig:DigestValue>
            </dsig:Reference>
        </dsig:SignedInfo>
        <dsig:SignatureValue>AA==</dsig:SignatureValue>
    </pin:Signature>
</pin:Plugin>

```

6.5.2 AXMEDIS Plug-in examples: Developing a Protection Tool Plug-In

This example for the integration of a protection tool plug-in is based on OpenSSL. An example of protection tool plug-in can be found in the following folder in the repository:

<https://cvs.axmedis.org/repos/Framework/doc/test/axom/ppplugin1/>

The folder contains the XML profile of the plug-in and the source code for the related dynamic library.

As described in the previous section, `Alg` is an abstract class which is the base class of all Protection Tools. Therefore, all protection functionality has to be derived from the base class `Alg`. In the following example, the class `AlgBlowfish` integrates OpenSSL functionality – namely the Blowfish encryption/decryption functionality and is derived from `Alg`.

```
class AlgBlowfish : public Alg
{
private:
    unsigned long int i;
    unsigned char Key[56];
    unsigned char IV[8];
    int mode;
    BF_KEY key;
    int num;

public:
    AlgBlowfish(char *,int);
    ~AlgBlowfish();
    virtual unsigned long int BlockProcess(unsigned long int);
};
```

The `BlockProcess` function has to contain the encryption/decryption functionality.

```
unsigned long int AlgBlowfish::BlockProcess(unsigned long int howmany)
{
    BF_cfb64_encrypt(InBuffer, OutBuffer, howmany, &key, IV, &num, mode);
    state = DONE;
    return howmany;
}
```

In the case of the available blowfish implementation, the parameter `mode` determines about encryption or decryption. This parameter as well as the other parameters (like the key) has to be set in the constructor.

In the simple example `AlgDEC` respectively `AlgINC`, different functions are implemented for decreasing respectively increasing individual characters. Therefore, two different classes are implement. Due to the symmetric the following example only shows `AlgDEC`.

```
class AlgDEC : public Alg
{
public:
    AlgDEC();
    ~AlgDEC();
    virtual unsigned long int BlockProcess(unsigned long int);
};
```


The corresponding `BlockProcess` function simply implements the required functionality:

```
unsigned long int AlgDEC::BlockProcess(unsigned long int howmany)
{
    unsigned long int i;

    for(i=0;i<howmany;i++) OutBuffer[i] = InBuffer[i] - 1;
    state = DONE;
    return howmany;
}
```

As described in the previous section the following DLL - functions have to be implemented:

```
extern "C" Alg* createIPMPTool(const std::string& toolID, bool encoding)
extern "C" void releaseIPMPTool(Alg* tool)
```

The function `createIPMPTool` creates the required objects for encryption and decryption of the content. Parameters are:

- The string `toolID` that identifies the selected algorithm (if multiple algorithms are available).
- The boolean variable `encoding` selects encryption respectively decryption.

The code example for the previously outlined algorithms `AlgBlowfish`, `AlgDEC` and `AlgINC` is shown below.

```
extern "C" __declspec(dllexport) Alg* createIPMPTool(const std::string& toolID,
                                                    bool encoding)
{
    if(toolID=="urn:axmedis:ipmp:tool:id:0001")
    {
        char data1[] =
"qwertyuiopqwertyuiopqwertyuiopqwertyuiopqwertyuiopqwertyuiop1234";
        return new AlgBlowfish(data1,encoding?0:1);
    }
    else if(toolID=="urn:axmedis:ipmp:tool:id:0002")
        return new AlgNull();
    else if(toolID=="urn:axmedis:ipmp:tool:id:0003")
    {
        if(encoding)
            return new AlgINC();
        else
            return new AlgDEC();
    }
    return NULL;
}
```

For completeness, the corresponding objects have to be freed. This is done by calling the function `releaseIPMPTool`.

```
extern "C" __declspec(dllexport) void releaseIPMPTool(Alg* tool)
{
    delete tool;
}
```

To finalize the plug-in development, an XML-description of the available functionality has to be described.

The corresponding XML file for the above described functionality and an additional dummy function is given below:

```
<?xml version="1.0" encoding="utf-8"?>
<pin:Plugin xmlns:pp="http://www.axmedis.org/ipmp-tool-schema"
xmlns:pin="http://www.axmedis.org/plugin-schema"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ipmpinfo="urn:mpeg:mpeg21:2004:01-IPMPINFO-NS"
xmlns:prm="http://www.axmedis.org/parameter"
xsi:schemaLocation="http://www.axmedis.org/ipmp-tool-schema ipmp-tool-schema.xsd
http://www.axmedis.org/plugin-schema plugin-schema.xsd urn:mpeg:mpeg21:2004:01-
IPMPINFO-NS ipmpinfo.xsd">
  <pin:GeneralDescriptor>
    <pin:Category>IPMPTool</pin:Category>
    <pin:Identifier>ppplugin1</pin:Identifier>
    <pin:Library>Vallotti's protection processor plugin 1</pin:Library>
    <pin:Version>0.0.1</pin:Version>
    <pin:Vendor>Axmedis</pin:Vendor>
    <pin:MainLibrary>ppplugin1.dll</pin:MainLibrary>
    <pin:Description>Provides some ipmp tools</pin:Description>
  </pin:GeneralDescriptor>
  <pin:ComponentsSignature>
    <dsig:SignedInfo>
      <dsig:CanonicalizationMethod Algorithm=""/>
      <dsig:SignatureMethod Algorithm=""/>
      <dsig:Reference>
        <dsig:DigestMethod Algorithm=""/>
        <dsig:DigestValue>AA==</dsig:DigestValue>
      </dsig:Reference>
    </dsig:SignedInfo>
    <dsig:SignatureValue
xmlns="http://www.w3.org/2000/09/xmldsig#">AA==</dsig:SignatureValue>
  </pin:ComponentsSignature>
  <pp:IPMPToolList>
    <pp:Tool pp:type="authoring">
      <ipmpinfo:IPMPToolID>urn:axmedis:ipmp:tool:id:0001</ipmpinfo:IPMPToolID>
      <pp:Name>Blowfish</pp:Name>
      <pp:Description>It perform encryption and decryption using
Blowfish</pp:Description>
    </pp:Tool>
    <pp:Tool pp:type="playing">
      <ipmpinfo:IPMPToolID>urn:axmedis:ipmp:tool:id:0002</ipmpinfo:IPMPToolID>
      <pp:Name>Null</pp:Name>
      <pp:Description>It makes nothing ;-)</pp:Description>
    </pp:Tool>
    <pp:Tool pp:type="authoring">
      <ipmpinfo:IPMPToolID>urn:axmedis:ipmp:tool:id:0003</ipmpinfo:IPMPToolID>
      <pp:Name>Caesar cipher</pp:Name>
      <pp:Description>It perform encryption and decryption using
Caesar cipher</pp:Description>
    </pp:Tool>
  </pp:IPMPToolList>
</pin:Signature>
```

```

    <dsig:SignedInfo>
      <dsig:CanonicalizationMethod Algorithm="" />
      <dsig:SignatureMethod Algorithm="" />
      <dsig:Reference>
        <dsig:DigestMethod Algorithm="" />
        <dsig:DigestValue>AA==</dsig:DigestValue>
      </dsig:Reference>
    </dsig:SignedInfo>
    <dsig:SignatureValue>AA==</dsig:SignatureValue>
  </pin:Signature>
</pin:Plugin>

```

6.6 AXMEDIS Plug in Content Processing and Protection Tools

This section provides a short overview of the existing functionality that is integrated via the plug-in interface. Technical details can be found in DE-3-1-2-2-7 Specification of AXMEDIS - External Processing Algorithms.

6.6.1 Adaptation Tools and Algorithms for Text

Document adaptation tools provide functions which can convert a text document file modifying its format.

6.6.2 Adaptation Tools and Algorithms for Video

The adaptation tools and algorithms for video provide the basic functions that are needed to convert video files. The module, which is initially provided, is based on the FFmpeg library (<http://ffmpeg.sourceforge.net/index.php>).

FFmpeg is “a set of free computer programs that can record, convert and stream digital audio and video. It includes libavcodec, a leading audio/video codec library. FFmpeg is developed under Linux, but it can compile under most operating systems, including Windows ... Notable is that most FFmpeg developers are part of either the MPlayer, xine or VideoLAN project as well.” (see <http://en.wikipedia.org/wiki/FFmpeg>)

6.6.3 Adaptation Tools and Algorithms for Images

This module gives the possibility to use algorithms and tools for adaptation of images. The main adaptation functions needed by the AXMEDIS Framework could be summarised in:

- Scaling
- Resolution improvement/reduction
- Colour to Greyscale conversions
- Format transcoding
- Composition with other images
- Adding widgets and graphic motifs
- Text drawing
- Image decomposition

These functions are implemented by defining specific algorithms or using graphic libraries. An example of library is given by the *ImageMagick Library*.

6.6.4 Adaptation Tools and Algorithms for Audio

The FFmpeg Audio Transcoding function can be used to convert an audio file into a different format and/or codec (please refer to section 4.4.10 for a complete description of formats and codecs supported in decoding and encoding). Apart from the bit rate reduction depending on the selected codec, one can further reduce the size of the resulting audio file by changing its sample rate and its number of audio channels. Moreover one can select only a specific portion of the input file to produce the resulting output file by specifying starting and ending points in the input file.

The libsndfile Audio Transcoding function can be used to convert an audio file into a different format and/or codec (please refer to section 4.4.10 for a complete description of formats and codecs supported in decoding and encoding). Apart from the bit rate reduction depending on the selected codec, one can further reduce the size of the resulting audio file by changing its sample rate and its number of audio channels. Moreover one can select only a specific portion of the input file to produce the resulting output file by specifying starting and ending points in the input file.

6.6.5 Adaptation Tools and Algorithms for Multimedia

Text and images have long been the main resources for Web content. Yet, as new formats emerge, rich multimedia presentations are making their entrance into the Web world and are being increasingly used for newscasts, educational material, entertainment etc. At the same time, the 3rd Generation Partnership Project and the 3rd Generation Partnership Project 2 seek to provide uniform delivery of rich multimedia over newly evolved, broadband mobile networks (3rd generation networks) to the latest enabled cell phones.

By rich media, we refer here to a broad range of digital media choreographing audio, video, text, graphics and synthetic animations in real time. Another key feature of such media is interactivity. Rich media may indeed respond directly to user interactions allowing for new ways of consuming media. For example, a pre-recorded webcast may be coupled with a synchronized slide show that allows user interaction with the mouse.

In the recent years, a large number of new multimedia formats have been created independently from each other including among others Quicktime, RealVideo, Advanced Streaming Format, Shockwave, SMIL and MPEG-4 BIFS. AXMEDIS Multimedia Adaptation tools focus on the last two since they are non-proprietary and support a large variety of usage scenario.

To adapt multimedia files two goals are considered: direct transcoding of a multimedia file towards another multimedia file format and adaptation of the simpler media files embedded into the multimedia file. The later goal asks for means to extract simple media files from multimedia files and means to reintegrate adapted media files into the richer multimedia file. More specifically, the multimedia adaptation tool should provide:

- Transcoding functions: SMIL to MP4, MP4 to ISMA compliant MP4, MP4 to 3GP compliant MP4, SWF to MP4.
- Extraction functions: extraction of media files from richer media files (for example, extracting a video file from a multimedia file to apply specific video adaptation functions).
- Embedding functions: embedding of media files into richer media files (for example, adding an MP3 file into a MP4 file which already contains a video track and a subtitle track).

6.6.6 Adaptation Tools and Algorithms for Metadata

The MetaDataMapper library is used by the metadata mapper editor and the metadata adaption tools to create mapping information between source and target XML metadata documents and to convert a source XML metadata document to a target metadata according to existing mapping information.

When used by the metadata mapper editor, the metadatamapper library builds an XSLT document from the mapping information provided by the GUI

The Mapper uses the Apache Xerces-C parser to parse the input XML documents and build an XSLT document containing mapping information. All libraries used are documented in the Used Libraries section. Xalan is used to transform metadata documents according to an XSLT document.

6.6.7 Adaptation Tools and Algorithms for DRM information

DRM adaptation involves the adaptation of the related licenses, as derived AXMEDIS objects or digital resources within the AXMEDIS objects can be seen as new creations with regard to original ones. Therefore,

new licenses must be created during the adaptation process, always respecting the terms and conditions fixed in the original license or licenses for the adapted AXMEDIS objects or contents within these objects.

Nevertheless, DRM information (mainly licenses and PARs) inside the AXMEDIS project, that are related to AXMEDIS Objects will be expressed in XML language by using MPEG-21 REL.

In order to adapt this information to different rights expression languages, also based in XML or to adapt a license to be more compact in order to use it into portable devices (for instance, mobile phones or PDAs), we will make use of existing libraries for manipulating XML documents.

The main adaptation function produced by this module can be summarised in:

- Compacting licenses for their use in portable devices
- Translating licenses from one rights expression language to another
- Automatic generation of a license when an adaptation over the content it applies is done

For XML DRM rules transcoding, the *Xerces Libraries* can be used to parse a given piece of XML data.

At the present moment, the adaptations foreseen regarding the current state of the art are between MPEG-21 REL and OMA DRM REL (which is based on ODRL) and MPEG-21 REL profiles. Evolution in the state of the art may involve more adaptations available.

6.6.8 Adaptation Tools and Algorithms for Ringtones

Ringtone Adaptation refers to the adaptation of ringtones of popular formats to enhance usability and manage the variable delivery to cater for heterogeneous client devices and user requirements on-demand. It uses external libraries like FFMPEG, LIBSNDFILE and TIMIDITY to convert the ring tones from one format to another and to resample it based on the client device.

6.6.9 Descriptor extractor as fingerprint for Text files

Automatic keyword extraction aims to provide the user with a set of descriptors which represent the contents of the document, and which are further exploited to perform advanced searches within the textual repository. Extracted keywords can also be used to identify higher levels of descriptors, such as the document domain.

The procedure is divided into three main steps:

- 1) **Comparative frequency analysis: mono-term keywords extraction:** This process is performed through an integration of resources and standard algorithms, by means of comparison between the document and the referring universe, represented by a general corpus. The output is a list of nouns in the document, ordered by TF.IDF scores
- 2) **Semantic analysis: mono-term keywords semantic disambiguation and domain detection:** Since nouns are potentially ambiguous with respect with their semantics, a word sense disambiguation (WSD) procedure is run over the output of the previous step. This would allow, in principle, the translatability of keywords in different languages. WordNet Domains database is also exploited to determine the “area of discussion” to which each keyword belongs, so providing other keys for content identification.
- 3) **Internal analysis of lexical associations: multi-term keywords detection:** The extracted keywords are further refined from the point of view of the accuracy, of the content identification, and of the value of the descriptors, referring to language properties of word association: high frequency collocations within the text are considered more definite and highly representative of its content.

6.6.10 Descriptor extractor as fingerprint for Audio files

With electronic music distribution (EMD), music catalogues have become huge. The biggest online services now propose around 2 million tracks while personal users have the possibility to carry thousand of songs on their portable music player. In fact, the amount of digital music is now urging for reliable and fast tools for content analysis and description, to be used for searches, content queries and interactive access. In the

context of the AXMEDIS project, a number of algorithms for audio content analysis and description have been developed and implemented to ease audio content retrieval and browsing into collections.

6.6.11 Descriptor extractor as fingerprint for Video Files

Within the AXMEDIS project no research on the content descriptors for video is performed. Instead, the extensibility of the AXMEDIS plug-in interface is exploited and validated by integrating existing state-of-the-art algorithms.

The MPEG-7 eXperimental Model (XM) is the MPEG-7 Reference Software and for example (for German people) it is available for downloading at <http://www.lis.e-technik.tu-muenchen.de/research/bv/topics/mmdb/mpeg7.html> and via CVS.¹ The MPEG-7 XM Reference software includes state-of-the-art content description algorithms for audio-visual content.

Some of the available functionality was selected based on the identified user requirements.

6.6.12 Fingerprint Estimation for Text files

The text fingerprint plug-in output will be a string in which the value is stored. Moreover the document comparison functionality also implemented in the plug-in will give as a result a normalized floating point value representing the degree of similarity between two given input documents.

The text fingerprints plug-in aim is twofold, it provides a way of calculating a fingerprint value of the documents provided as input, moreover it provides functionalities for similarity estimation between two documents without making prior assumptions on the language. The fingerprint algorithm hashes the ASCII representation of the input file and gives as result a string in which the fingerprint value is stored (full or selective hash values based on the analyzed document structure).

For similarity comparison a plug-in specific function is provided. This functionality could be exploited by several use cases including: identification of content from a sub part or when the different formats comparison is not straightforward, plagiarism detection and so on. The algorithm allows for robust multilevel comparison of documents taking into account document structure and leveraging the plagiarist behaviour, which is modeled as a combination of 3 basic actions: insertion, deletion, substitution. We recognize that this behavior may occur at various level of the document structure: the plagiarist may insert, delete or substitute a word, period or a paragraph. The procedure consists in two main steps: document structure extraction and plagiarism function calculation. We propose a recursive plagiarism evaluation function to be evaluated at each level of the document structure which is based on the Levenshtein edit distance.

For what concerns fingerprint it has to be said that in MPEG-7 this kind of meta-information (for verification of the text documents) is not considered so far. Within AXMEDIS it has to be identified, how cryptographic hash values can be integrated best into MPEG-7 and the relationship with MPEG-21.

6.6.13 Fingerprint Estimation for Audio files

The Audio fingerprint plug-in is a tool that extracts an audio fingerprint of a given audio stream within a multimedia file. The audio stream can be embedded either in a normal audio file (mpg, wav, wma, etc...) or within a video file (mpeg, wmv, avi, etc...).

The Fingerprint extractors for audio files automatically calculate a digest describing its main characteristics in way suitable for automatic verification of AXMEDIS objects. Thus the descriptor is a low level description according to the previous definition.

¹ Access to the CVS is only possible through registration at the national standardization body, in each country there is a different standardization body connected to ISO and MPEG.

Audio files are streams. These input stream is segmented. That means that for each segment a “sub-fingerprint” is calculated. Depending on the length of the input sequence, typically a request to the database do not result in a single fingerprint but in an array of fingerprints.

6.6.14 Fingerprint Estimation for Video files

Video files are related to audio as video also is time dependent: They are streams. The input stream is segmented. For each segment a “sub-fingerprint” is calculated. Depending on the length of the input sequence, typically a request to the database do not result in a single fingerprint but in an array of fingerprints.

6.6.15 Fingerprint Estimation for Metadata

For the verification of the objects metadata only cryptographic hash functions are feasible. This hash function is applied to all kinds of meta-data of an AXMEDIS object.

6.6.16 Fingerprint Estimation for Generic Files

For the verification of the objects metadata only cryptographic hash functions are feasible. This hash function is applied to all kinds of meta-data of an AXMEDIS object.

6.6.17 External Protection Libraries

The information age has seen the development of electronic pathways that carry vast amounts of valuable commercial content between individuals and companies. Unfortunately the unprecedented levels of access provided by systems like the Internet also expose this data to breaches of confidentiality, disruption of service, and copyrights infringements. For this reason, in the content distribution field, many applications use more and more DRM (Digital rights Managements) solutions.

This means that content use/manage applications need security module implemented within. Unfortunately the security systems required to protect data are generally extremely difficult to design and implement, and even when available tend to require considerable understanding of the underlying principles in order to be used. This has lead to a proliferation of “snake oil” products that offer only illusionary security, or to organizations holding back from deploying online information systems because the means to secure them are not readily available, or because they employed weak, easily broken security that was unacceptable to users.

The cryptlib security library provides a complete set of cryptographic algorithms that fit the Axmedis needs. The following subsections describe the external library that may be used in the AXMEDIS framework to implement the needed cryptography functionalities.

7 Guidelines on AXMEDIS Component Validation and Acceptance (FUPF)

In order to correctly set up the AXMEDIS framework, each implemented and supplied component should be validated and certified before allowing its use by content creators, content providers and final users inside AXMEDIS. This will help in guaranteeing that the implemented component meets the requirements.

The validation will depend on several issues and will seek to establish the degree to which a given component fulfils the following performance criteria:

- It does what it is supposed to do
- It does its job in a reasonable time
- It can be integrated with other components
- It is reliable
- It is robust
- It accomplishes test cases
- It is conformant with the AXMEDIS framework

In the following sections we will describe the general guidelines for component validation and acceptance in the AXMEDIS framework.

7.1 Component validation

The component validation process should involve the checking and analysis of the component in order to verify that the component meets the requirements demanded of it.

The main validation activities are the revision and testing of the component.

Review phase main steps:

- Review phase involves the manual review of the component, directly evaluating it. It will help in determining that the requirements, design concepts and specifications have been met. The revision of a component can include several activities, like peer reviewing or code inspection.
- The result of the review should be a report where the reviewing team explains the revision performed, the errors found and other.

Testing phase involves the testing of the component at different levels, from unit test through to system test.

The main steps in the testing phase are the following ones:

- During development phase of the component, unit tests should be done in order to check that the functionality being implemented is the correct one.
 - This task should be done by the development team of the component, as they have the complete knowledge over it.
 - Modifications over the specification should be reported
- When the development of a component has been completed, an initial unit testing phase and an integration test is needed to evaluate how the unit performs and how it interacts with other units.
 - This task should be done with the cooperation of the development teams of the integrated components. The result of this task should be reported in order to perform the needed actions for solving the problems found.
 - This task also needs to evaluate the following aspects, and report on them:
 - Documentation provided

- Differences with the specification, if any
 - Interfaces among components
- The rest of tests should involve the whole system and it will be different depending on the demonstrator. So system testing should be demonstrator testing for the AXMEDIS framework, at least in a first phase.

In order to perform the above tests, the test cases identified and described in WP2 and the content for test cases created and described in WP8 should be taken into account.

Apart from tests and reviews, component validation should also include the examination of the degree to which the component is properly documented, including updates in the specification documents, documentation of the programming interface, usage manual (specially for end-user or business-user applications) or installation manual. This information should be included in the AXMEDIS framework part associated to this component.

The major requirements and functionalities of the component need to be mapped to the component under validation, in order to check that the component accomplishes them.

7.2 Component Submission

This section describes what has to be reported when a component is submitted into the AXMEDIS framework.

Module name	Name of the module being reviewed.
Module description	General purpose of the module.
Major requirements	To be provided by the component owner
Major related components	To be provided by the component owner
List of Use Cases	To be provided by the component owner
List of Test Cases	<ul style="list-style-type: none"> • To be provided by the component owner • Accepted by the validators • Additional test cases may be provided.
Used Libraries	To be provided by the component owner Versions and library related information should be provided
Languages	Programming languages
Operating system(s)	A list and information to compile for different OS
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.

7.3 Review and Acceptance report form

This section gives an example of what has to be reported when a module is being reviewed. The main information to be reported is described in the following table.

Review ID	Identifier of the review.
Module names	Name of the modules being reviewed.
Integration description	General purpose of the integration test review.
List of use cases	Related Use Cases
List of Test Cases	Related Test Cases
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.
Participants	Names, organisations and roles of people involved in the review.

ID	Who	Date	Issue location and description
Issue number	Name and organisation of the person who finds the issue	Date when the issue is found	Part of the module where the issue is found and description of the issues found in the module during revision and. The location could be the source code file, web page, etc; it will depend on the type of application.

7.4 Verification report form

This section gives an example of what has to be reported when a module is verified, after it has been reviewed and some issues have been found. The information to be reported is described in the following table.

Verification ID	Identifier of the verification.
Review ID	Identifier of the review to which this verification refers.
Module names	Name of the modules being verified.
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.

ID	Who	Date	Status	Issue description and resolution
Related Review Issue number	Name and organisation of the person who verifies the issue	Date when the verification is performed	Solved, Proposed or Not Solved	How the issue found during the revision process has to be solved or has been solved

Regarding the Status column, it has to be filled in the following manner:

- Solved: when the issue in the review report has been solved
- Proposed: when the issue in the review report has been proposed but not solved
- Not solved: when the issue in the review report has neither been proposed nor solved

7.5 Component acceptance

Component acceptance mostly involves those components, which are addressed to application users, either business or final users.

Acceptance testing allows users (or project partners which can assume this role), to test the functionality of the system against the requirements and use cases. Each kind of tool should be tested by a key user on this area. For instance, the component acceptance test of a content production tool should be done by a user that is skilled in the area together with part of the development team, in order to get the comments on the tool behaviour.

It will be very useful to follow the test cases that involve components to be accepted in order to check if they are correct. A report on if test cases are accomplished or not should be done by test participants.

7.6 Start up of component validation and acceptance

In order to start up the component validation and acceptance in the AXMEDIS framework, the responsible of each module should:

- Follow the CVS repository guidelines

- Prepare unit tests
- Perform unit tests and give a brief report on them
- Prepare integration tests
- Perform integration tests. They may be done by the partners participating in the integration together or not. In the latter case, some communication mechanisms will be established in order to solve possible integration problems as soon as possible. For instance, if one partner is performing an integration test of one of his components with other partner's component, a videoconference or audio-conference might be set up before, during or after tests are done
- Give a brief report on the integration test, outlining problems found, categorising them (application error, bad parameter value, inconsistent API, etc)

These reports should be available in the portal to the rest of the partners in order to improve the component knowledge and maintenance. A new directory / portal section should be created to contain them. They could also be included into the CVS repository, together with the corresponding application, web service, library, etc.

7.7 Periodic verification

During project development, components will be improved / updated to solve problems found during the different testing phases or to meet new requirements found during the acceptance tests or the evolution of commercial software products and standards affecting the AXMEDIS framework components.

In this way, there will be the need to inform the rest of the partners of any components having been updated, so that new integration tests are performed if needed.

Reports on the unit tests performed over the new or updated components have to be given together with the new version of the component.

The periodic verification steps should be as follows:

- 1 Update component into the corresponding directory of the CVS repository, indicating that it is a new version.
- 2 Send e-mail to the reflector and / or developers mailing list to inform that an update on a component has been done and including the results of the related regression test(s) when needed.
- 3 Partners using the updated component should:
 - 3.1 Perform integration tests with the new version of the component. The participation / support of the component responsible may be requested.
 - 3.2 Report errors / problems detected during the integration test, if any. The report has to be uploaded in the portal, in order that partners are informed of the results of the test.
 - 3.3 If needed, after integration errors / problems reported in 3.2 are solved, components using the initially updated component should be also updated in the CVS repository.
 - 3.4 Go to verification step 2 for the component(s) updated in step 3.3.

As a modification in one component may involve the modification of many other components, regression and integration tests should be systematically done. In some cases, it could happen that a component will no longer work with lower versions of libraries and components. This should be specified in the documentation of the component.

7.8 Acceptance testing

Acceptance of a component involves that it has previously been validated and verified, making the corresponding unit and integration tests. Once they have been passed, acceptance tests should be done. They may involve other partners, final users, user group experts, etc.

DE5.1.3.1 – AXMEDIS Framework Guidelines

Acceptance test results could be reported using the review form described in section 7.3.

Once the component has been accepted, it can be made publicly available in the AXMEDIS framework. Moreover, it may be further optimised.

8 Guidelines on AXMEDIS Framework Validation (EXITECH)

8.1 Source Code repository model

8.1.1 Premises

In order to better understand the guidelines contained in this document, it is necessary to underline the premises under which they are valid and the context in which the repository for AXMEDIS framework, web service and application, hereinafter “the CVS repository” has been born, the name derives from the definition reported in Annex I. the structure of the CVS repository is oriented at who will use the AXMEDIS framework and in particular to partners involved in WP9 demonstrators and those that will contribute to AXMEDIS with take up actions and projects. It is for them that we are mainly creating this framework.

As stated by the project manager, that requested these guidelines according to the schedule of Annex I:

- The CVS repository is not a support for development and developers, since each team in each company has to create the environment for development inside the company;
- The CVS repository is not an instrument for integration, since it has no configuration management process inside, at least now, a task in WP5 will define the guidelines for the integration;
- The CVS repository is not an instrument for automated building and continuous integration, since it has no management of automatic building;
- The CVS repository is not an instrument for automated building and continuous integration, since it has no management of automatic building and no bound on submission due to a failure in compilation;
- The CVS repository is a support for management of the project, and for the versioning of the AXFW and its detailed components, and for the creation/storage of demonstrators;
- The CVS repository has to support versioning only at predefined time intervals (that can be also a time frame of months or weeks, this will depend on the project evolution and status);
- Each contribution at the CVS repository has to be self-contained in the sense that has to provide comprehensive support for compiling, executing, deploying the AXMEDIS part submitted;
- Each contribution at the CVS repository has to be completed with documentation according what has been formally defined in these guidelines, and revision of the specification document starting from the official version;
- Each contribution to the CVS repository has to be completed with test cases and test code for allowing regression testing, a more detailed description of the activity of testing will be described in specific guidelines on testing, regression and acceptance testing.

8.1.2 Repository Structure

After all these necessary premises the guidelines for the structure and the content of the repository can be detailed:

The repository is structured in three main parts:

- Framework
- WebServices
- Application

Each part will contain a set of subdirectory for putting all the necessary files. Not all the directories are mandatory for each kind of application types as detailed in the following. In any case the following set of directories will be present under each specified parts:

- include: directory only for C/C++ application that have include files;
- source: directory containing the source code;
- doc: directory containing the documentation as specification documentation, code documentation, test documentation, dependencies and configuration documentation, other documentation. For each kind of documentation a detailed description will be given;
- lib: directory containing all the necessary libraries (lib, dll, jar, etc) needed for the AXMEDIS part to work;

- bin: directory containing the result of building process that can be a jar, an executable, a library, a war or whatever.

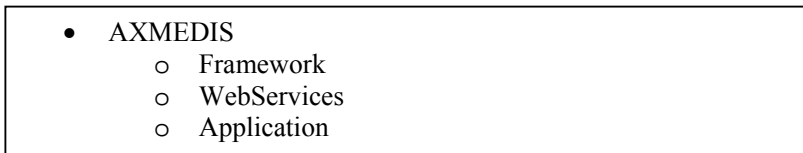
The doc directory has to be split in different directories according to the following structure:

- specification: this directory must contain the UML diagrams of all the classes and in order to avoid the problems due to incompatible format the diagrams have to be submitted in XMI format. In the same directory, if present, also the XML schemas and WSDL files have to be put. Any other format for UML with respect to XMI format will not be accepted;
- code: in this directory the documentation of code must be put. The documentation should be Javadoc, doc++ or other auto-generated documentation (preferred) or other documentation format. In any case documentation must be put in one of the following formats: HTML or TXT (preferred) and/or RTF or PDF (not preferred but acceptable). Any other proprietary format such as Microsoft Word docs, OpenOffice docs, or other will be refused;
- test: this directory must contain all the test cases necessary to test the part, the source code and building scripts for testing. Automatic testing is recommended such as JUnit (i.e. <http://www.junit.org/>) for Java, CppUnit (i.e. <http://sourceforge.net/projects/cppunit/>) for C++, PHPUnit (i.e. <http://www.phpunit.de/en/index.php>) for PHP and so on, and script for automating testing are kindly requested such as *Batch*, *Makefile*, *Ant* and so on.
- configuration-deployment
- other

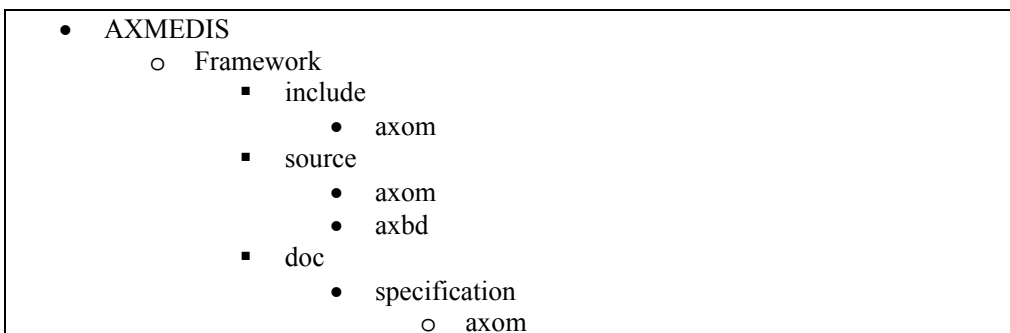
This structure has been selected:

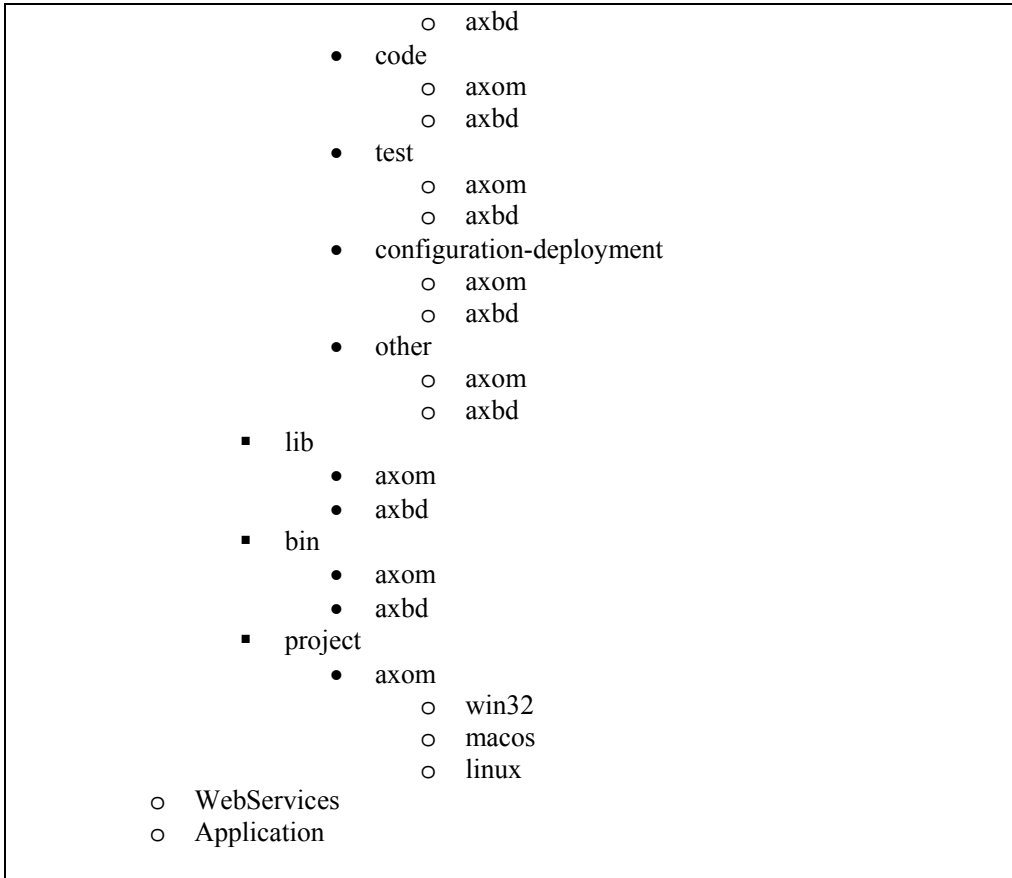
- To give the best integrated view to the future users of the AXFW. In this way we can give to them in a simple manner access to separate levels. For example, access to source code, only to applications, only to the includes, etc.
- To provide and collect the history of all the executable applications developed, this will permit to provide the history of development according to annex I of the contract and to provide demonstration without the need of recompiling and involving other partners
- To provide access in a simple manner at all the code and applications to be test for the partners that have to test and verify the tools even without the presence of who has produced the code.

Under each of the directories defined, a directory for each project part has to be created, so that the final structure will be something like:



The Framework Structure is described in the following picture:

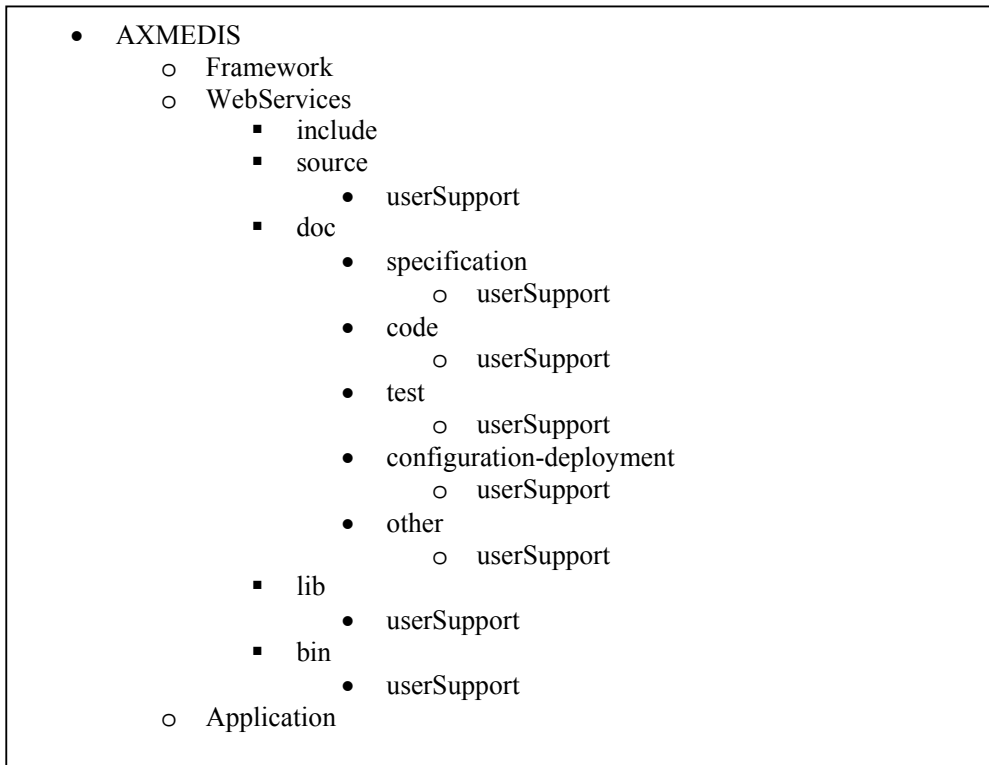




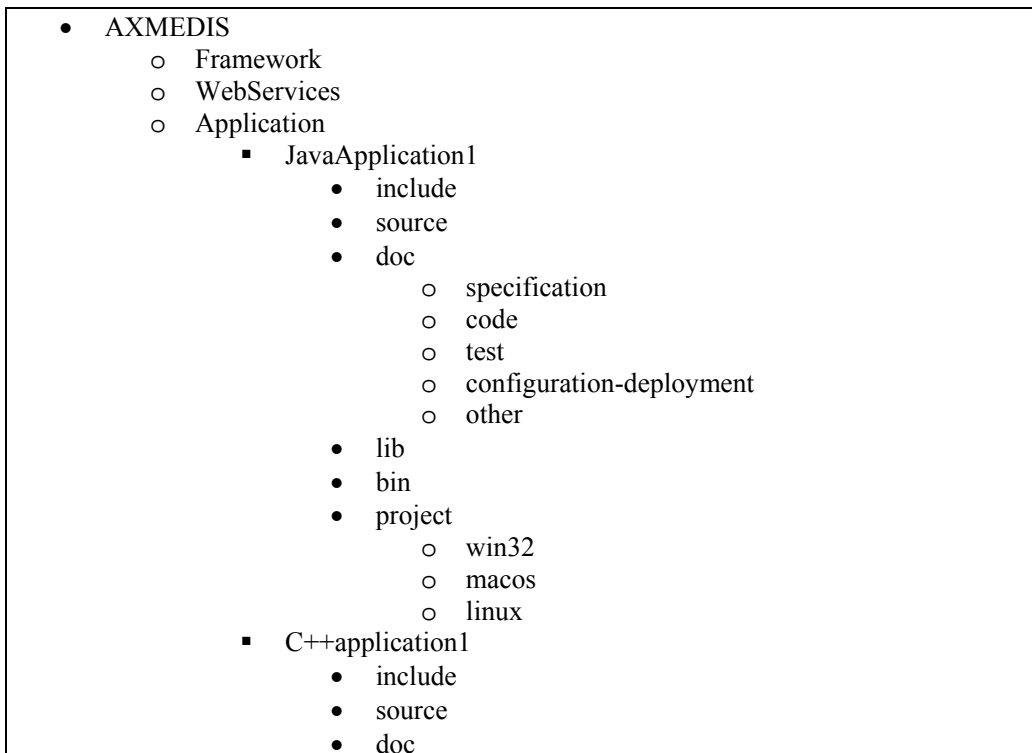
Into the “Framework” part of the repository it is supposed to have:

- object models
- loader and saver
- communication protocols
- DB access
- data transforming logic
- algorithms
- DLL
- plug ins
- libraries
- scripts
- web service support
- etc.

The WebService Structure is described in the following picture:



And finally the Application structure can be depicted as follows:



- specification
- code
- test
- configuration-deployment
- other
- lib
- bin
- project
 - win32
 - macos
 - linux

As applications there are considered:

- User interfaces
- Skins
- Main classes that can generate an executable by using framework content
- all the test tools needed for other parts (protocols) verification.
- Etc.

It is possible, depending on the software that is realized, to define some guidelines for the different kind on file to be put in each directory as stated in the next paragraphs.

In the following the general guidelines will be reported and in the more specific subsection only the changes that applies to the general structure will be reported.

8.1.3 General Guidelines

8.1.3.1 Directory include (optional: only for C/C++)

All the include (**.h, .hpp, .hh, etc**) that are generated for the project by the developers.

Inside this directory, a sub directory for each component will be created.

8.1.3.2 Directory source (mandatory)

All the source code organized in subdirectory defined by each development team (**.c, .cxx, .cpp, .java, etc**) that are generated for the project by the developers.

Inside this directory, a sub directory for each component will be created. Inside this directory the partner can organize the source code in the best manner according to the project type; some example follows:

- A set of subdirectories with source code put inside
- The source code directly placed in the directory
- A hierarchy of directory differentiating for source type (i.e. Java, C++, WebPages, etc)

8.1.3.3 Directory doc/specification

In this directory must be placed:

- the *UML diagram* in **XMI or Visio formats** in order to guarantee maximum interoperability and independence by UML editor software vendors that are generated for the software under development – mandatory for modules having OO classes;
- the *XML schema* adopted for the XML test case or for other activities related to the software under examination in **XML format** – mandatory if the schema was used;
- the *WSDL* used by the application for accessing as a client to web services on the server side in **plain text format** – mandatory if the WSDL was used.

8.1.3.4 Directory doc/code (mandatory)

In this directory the documentation of the code must be present. No submission without documentation can be acceptable. It is suggested to use automatic documentation generation according for example to JavaDoc or Doc++ style. In any case the documentation can be submitted only in the following formats in order to guarantee the maximum interoperability and independence by software vendors:

- HTML (preferred)
- Plain TEXT (preferred)
- RTF
- DOC

No other format will be accepted.

8.1.3.5 Directory doc/test (mandatory)

In this directory all the test cases and the test code must be submitted together with the operational instruction to test the code. Automatic testing is recommended and scripts for automating testing are kindly requested as *Batch file*, *Makefile*, *Ant* and so on.

When in test cases you are referring to some objects or digital resource please provide reference according to the documentation reported in the official deliverable on Test Cases, see WP2 and WP8 deliverables.

The documentation on how to test software must be inserted in a **plain text** file named *HowToTest.txt* in the root directory of the doc/test. This file must contain a reference to additional libraries (to be put on the lib directory) that are necessary for testing and not for deployment.

8.1.3.6 Directory doc/configuration-deployment (optional: only for modules requiring deployment)

In this directory a **plain text** file named *configuration-dependencies.txt* must be placed in the root of the directory doc/configuration-deployment and it must be filled with the information on dependencies of this software piece from other software pieces that are present in the repository tree together with the related version that have been tested to correctly operates.

In addition a **plain text** file named *deployment.txt* must be placed in the root of the directory doc/configuration-deployment containing all the information that are needed to know how to deploy the software artifact (such as the library that are needed, environment variables, additional software needed, target operating system, etc).

Any other file can be added, but the only formats allowed are:

- HTML
- Plain TEXT
- RTF
- DOC

8.1.3.7 Directory doc/other (optional)

This directory can contain any additional document useful for the development community. The only bound is the format that must be chosen among:

- HTML
- Plain TEXT
- RTF
- PDF

8.1.3.8 Directory lib (mandatory)

This directory must contain all the libraries that are not statically linked to the application and that are needed for the application in order to work. In the case the no library is needed an empty file named *no-lib-needed.txt* must be placed in the lib root directory. In the case of multiplatform development a directory for each platform have to be created and the lib must be put in the correct directory.

8.1.3.9 Directory bin (mandatory)

This directory must contain the product of the module and all the components (configuration files, etc) that are necessary for a correct exploiting of the software. It is considered product: an executable, a library, etc. For application that do not have an executable, the jar, war, dll, lib or the result of the module that have to be deployed must be put here. In the case of multiplatform development a directory for each platform have to be created and the executable must be put in the correct directory.

8.1.3.10 Directory project (optional: only for modules requiring building)

This directory must contain the makefiles or IDE projects to build the corresponding library or application. This directory can contain also the minimal set of file (in a unique zip file for easy recompiling the module in a defined IDE of the user. This zip file must contain only the minimal set of file, i.e. those after a project clean up.

8.1.4 C/C++ Application

8.1.4.1 Directory include

Must be present

8.1.4.2 Directory source

No change

8.1.4.3 Directory doc/specification

No change

8.1.4.4 Directory doc/test

No change

8.1.4.5 Directory doc/configuration-deployment

No change

8.1.4.6 Directory doc/other

No change

8.1.4.7 Directory lib

No change

8.1.4.8 Directory bin

No change

8.1.5 C/C++ Dynamic Library

8.1.5.1 Directory include

Must be present

8.1.5.2 Directory source

No change

8.1.5.3 Directory doc/specification

No change

8.1.5.4 Directory doc/code

No change

8.1.5.5 Directory doc/test

No change

8.1.5.6 Directory doc/configuration-deployment

No change

8.1.5.7 Directory doc/other

No change

8.1.5.8 Directory lib

No change

8.1.5.9 Directory bin

No change

8.1.6 C/C++ Static Library

8.1.6.1 Directory include

Must be present

8.1.6.2 Directory source

No change

8.1.6.3 Directory doc/specification

No change

8.1.6.4 Directory doc/test

No change

8.1.6.5 Directory doc/configuration-deployment

No change

8.1.6.6 Directory doc/other

No change

8.1.6.7 Directory lib

No change

8.1.6.8 Directory bin

No change

8.1.7 C/C++ WebService

8.1.7.1 Directory include

Must be present

8.1.7.2 Directory source

No change

8.1.7.3 Directory doc/specification

The WSDL used by the application for accessing as a client to other web services and WSDL used for generating the server side path or the web service under development on the server side in **plain text format**.

8.1.7.4 Directory doc/code

No change

8.1.7.5 Directory doc/test

No change

8.1.7.6 Directory doc/configuration-deployment

No change

8.1.7.7 Directory doc/other

No change

8.1.7.8 Directory lib

No change

8.1.7.9 Directory bin

No change

8.1.8 Java Application

8.1.8.1 Directory include

Not present

8.1.8.2 Directory source

No change

8.1.8.3 Directory doc/specification

No change

8.1.8.4 Directory doc/code

No change

8.1.8.5 Directory doc/test

No change

8.1.8.6 Directory doc/configuration-deployment

No change

8.1.8.7 Directory doc/other

No change

8.1.8.8 Directory lib

This directory must contain all the libraries in JAR format that are needed by the java application. In the case the no library is needed an empty file named *no-lib-needed.txt* must be placed in the lib root directory.

8.1.8.9 Directory bin

This directory must contain the application packaged in a JAR file with a batch file for executing it. In the case of multiplatform script for launching the application, a directory for each platform has to be created and the script must be put in the correct directory.

8.1.9 Java Library (JAR)

8.1.9.1 Directory include

Not present

8.1.9.2 Directory source

No change

8.1.9.3 Directory doc/specification

No change

8.1.9.4 Directory doc/code

No change

8.1.9.5 Directory doc/test

No change

8.1.9.6 Directory doc/configuration-deployment

No change

8.1.9.7 Directory doc/other

No change

8.1.9.8 Directory lib

This directory must contain all the libraries in JAR format that are needed by the java library. In the case the no library is needed an empty file named *no-lib-needed.txt* must be placed in the lib root directory.

8.1.9.9 Directory bin

This directory must contain the library packaged in a JAR file.

8.1.10 Java Webservice

8.1.10.1 Directory include

Not present

8.1.10.2 Directory source

No change

8.1.10.3 Directory doc/specification

The *WSDL* used by the application for accessing as a client to other web services and *WSDL* used for generating the server side path or the web service under development on the server side in **plain text format**.

8.1.10.4 Directory doc/code

No change

8.1.10.5 Directory doc/test

No change

8.1.10.6 Directory doc/configuration-deployment

No change

8.1.10.7 Directory doc/other

No change

8.1.10.8 Directory lib

No change

8.1.10.9 Directory bin

No change

8.2 Guidelines for the submission process

The objective of this process is to guarantee that each partner will “commit” the MODULES resulting from the development activities into a central repository (i.e.: the CVS tree) according to the defined standard format. The defined format takes into account all the technical components and documentation that is needed by other partners or external users to download the components and to adopt them into the applications.

The CVS is a complicated tool that is easy enough to use for normal users. A repository of files, arranged in a directory structure as described in the next paragraph, will be stored in the central server and all partners and future users could **check out** copies of this repository. This would create a copy of the (current version of the) files in the users local hard drive.

The management of the central repository is under the responsibility of Exitech and DSI.

Each partner could:

- Update the copy of the CVS tree to reflect the changes made since the last updated.
- Commit the changes to the CVS tree so that others can see them.

Each partner could only “add” a new file. A new version of a file will update the old one.

No “delete” operation should be done on the central CVS, unless it is necessary to guarantee the congruency at the system level.

All the partners will make changes to their own MODULES at their leisure (without connection to the server), and subsequently **commit** these changes to the main repository. It will not be possible for more than one user to work on (and commit changes in) the same MODULE. Each module has a Responsible.

Because CVS retains all previous versions of all files that were ever part of the repository, so it will be possible (though not easy) to recover from mistakes.

For creating new folders into the repository a request has to be send to the project coordinator with the repository administrator in cc. The email has to contain the names of the folders to be created, the content description and any other relevant information.

When a commit is performed an index.txt file has to be also updated (or created), for any folder, where the committed data is briefly explained.

The verification process

The verification process (Responsible: Exitech) will guarantee that the both technical contents and documentation will have been uploaded for each module. The Responsible will perform only a formal verification of the submitted data. Errors into the committed contribution will be communicated to the partner by the project coordinator.

Each partner will be responsible for the completeness and correctness of the modules.

The CVS process will not guarantee the integration of the components; anyway all the technical information is required to permit the “integrators” and “testers” to evaluate the components and to test according to the testing procedures.

The CVS system will guarantee the versioning functionalities; it will not provide configuration management functions.

Submission plan

Each partner has to submit the total contribution according to the development plan; anyway each partner could “commit” the new version of a component after any “relevant” upgrade (bug fix, new functionalities, updated documentation).

The maintainers of the CVS will verify after each “commit” the completeness of the documentation. Moreover, some statistics will be provided automatically for each component (i.e.: file age, number of submission per partners, etc).

The objective of this process is to guarantee that each partner will “commit” the MODULES resulting from the development activities into a central repository (i.e.: the CVS tree) according to the defined standard format. The defined format takes into account all the technical components and documentation that is needed by other partners or external users to download the components and to adopt them into the applications.

8.3 Guidelines for checking-out, updating committing

For managing the AXMEDIS repository the Subversion (<http://subversion.tigris.org/>) platform has been chosen.

A complete Subversion documentation and also a Subversion book are available on the project website <http://subversion.tigris.org/servlets/ProjectDocumentList>.

Subversion is a free/open-source version control system. That is, Subversion manages files and directories over time. A tree of files is placed into a central repository. The repository is something like an ordinary file server, except that it remembers every change ever made to your files and directories. This allows recovering older versions of data, or examining the history of how data changed.

Subversion can access its repository across networks, which allows it to be used by people on different computers.

Some of Subversion capabilities are:

Directory versioning

CVS only tracks the history of individual files, but Subversion implements a “virtual” versioned file system that tracks changes to whole directory trees over time. Files and directories are versioned.

True version history

Atomic commits

A collection of modifications either goes into the repository completely, or not at all. This allows developers to construct and commit changes as logical chunks, and prevents problems that can occur when only a portion of a set of changes is successfully sent to the repository.

Versioned metadata

Each file and directory has a set of properties—keys and their values— associated with it. You can create and store any arbitrary key/value pairs you wish. Properties are versioned over time, just like file contents.

Choice of network layers

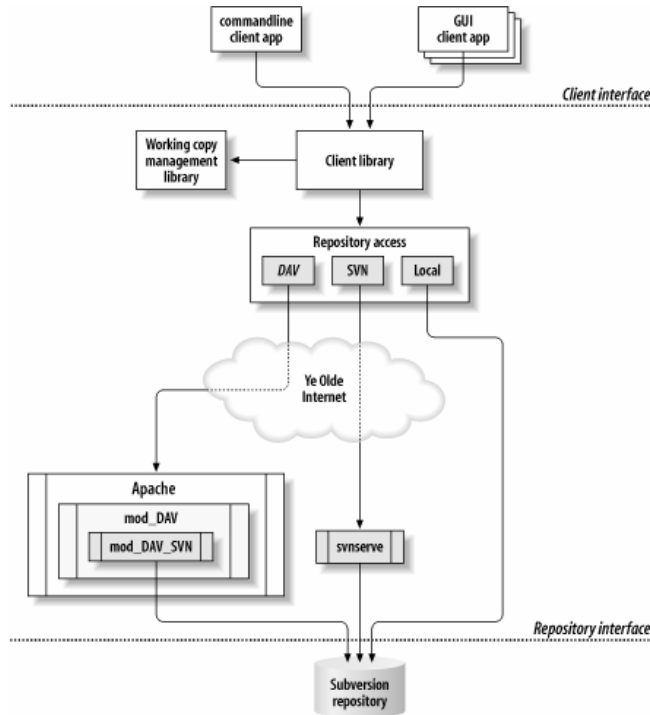
Consistent data handling

Subversion expresses file differences using a binary differencing algorithm, which works identically on both text (human-readable) and binary (human-unreadable) files. Both types of files are stored equally compressed in the repository, and differences are transmitted in both directions across the network.

Efficient branching and tagging

Subversion creates branches and tags by simply copying the project, using a mechanism similar to a hard-link. Thus these operations take only a very small, constant amount of time.

Subversion architecture:



Subversion architecture (extracted from: <http://svnbook.red-bean.com/en/1.0/svn-book.html>)

The AXMEDIS repository will be configured using the Apache “mod_DAV” & the Subversion “mod_DAV_SVN” since the “svnservice” seems to give some limits to the user configuration.

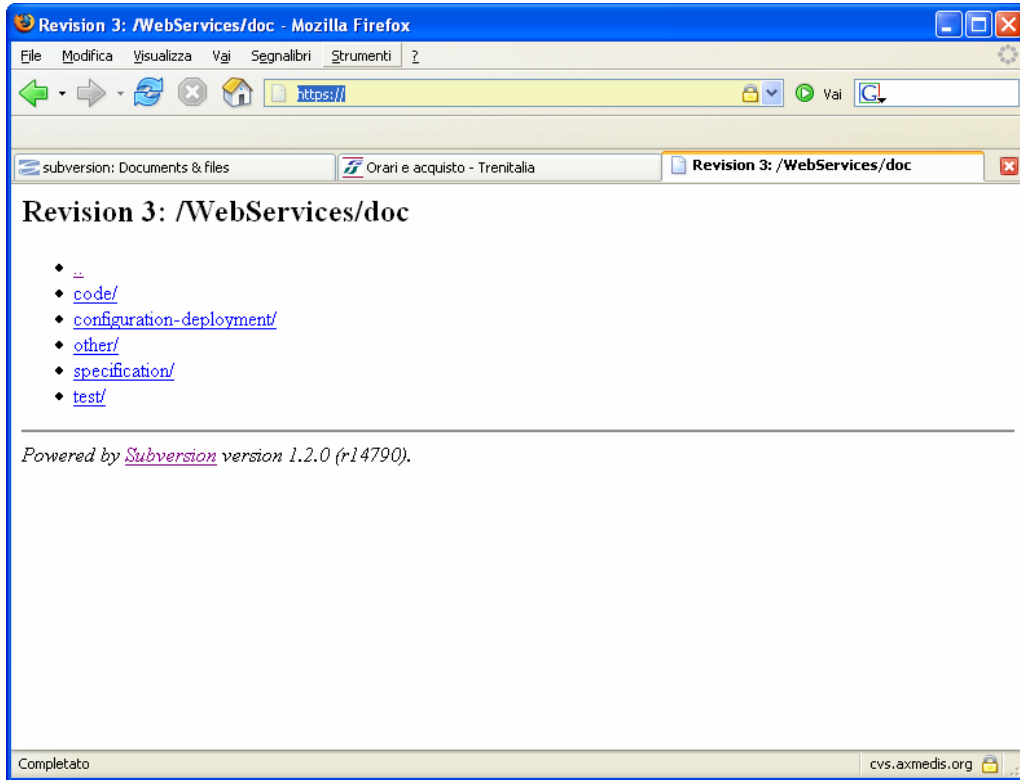
The AXMEDIS repository will be available on a SSL protected site. The repository will be protected with an account and a password and the internal folder will be configured in order to allow the access only to the selected users.

The repository can be accessed for reading purposes via browser, command line and using Subversion compatible GUI clients.

Web access

It will be possible to access and browse the (full or part of) repository according to the user rights. It is proposed to give one account for each partner.

A possible view of the repository could be the one shown in the next figure.



Command line client access

Command line clients are available for all the operating systems (MAC OS X, Linux, Solaris, BSD, and Windows).

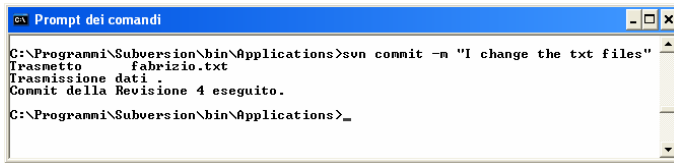
For getting a working copy of the Application Folder it is enough to run the “svn checkout” command. The following figure shows the Application folder checkout from a windows command line subversion client.

```

C:\Programmi\Subversion\bin>svn checkout http://cvs.axmedis.org/repos/Applicatio
ns
Reame di autenticazione: <http://cvs.axmedis.org:80> axmedis subversion reposito
ry
Password per 'narius':
Reame di autenticazione: <http://cvs.axmedis.org:80> axmedis subversion reposito
ry
Username: exitech
Password per 'exitech': *****
A Applications\exampleJavaApplication
A Applications\exampleJavaApplication\source
A Applications\exampleJavaApplication\doc
A Applications\exampleJavaApplication\doc\test
A Applications\exampleJavaApplication\doc\specification
A Applications\exampleJavaApplication\doc\other
A Applications\exampleJavaApplication\doc\configuration-deployment
A Applications\exampleJavaApplication\lib
A Applications\exampleJavaApplication\lib\bin
A Applications\exampleCppApplication
A Applications\exampleCppApplication\source
A Applications\exampleCppApplication\include
A Applications\exampleCppApplication\doc
A Applications\exampleCppApplication\doc\test
A Applications\exampleCppApplication\doc\specification
A Applications\exampleCppApplication\doc\other
A Applications\exampleCppApplication\doc\configuration-deployment
A Applications\exampleCppApplication\doc\code
A Applications\exampleCppApplication\lib
A Applications\exampleCppApplication\lib\bin
A Applications\fabrizio.txt
Estratta revisione 3.
C:\Programmi\Subversion\bin>

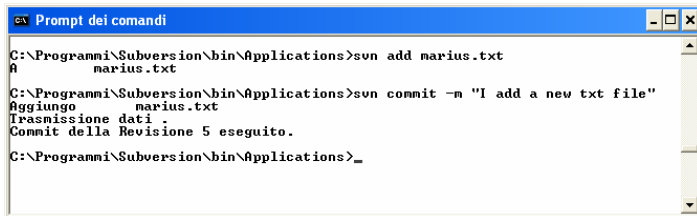
```

Let suppose that the fabrizio.txt file was changed and a new file is added to the working directory. The changes can be committed by using the “svn commit” as follow:



```
sv Prompt dei comandi
C:\Programmi\Subversion\bin\Applications>svn commit -m "I change the txt files"
Trasmetto      fabrizio.txt
Trasmissione dati .
Commit della Revisione 4 eseguito.
C:\Programmi\Subversion\bin\Applications>_
```

If the new file has to be added to the repository it has to be first added to the working directory:



```
sv Prompt dei comandi
C:\Programmi\Subversion\bin\Applications>svn add marius.txt
A
marius.txt
C:\Programmi\Subversion\bin\Applications>svn commit -m "I add a new txt file"
Trasmissione dati
Aggiungo      marius.txt
Commit della Revisione 5 eseguito.
C:\Programmi\Subversion\bin\Applications>_
```

A list of the other Subversion command line is available on the Subversion book (<http://svnbook.red-bean.com/en/1.0/svn-book.html>) at Chapter 9 (<http://svnbook.red-bean.com/en/1.0/svn-book.html#svn-ch-9>).

Using a GUI Subversion client

Several subversion clients are available. For a list you can see: http://subversion.tigris.org/project_links.html at the clients and plug-in section.

Also some development environments include GUI subversion clients.

8.4 Guidelines for adding new external libraries to the AXFW (ALL)

This section describes the process for adding new external libraries to be used by the tools inside the AXFW, apart from the ones declared in the specification documents.

8.4.1 Addition of external libraries

In order to add a new library, the following directives have to be followed:

- 1) Any library that is being used for the development of any tool to be included in the AXFW has to be authorised before its usage.
- 2) An AXMEDIS partner cannot decide autonomously to make some GPL or LGPL if it comes from the project results. The coding of any part is IPR of the project and thus it has to be authorised and any implication has to be verified.

The violation of these directives implies may lead at the exclusion of a partner for bad behaviour. The CA reports most of the details on this issue.

8.4.2 Authorisation of external libraries use

The inclusion of any library has to be authorised by sending a request to the project coordinator and the general reflector indicating:

- what would you like to use
- which OS
- why
- license model of the library

For instance, the authorisation will not be given if the module is GPL. It could be discussed depending on the point in which the module is used if it should be GPL. In any case has to be separately discussed.

Finally, the delivery of code in GPL or LGPL is not authorised without an explicit authorisation of the AXMEDIS consortium.

The list of accepted libraries is reported in the AXMEDIS portal (www.axmedis.org).

Each accepted library has to be posted on the portal in the corresponding folder with:

- library in binary and source code
- documentation of the library for its compilation and usage
- license

8.5 Source Code Repository Present Status

8.5.1 Webservice Formal Validation (Revision 1312 of the repository)

The situation of the web service repository is quite good and is depicted below.

		source	bin	lib	include	project	doc specification	doc configuration-deployment	doc code	doc other	doc test
EXITECH	AxdbQSWs										
EXITECH	AXDBSupportWS										
EXITECH	LoadeSaverWS										
EXITECH	QuerySupportWS										
EXITECH	UserSelection Archive										
FUPF	PMSWs										
FUPF	AXCSAXCWs										
DSI	axcs registration		war needed			missing	new/obsolete			empty	
FUPF	AXCSAXSWs										
DSI	axcs-oid-generator		war needed			missing	new/obsolete				
DSI	axcs-reporting		war needed			missing	new/obsolete			empty	
DSI	axcs-statistics		war needed			missing	new/obsolete			empty	
EXITECH	LockUnlockWS										

From a statistical point of view we have the following general figures:

OK	69,2%
REC	0,0%
TBR	30,8%
CRIT	0,0%

Where:

- OK, means that no improvement is needed
- REC, means that the situation can be Recoverable with a low effort
- TBR, means that the situation has to be recovered
- CRIT, means that the situation is critical and it is urgent a recovery action.

From the point of view of the partner involved in this part we have the following result:

	OK	REC	TBR	CRIT	OVERALL
EXITECH	100%	0%	0%	0%	0,00
FUPF	100%	0%	0%	0%	0,00
DSI	0%	0%	100%	0%	2,00

Where an Overall value between 0 and 1 is considered as OK, an Overall between 1 and 2 is considered Recoverable and an Overall greater than 2 is considered Critical.

8.5.2 Application Formal Validation (Revision 1312 of the repository)

The situation of the web service repository is quite good and is depicted below.

		NOTES:	source	bin	lib	include	project	doc specification	doc configuration-deployment	doc code	doc other	doc test
DSI	axactivex	3	index.txt		missing		haveToBuild	empty		empty		
CRS4	axbdsim	3	index.txt	empty	empty		missing	missing				haveToBuild
DSI	axeditor	2	cpp		missing		haveToBuild	empty				
CRS4	axeptool_UI	DEPRECATED										
CRS4	axeptool_console	DEPRECATED										
CRS4	axmedia	DEPRECATED										
SEJER	AxMozillaPlayer	3			missing	empty	haveToBuild	missing		missing		missing
SEJER	axmozillaplugin	2	cpp		missing		haveToBuild	empty	haveToDeploy		empty	
DSI	axplayer	3	index.txt	cpp	missing		haveToBuild	missing		missing		missing
UNILEEDS	axpngeng	2	index.txt	cpp	empty		haveToBuild	empty				haveToBuild
DSI	collector indexer	3	index.txt	tgz	missing	missing	missing	missing		missing		haveToBuild
FUPF	contractgen	1	cpp				haveToBuild					
EPFL	Cryptlib	3	index.txt	missing	missing	missing	missing	missing	STRANGE FOLDERS HERE			missing
FUPF	drmeditorviewer (ex drmeditor)	0	cpp									
ILABS	kioskcomponents	1	index.txt				haveToBuild					
TISCALI	mediacub	3	index.txt	missing	missing	missing	missing	missing	STRANGE FOLDERS HERE			missing
UNILEEDS	metadatamapperGUI	3	index.txt	cpp	empty		haveToBuild	empty		empty		empty
ILABS	mobilecomponents	1	index.txt				haveToBuild					
EPFL	OGL_EPFL	3	index.txt	missing	missing	missing	missing	missing	STRANGE FOLDERS HERE			missing
EPFL	osmo	3	index.txt	no folder	missing		missing	missing	STRANGE FOLDERS HERE			missing
UNILEEDS	pnpeditor	2	cpp					empty				haveToBuild
EPFL	PreProcessor	3	index.txt	missing	missing	missing	missing	missing	STRANGE FOLDERS HERE			missing
CRS4	query_user_interface	DEPRECATED										
DSI	ruleeditor	2			missing		haveToBuild	empty				
DSI	rulexecutor	3	wrong folder "client"		missing	empty	haveToBuild	empty		empty		
DSI	rulescheduler	2	wrong folder "clipart"	cpp	missing		haveToBuild	empty				
EPFL	visualeditor	3	index.txt	cpp			missing	missing		missing		missing

From a statistical point of view we have the following general figures:

OK	4,3%
REC	13,0%
TBR	26,1%
CRIT	56,5%

Where:

- OK, means that no improvement is needed
- REC, means that the situation can be Recoverable with a low effort
- TBR, means that the situation has to be recovered
- CRIT, means that the situation is critical and ir urgent a recovery action.

From the proint of view of the partner involved in this part we have the following result:

	OK	REC	TBR	CRIT	OVERALL	
FUPF	50%	50%		0%	0%	0,5
ILABS	0%	100%		0%	0%	1
UNILEEDS	0%	0%		66%	34%	2,34
SEJER	0%	0%		50%	50%	2,5
DSI	0%	0%		43%	57%	2,5714286
EPFL	0%	0%		0%	100%	3

Where an Overall value between 0 and 1 is considered as OK, an Overall between 1 and 2 is considered Recoverable and an Overall greater that 2 is considered Critical.

8.5.3 Framework Formal Validation (Revision 1312 of the repository)

The situation of the web service repository is quite good and is depicted below.

		updates	no updates				yes				yes		
	NAME	FLAG	NOTES:	source	bin	lib	include	project	doc	doc	doc	doc	doc
									specification	figuration-deploym	code	other	test
EPFL	adaptation		1	cpp				new/ok		new/ok			
FUPF	authorisation support		1	cpp		missing		new/ok					
CRS4	ax-p2p-webcache												
DSI	axcs		2					missing		new/ok			
FUPF	axcs-axcv		0										
FUPF	axcs-axs		0										
FUPF	axcsproxy												
EXITECH	AXDB		0										
EXITECH	AXDB-administrative		0										
FUPF	AXDB-license		1					empty					
FUPF	axdrmviewer		3	cpp	missing	missing		missing	missing		missing		missing
DSI	axeditor		1	cpp		empty		new/ok					
CRS4	axeptool												
CRS4	axeptool-modules												
DSI	axom		2	cpp		empty		new/ok		empty		empty	
EPFL	behaviour editor viewer		3	cpp		missing		new/ok	missing		missing		missing
EXITECH	camart		0										
DSI	command reporting		3	cpp				new/ok	missing		missing		HowToTest.txt
DSI	common		2	cpp				new/ok	empty				
FUPF	content consumption status		2	cpp		missing		new/ok		new/ok			
DSI	content formatter		3	cpp		missing		new/ok	missing		missing		HowToTest.txt
DSI	content processing		3	cpp	missing	missing		new/ok	missing		missing		missing
FUPF	contractgen		3	cpp		missing		new/ok	missing				
EPFL	cppunit		3	cpp		missing		missing	missing		missing		missing
LABITA	descriptor		1	cpp				new/ok		new/ok			HowToTest.txt
SEJER	documentviewer		3	cpp		missing		new/ok	missing		missing		
FUPF	drm editor viewer		3	cpp				new/ok	missing		missing		missing
FUPF	drmeditorviewer/												
FUPF	drmsupport-deprecated/												
FUPF	drmsupport		3	missing	missing			missing					HowToTest.txt
FUPF	encdecscup		1	cpp		empty		new/ok		new/ok			
FUPF	encdecscup-plugin/		3	cpp		missing		new/ok	missing		missing		missing
FHGIGD	fingerprint		1	cpp				new/ok		new/ok			HowToTest.txt
DSI	gridsupport		2	cpp				new/ok	empty				
DSI	jsaxclasses		2	cpp				new/ok	empty				
FUPF	keygen		1	cpp				new/ok		new/ok			
FUPF	licencecreator		3	missing	missing	empty	missing	missing			missing		missing
FUPF	licensegenerator		3	cpp				new/ok	missing				missing
FUPF	licensemanager		1	cpp		empty		new/ok					
FUPF	licensemodel		1	cpp		empty		new/ok					
FUPF	licenseverificator		1	cpp		empty		new/ok					
DSI	metadata_editor_viewer		3	cpp				new/ok	missing		missing		missing
UNILEEDS	metadateditor		3	cpp				new/ok	missing		missing		HowToTest.txt
UNILEEDS	metadatamapper		2	cpp				new/ok	missing				HowToTest.txt
UNILEEDS	metadatamapper_editor_viewer		3	cpp				new/ok	missing		missing		missing
UNILEEDS	metadatamodel		2	cpp				new/ok	missing				
DSI	mpeg4player		3	cpp				new/ok	missing		missing		missing
DSI	object_editor_viewer		3	cpp				new/ok	missing		missing		missing
FUPF	parquery support		0										
DSI	pluginmanager		2	cpp				new/ok			empty		HowToTest.txt
FUPF	pmsclient		1	cpp				new/ok					
UNILEEDS	prpeditor		1	cpp				new/ok					HowToTest.txt
UNILEEDS	prpmodel		1	cpp				new/ok					
DSI	protection_editor_viewer		3	cpp				new/ok	missing		missing		missing
FUPF	protectioninformanager		1	cpp				new/ok					
EXITECH	query-support-for-AXDB-core		0										
FHGIGD	query_on_demand		1					new/ok		new/ok			
FUPF	rddserver		1	cpp				new/ok					
DSI	ruleeditor		2	cpp				new/ok	empty				
DSI	ruleexecutor		2	cpp				new/ok	empty				
DSI	rulemodel		1	cpp				new/ok					
DSI	rulescheduler		2	cpp				new/ok	empty				
FUPF	seccommsup-deprecated/												
FUPF	seccommsup		3	missing	missing			missing					missing
FUPF	securecache		2	cpp				new/ok		new/ok			missing
DSI	selectioneditor		3	cpp				new/ok	missing		missing		missing
DSI	smil_player		3	cpp				new/ok	missing		missing		missing
EPFL	visualeditor/												
DSI	visual_editor_viewer		3	cpp				new/ok	missing		missing		missing
IRC	win32/												
IRC	workflow_database_channel		3	cpp				new/ok	missing		missing		missing
IRC	workflow_editor_channel		1	cpp				new/ok					missing
IRC	workflow_editor_viewer		3	cpp				new/ok	missing		missing		missing
IRC	workflow_engine_channel		1	cpp				new/ok					missing
IRC	workflow_rule_editor_channel		1	cpp				new/ok					missing

From a statistical point of view we have the following general figures:

OK	10,6%
REC	31,8%
TBR	19,7%
CRIT	37,9%

Where:

- OK, means that no improvement is needed
- REC, means that the situation can be Recoverable with a low effort
- TBR, means that the situation has to be recovered
- CRIT, means that the situation is critical and it is urgent to take a recovery action.

From the point of view of the partner involved in this part we have the following result:

	OK	REC	TBR	CRIT	OVERALL
EXITECH	100%	0%	0%	0%	0
DIPITA	0%	100%	0%	0%	1
FHGIGD	0%	100%	0%	0%	1
FUPF	13%	39%	9%	39%	1,7391304
IRC	0%	60%	0%	40%	1,8
UNILEEDS	0%	33%	33%	33%	2
EPFL	0%	33%	0%	67%	2,3333333
DSI	0%	10%	43%	48%	2,3809524
SEJER	0%	0%	0%	100%	3

Where an Overall value between 0 and 1 is considered as OK, an Overall between 1 and 2 is considered Recoverable and an Overall greater than 2 is considered Critical.

An analysis of the whole project in terms of source code repository consists in the figures reported in the table below:

Weighted Overall WebServices	0.615
Weighted Overall Framework	1.879
Weighted Overall Application	2.239
Weighted Overall Project	1.799

As always an Overall value between 0 and 1 is considered as OK, an Overall between 1 and 2 is considered Recoverable and an Overall greater than 2 is considered Critical.

9 Guidelines on AXMEDIS Framework Integration and Maintenance (FUPF)

This section aims to describe how to validate the AXMEDIS Framework by providing the tools or procedures to perform an integration testing of the different modules.

For this purpose, it is assumed that a Unit testing initial step has been already performed on the AXMEDIS modules, so that what has to be tested here is not the correctness of the modules operation but the interaction and compatibility with other modules.

For each of the AXMEDIS modules, it is expected that the involved partners provide the information detailed in next subsections.

9.1 Linked Modules

A list of modules that interact directly or indirectly with the present module and against which some aspects have to be verified to have a successful integration of the AXMEDIS Framework.

9.2 Integration Testing How To

A reference to the tools that have been created for the automatic or semi-automatic testing of the integration with other modules or, when automatic tests become difficult to be developed, the description of the procedure to be followed and the aspects to be verified during the integration testing.

9.3 Integration review report form

The reports regarding the results obtained during the review of the integration of different modules. Each report describes the issues and errors found during the review.

The main information to be reported when a module is being reviewed is described in the following table.

Review ID	Identifier of the review.
Module names	Name of the modules being reviewed.
Integration description	General purpose of the integration test review.
List of use cases	Related Use Cases
List of Test Cases	Related Test Cases
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.
Participants	Names, organisations and roles of people involved in the review.

ID	Who	Date	Issue location and description
Issue number	Name and organisation of the person who finds the issue	Date when the issue is found	Part of the module where the issue is found and description of the issues found in the module during revision and. The location could be the source code file, web page, etc; it will depend on the type of application.

9.4 Verification report form

Verification reports refer to the resolution or resolution proposal regarding review reports. They have to be created when a module is verified, after it has been reviewed and some issues have been found.

The information to be reported is described in the following table.

Verification ID	Identifier of the verification.
Review ID	Identifier of the review to which this verification refers.
Module names	Name of the modules being verified.
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.

ID	Who	Date	Status	Issue description and resolution
Related Review Issue number	Name and organisation of the person who verifies the issue	Date when the verification is performed	Solved, Proposed or Not Solved	How the issue found during the revision process has to be solved or has been solved

Regarding the Status column, it has to be filled in the following manner:

- Solved: when the issue in the review report has been solved
- Proposed: when the issue in the review report has been proposed but not solved
- Not solved: when the issue in the review report has neither been proposed nor solved

10 Guidelines on AXMEDIS Framework Regression testing (FUPF)

Regression testing is any type of [software testing](#) which seeks to uncover [regression bugs](#). Regression bugs occur whenever software functionality that previously worked as desired stops working or no longer works in the same way that was previously planned. Typically regression bugs occur as an unintended consequence of program changes.

Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have reemerged.

Experience has shown that as software is developed, this kind of reemergence of faults is quite common. Sometimes it occurs because a fix gets lost through poor [revision control](#) practices (or simple human error in revision control), but just as often a fix for a problem will be "fragile" - i.e. if some other change is made to the program, the fix no longer works. Finally, it has often been the case that when some feature is redesigned, the same mistakes will be made in the redesign that were made in the original implementation of the feature.

Therefore, in most software development situations it is considered good practice that when a bug is located and fixed, a test that exposes the bug is recorded and regularly retested after subsequent changes to the program. Although this may be done through manual testing procedures using programming techniques, it is often done using automated testing tools. Such a 'test suite' contains software tools that allows the testing environment to execute all the regression test cases automatically; some projects even set up automated systems to automatically re-run all regression tests at specified intervals and report any regressions. Common strategies are to run such a system after every successful compile (for small projects), every night, or once a week.

Regression testing is an integral part of the [extreme programming](#) software development methodology. In this methodology, design documents are replaced by extensive, repeatable, and automated testing of the entire software package at every stage in the [software development cycle](#).

Source: Wikipedia
http://en.wikipedia.org/wiki/Regression_testing

10.1 Unit test tools

The most common way for regression testing is to use unit tests. See below the list of unit test tools.

<http://www.testingfaqs.org/t-unit.html>

Generally unit test tools are language relatives. There are a lot of free tools available for regression testing in the languages used in AXMEDIS. These are some free popular unit testing tools for C++, C and Java.

<http://cppunit.sourceforge.net/>
<http://unitpp.sourceforge.net/>
<http://check.sourceforge.net/>
<http://junit.sourceforge.net/>

10.2 Regression testing guidelines

Due to the fact that most implementations of Axmedis are in C++, CppUnit seems to be the best choice. CppUnit is one of the most popular C++ testing free tools and because it is originally a port of JUnit, it will be close to Java (for Java features).

We recommend copying a version of CppUnit in the Axmedis framework for a global use.

Some recommendations:

- Every unit of the whole project which has predictable results or results that can be automatically tested should be tested.
- It is in charge of every partner to make his own tests.
- Regression test have to run with the actual version of the unit. Or the actual results (tests passed/failed/error) must be available with the regression tests.
- A modification of a unit with regression tests shouldn't be submitted without executing the regression tests. The result must be as good as the previous results (with no result specified, all tests must pass). If better results are obtained with a modification, don't forget to post the new results expected with it.

As said in recommendations it is in charge of every partner to make his own tests. Practically, in brief, it will be to make a new MVC++ project for every tested project, and to create testing files. A global testing can be produced directly by CppUnit due to the enormous amount of different workspaces of the whole project, but if it is requested, a batch file can be maintained.

One important thing to remember is that these kinds of tests are logical tests and consequently not user interfaces tests.

10.3 Testing using CppUnit

This section is dedicated to practical information and procedure to make testing code. First of all the SVN must be updated (to import CppUnit features).

Procedure:

1. Make a copy of your project (.vcproj).
2. Rename it to your_project_name_test.vcproj
3. Open your solution
4. Add the project your_project_name_test.vcproj (Menu: File/Add Project/Existing Project...)
5. Rename your new project as your_project_name_test (has originally your_project_name. By right-click on the project list)
6. Open your_project_name_test properties.
7. Modify if needed the these properties :
 - General
Configuration Type : Application(.exe)
 - Linker
 - General
Output File : change the file to your_project_name_test.exe
Additional Library Directories: add ../lib/cppunit
 - Input
Additional Dependencies : add cppunitd.lib in debug mode and cppunit.lib in release mode.
 - System
SubSystem : Console
8. Add Main.cpp in the project (is in source/cppunit)
9. Copy and modify to your own tests : your_module_test.h and your_module_test.cpp
10. Compile and execute
11. You can now watch your results in TestResult.xml (don't forget to post it with your module)

11 Guidelines on Performance Assessment and Optimisation (DSI)

Project of tests and performance analysis, verification and validation activities involving all modules in AXMEDIS Framework. This document expose methods and behaviours used to accomplish a full exhaustive test of all the modules, including Databases and related Web Services The latter chapters relate guidelines for Demo Package Production and File Installation .

11.1 Reference values

Please note to take into account the following ranges of values for the relevant variables:

- **AXOID: from 10.000 to 20.000.000 for each AXCS**
- **AXUID: from 10.000 to 2.000.000 for each AXCS**
- **AXDID: from 1.000 to 20.000 for each AXCS/PMS**
- **Grant conditions (rights) in the PMS database: from 10.000 to 20.000.000 for each PMS**
- **AXLID: Licenses ID: from 10.000 to 2.000.000 for each PMS server database**
- **AXMEDIS object size: from 100 Kbyte to 10 GByte**
- **Mean value of AXMEDIS objects:**
 - **Audio: 5 Mbyte, 5 minutes**
 - **Video:**
 - **Document:**
 - **MPEG4:**
 - **Images:**
- **Grant requests per year: from 200.000 to 2.000.000.000 for each PMS**
- **Action Log events per year: from 200.000 to 2.000.000.000 for each PMS**
- **AXTID: from 20.000 to 4.000.000 for each AXCS**
- **Number of queries on the AXMEDIS database per year:**
- **How many users per minute may I satisfy for a WS request (grant, registration, query, etc..) as a function of the status of the database (number of record, number of grants, number of action log, etc.)**
- **The number of users/requests that access at the same time fro the databases ranges:**
 - **AXMEDIS database: from 10 to 100**
 - **AXMEDIS registration portal from 1.000 to 100.000**
 - **AXMEDIS P2P database for searching objects: from 1000 to 100.000**
 - **AXMEDIS certificate production: from 1.000 to 20.000**
 - **AXCS Web Services: from 1.000 to 200.000**
- **AXCP total disk space: from 100 Mbyte to 10 TeraByte**
- **AXCP mean disk space per node:**
- **AXCP Number of nodes: from 1 to 100 (may be 200 to 2.000)**
- **AXCP Node CPU capability: fromMIPS/MPFLOPS to**
- **AXCP Node CPU percentage: from 10% to 100%**
- **For external function/procedure in plug ins: execution time normalised with respect to parameters for example Size X, Size Y, Time Duration, Frame per second, sample per second, bit/byte per pixel, size of file, etc.**

All the estimations have to contextualised in terms of the hardware and software used:

- **HW description:** CPU, memory, HD performance, network, etc.
- **SW configuration:** operative system, etc.

All the analysis has to be done taking into account the context and all the external influence, such as connection bandwidth throughput, hardware capabilities, other tasks running on the same machine, and so on.

11.2 Load tests meaning for databases and webservice

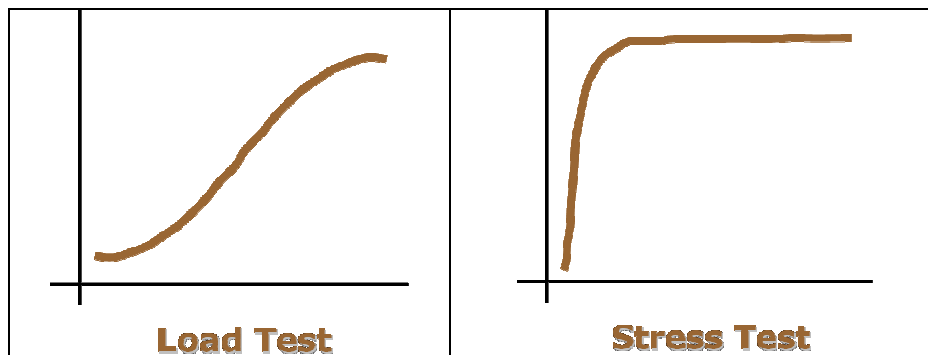
These tests have the purpose of getting a performance measure of databases and web services. They are performed with the scope of identifying the trend of the response time when the number of concurrency request increases. In this way it can get a sort of scalability measure of the tested resource.

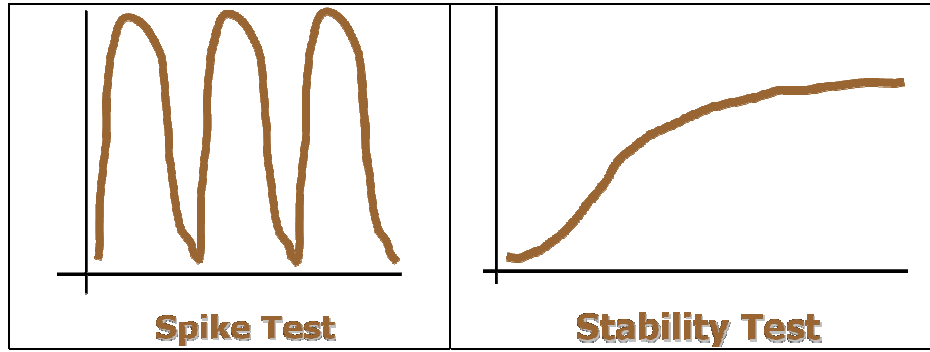
There are two main interesting measures: *time measure* and *capacity measure*. The first concerns the time needed to complete the required task and it is estimated starting the measure when the request is sent and ending when the response is received by the web service client. The second is about the maximum number of simultaneous requests that can be satisfied without get crashed (with a certain level of quality of service).

Concerning databases, it has to be detected the so called *reference* fields, which are the most significant fields of the databases in terms of number of tuples containing them.

Concerning web services, the most important performance measures are obtained executing the following tests:

- **Load Test.** The web service is put through a growing number of requests. The number of requests grows until the maximum tolerable number is reached. It is useful to understand the behaviour of the web service put through an overload of requests while operating in its normal environment.
- **Stress test.** The web service is put through load of requests greater than the standard one. It is useful to understand the behaviour of the web service if the number of request increase highly and unexpectedly, till it reaches a great value (near the maximum tolerable number).
- **Spike Test.** The web service is put through cyclic peaks of requests. A repeated great number of concurrency requests is useful to understand the web service behaviour when it is put through great load of requests.
- **Stability Test.** The web service is put through a standard load of requests for a long time. This test is useful to detect unexpected problems such as memory leaks.



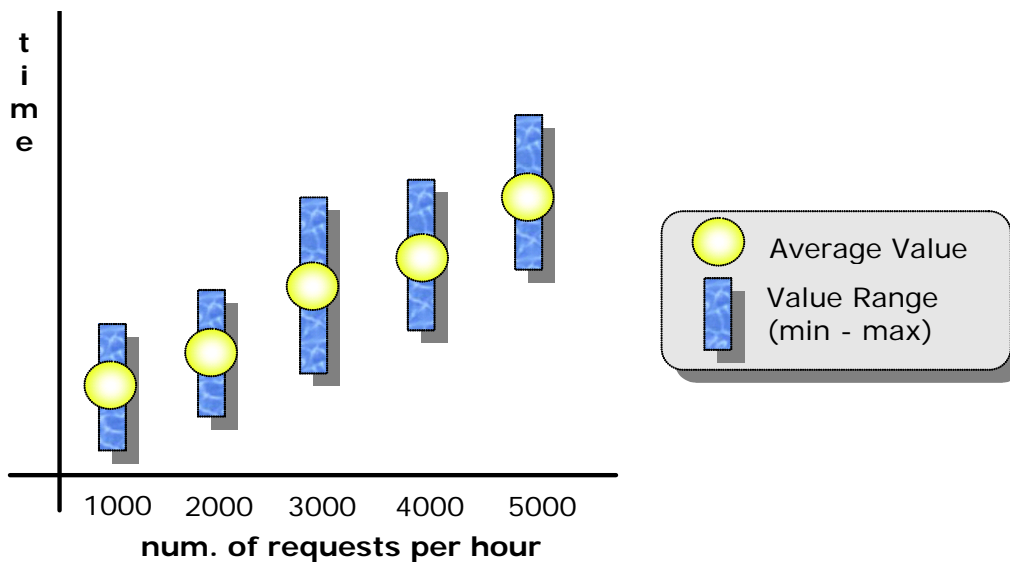


Typical trends of load and stress tests (number of requests per time)

A very common error while performing this kind of tests, is to consider only average values. It is a great error because lower and higher values are not taken into account even if they could be very important. Here is an example. Let us consider a web service which the target response time is 6 seconds and the average response time, measured for 100 clients is about 5 seconds. At a first sight it seems to be a good behaviour, but it is really not. Let us suppose the detected response times are distributed as follows:

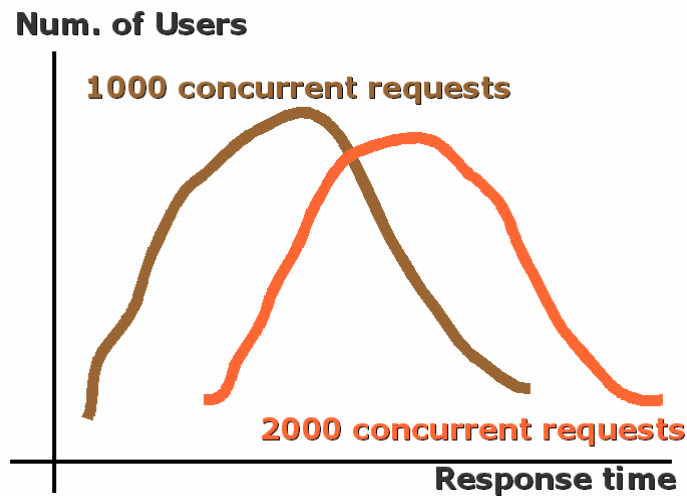
- 40 clients after a time of 3 seconds
- 25 clients after a time of 4 seconds
- 15 clients after a time of 7 seconds
- 20 clients after a time of 9 seconds.

In this scenario there are 40 clients that are fully satisfied, 25 clients that are satisfied, 15 clients that are not satisfied and 20 clients that are terribly unsatisfied. It is a situation quite different from considering only average values. It has to be considered also that an average value of 5 seconds could be produced by distribution times very different from this: 100 clients after 5 seconds (very improbable), 50 clients after 4 seconds and 50 clients after 6 seconds. Each of this distribution reflect a scenario that is very different from the others and from the first one. Here is evident the importance of the lower and the higher values.



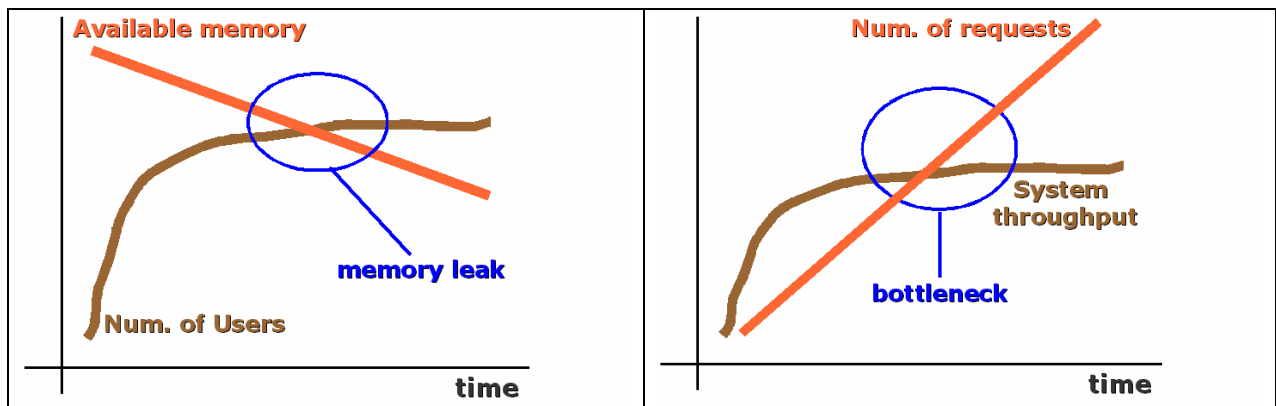
Lower and higher values for a distribution time scenario

Another important parameter that has to be taken into account to have a more accurate analysis is the distribution. There can be various kind of distribution, but the most important for the web services measure analysis is the number of users vs. response time for a fixed number of concurrent requests. It can be clearly represented in a graph where are indicated the different number of users (y-axis) that are served in the corresponding different times (x-axis), for a fixed number of concurrency requests. It is useful to understand the response time performances of the web service. If a different number of concurrency requests will be considered, the distribution trend will change. A suitable distribution is the one that groups small time values for a great number of users.



Number of Users per Response Time for two fixed number of concurrent requests (1000 req. brown coloured and 2000 req. orange coloured)

Load and stress analysis can also be used to discover implementation bugs. Indeed if system resources are monitored while performing the tests, it can be detected many bad behaviours of the web service such as bottleneck and memory leaks. If the available memory decreases and the load of requests remains nearly constant it will probably be a memory leak. If the number of requests increases and the system throughput doesn't augment it will probably be a bottleneck: the system reached its limit.



Memory leak and bottleneck conditions

It is not necessary to monitor all system resource: often it is sufficient to monitor the usage of base resources as memory, cpu, disk and so on.

11.3 AXMEDIS AXCS (DSI)

11.3.1 AXCS Registration and Certification database (DSI)

11.3.1.1 Table size / # of tuple

This test scenario checks the relationship about database size and the number of occurrences. Its reference fields are often the primary keys of the tables or the reference field for all the tables in a given database. A desirable behaviour leads to a linear growth of database size given a linear growth of reference fields. For AXCS Registration and Certification database the independent fields are:

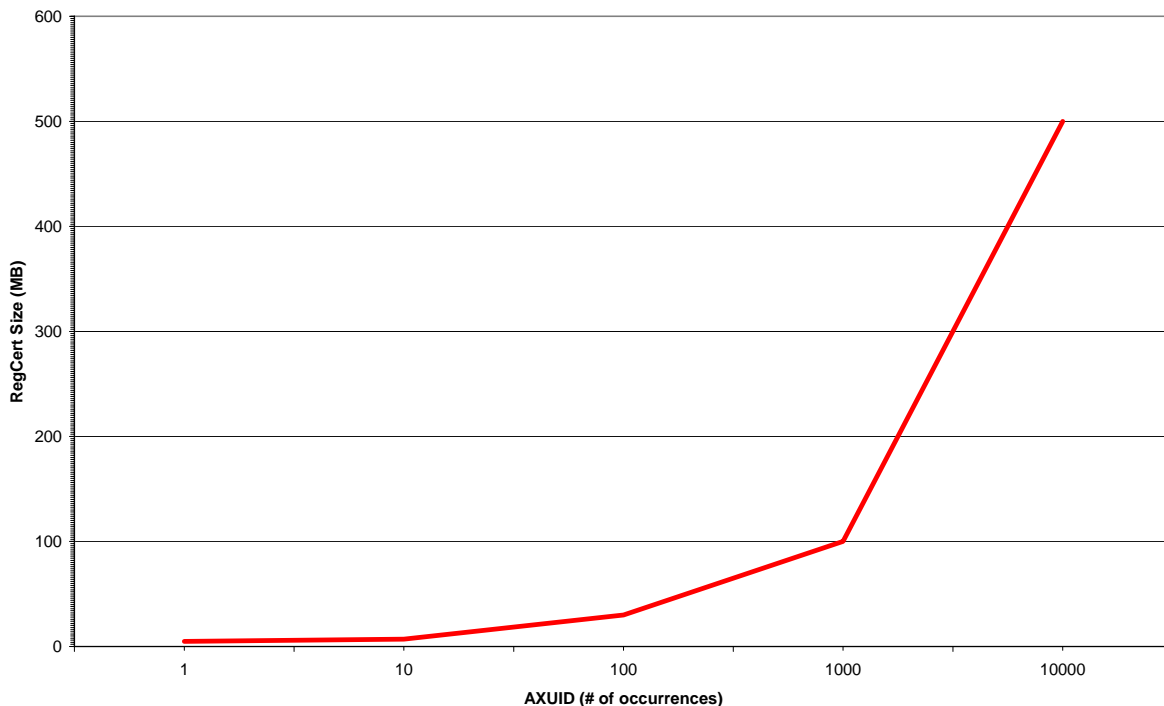
- AXUID – Number of occurrences of this field is directly related to b2busers, finalusers, genericusers, toolproducers, distributors and creators tables size. We can assume that : $\#of(b2busers) + \#of(finalusers) + \#of(distributors) + \#of(creators) \geq \#of(AXUID)$ where # is the number of tuple occurrences for each table. AXUID is also indirectly related to CertTools table size.
- AXTID – Number of occurrences of this field are indirectly related to RegTools table size, and directly related to CertTools table size.

Load tests on the database are performed following the reference given by these fields and so we can assume to create three load tests:

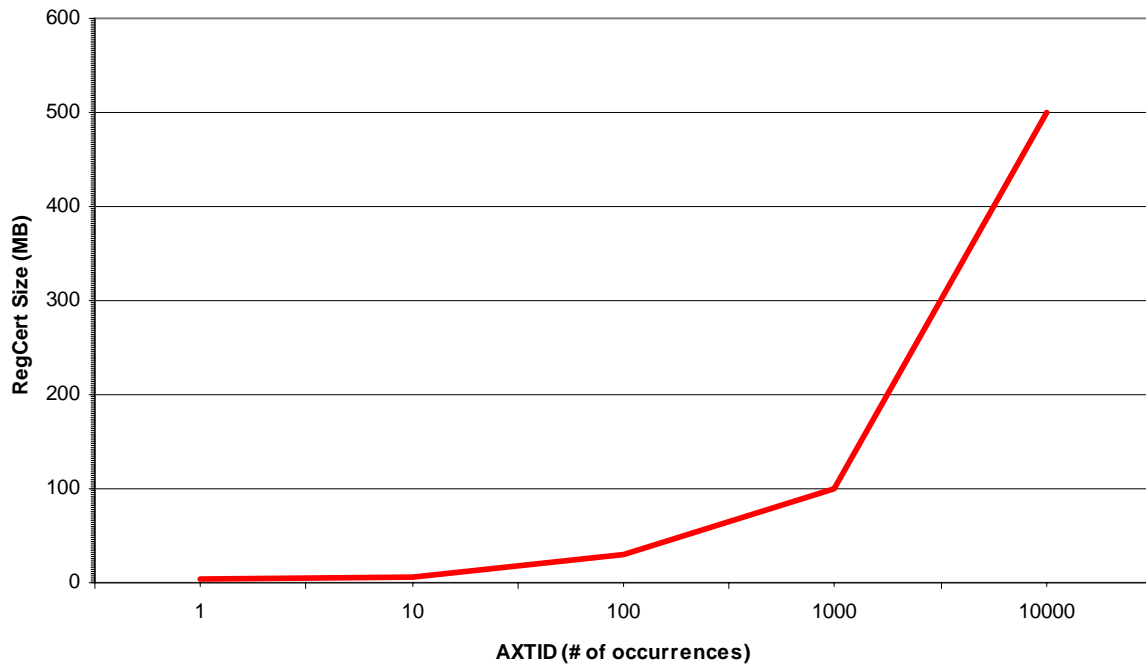
- A test driven by AXUID occurrences, assuming fixed AXTID and RegTools size, that measures the total size of database.
- A test driven by AXTID occurrences, assuming fixed AXUID, that measures the total size of the database. This test measures the incidence that number of certified tools has on the whole size of database

Graphics examples:

AXUID graphic



AXTID graphic



11.3.1.2 Search Time / # of tuple

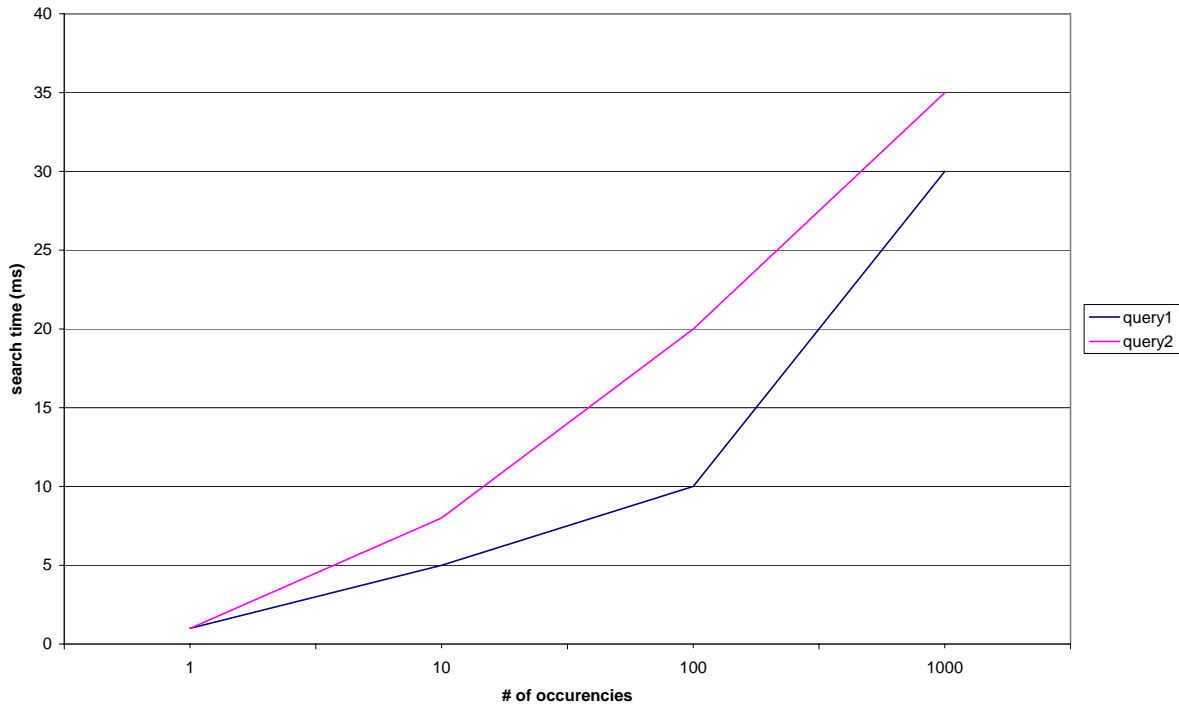
This test scenario shows how performances in executing queries decrease when database size growth. These kind of test is useful to determine the critical size of the database, beyond that size performances in query managing could decrease even exponentially. In this scenario results may have substantially changes due to the given query used to perform the test. A reasonable thing to do in projecting tests is to apply queries actually used by those web services that uses the database. Since the web service doesn't query directly the database but uses a middleware to access data, it is not possible to report exactly the queries performed by the web service to the database. Here are reported some significant queries to the AXCS Registration and Certification database.

- Insertion of data into the tables for a B2BUser with some role (creator, distributor, collecting society, tool producer)
- Insertion of data into the tables for a Final User
- Retrieval of data for a B2BUser (with some role) given the AXUID
- Retrieval of data for a Final User given the AXUID
- Retrieval of data for a some kind of user given some criteria (multiple retrieval)
- Retrieval of all data for all users (every row in genericuser table and related row in the other tables)

For each query presented the diagram is created showing # of occurrences of the target query vs. search time used to retrieve results occurred.

Moreover these tests has to be repeated considering different database sizes; these could be modelled keeping trace of previous test section results.

Graphics examples:

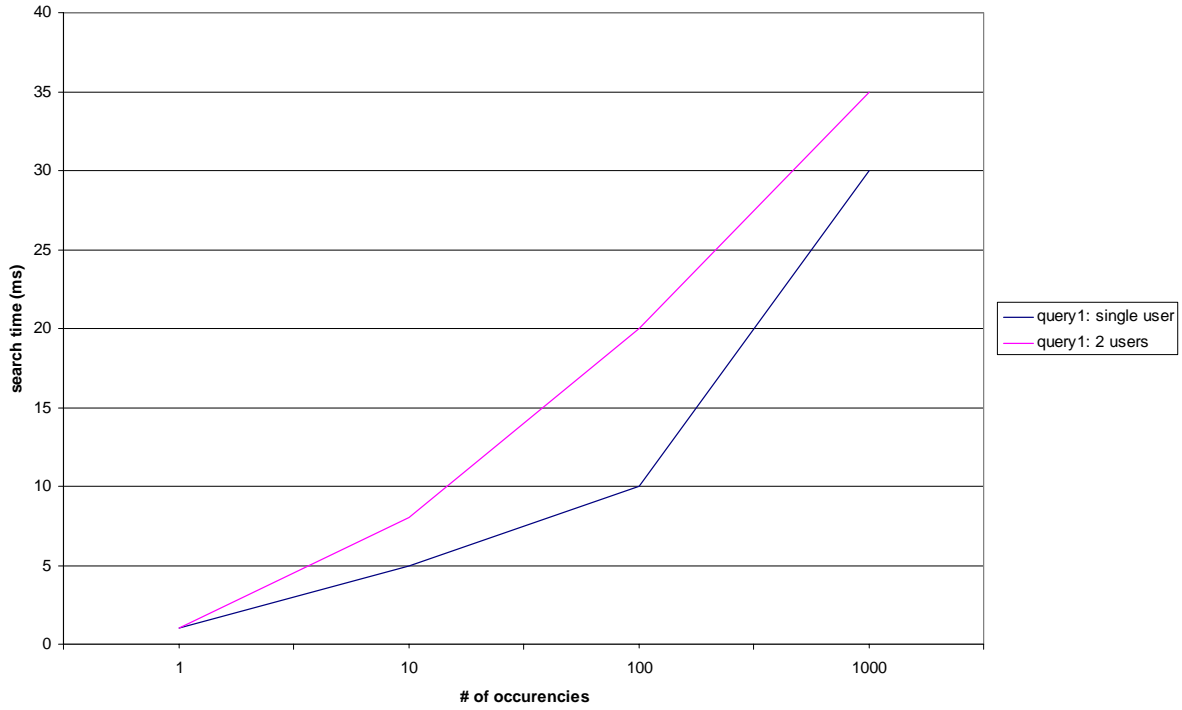


11.3.1.3 Single and Multiple access tests comparison

Another performance scenario that has to be considered is the case of multiple access, indeed in a concurrent access scenario a decrease in DBMS performances is expected. To state how this reduction will affect retrieve time for AXCS database queries is necessary to repeat search time tests in a multiple access scenario, driven by number of concurrent access to the database.

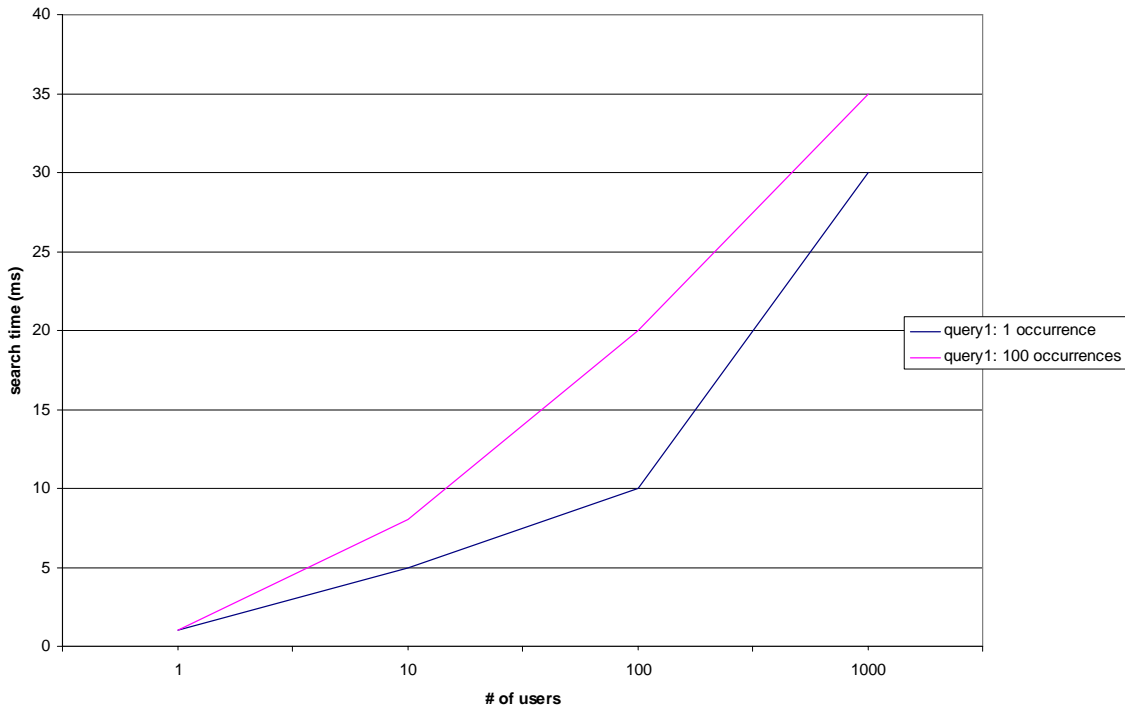
Graphics examples:

AXCS Registration and Certification Multiuser Query Graphic (1)



AXCS Registration and Certification Multiuser Query Graphic (2)

Y = search time X = # simultaneous connections – fixed: # of occurrences



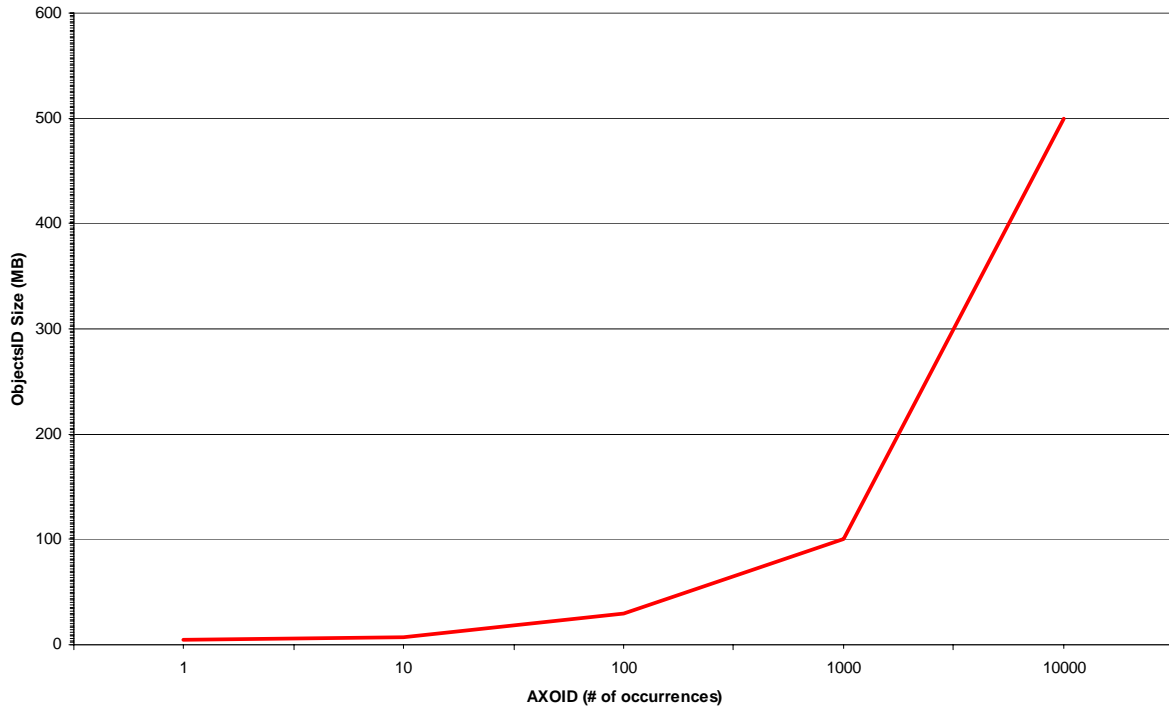
11.3.2 AXCS – ObjectsID (DSI)

11.3.2.1 Database size / # of tuple

This test scenario checks the relationship about database size and the number of tuples contained. ObjectsID tables finds their reference field in AXOID field. This field is the primary reference for all test realized for this database. Since AXOID field is the only reference measure, number of test is greatly reduced compared with the previous section

Graphics examples:

AXOID graphic



11.3.2.2 Search Time / # of tuple

This test scenario shows how performances in executing queries decrease when database size growth. These kind of test is useful to determine the critical size of the database, beyond that size performances in query managing could increase even exponentially. In this scenario results may have substantially changes due to the given query used to perform the test. A reasonable thing to do in projecting tests is to apply queries actually used by those web services that uses the database. Since the web service doesn't query directly the database but uses a middleware to access data, it is not possible to report exactly the queries performed by the web service to the database. Here are reported some significant queries to the AXCS ObjectsID database.

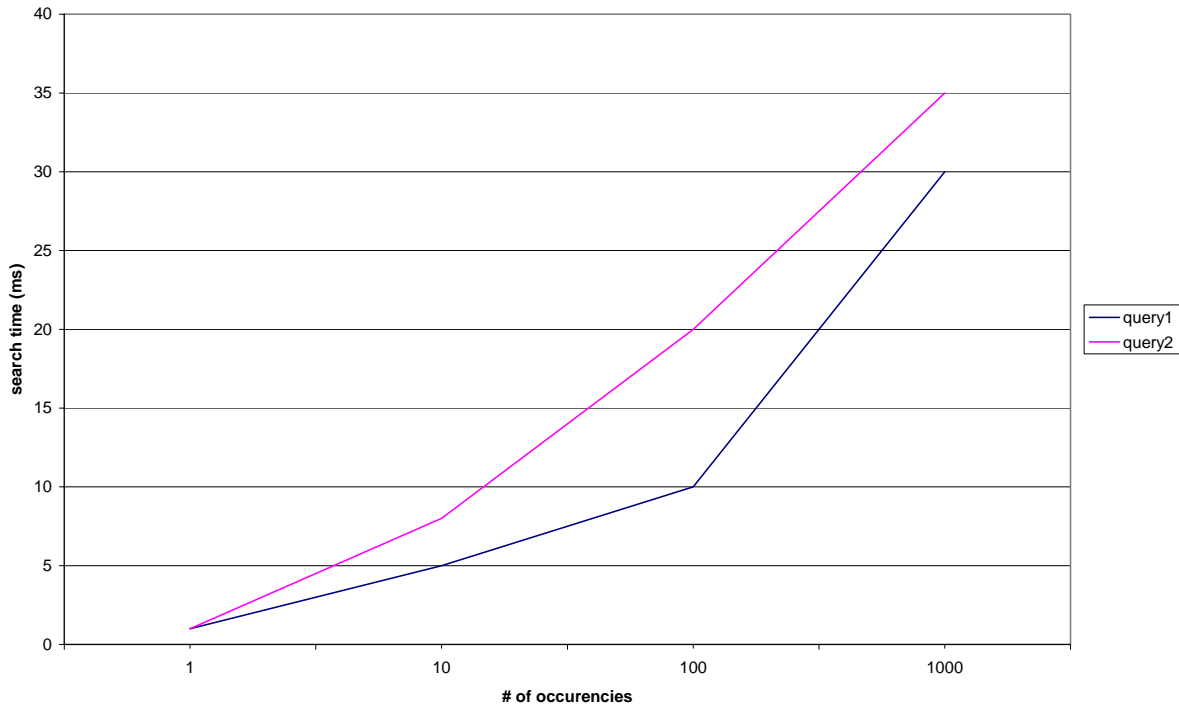
- Insertion of object related data into the objects table
- Retrieval of object related data from the objects table given an AXOID
- Retrieval of object related data from the objects table given some criteria (multiple retrieval)
- Retrieval of all data in the object table

For each query presented the diagram is created showing # of occurrences of the target query vs. search time used to retrieve results occurred.

This kind of test gives information not only about behaviours of different queries but also about evolution in search time when database increase its size, since a great number of occurrences as result for a query denotes a large size of given database. To link number of occurrences to database size refers to table size/# of tuples section.

Graphics examples:

Query graphic for AXCSObjectsID

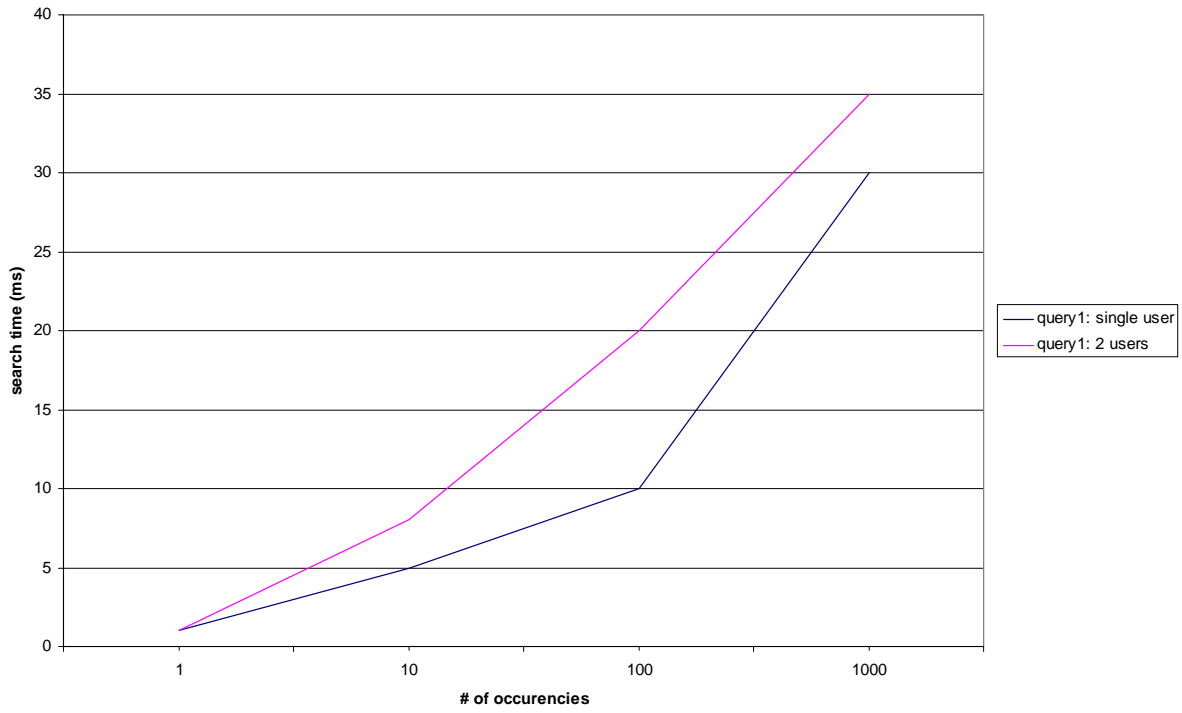


11.3.2.3 Single and Multiple access tests comparison

Another performance scenario that has to be considered is the case of multiple access, indeed in a concurrent access scenario a decrease in DBMS performances is expected. To state how this reduction will affect retrieve time for AXCS database queries is necessary to repeat search time tests in a multiple access scenario, driven by number of concurrent access to the database. One diagram for each query will be created as a result of this kind of measures.

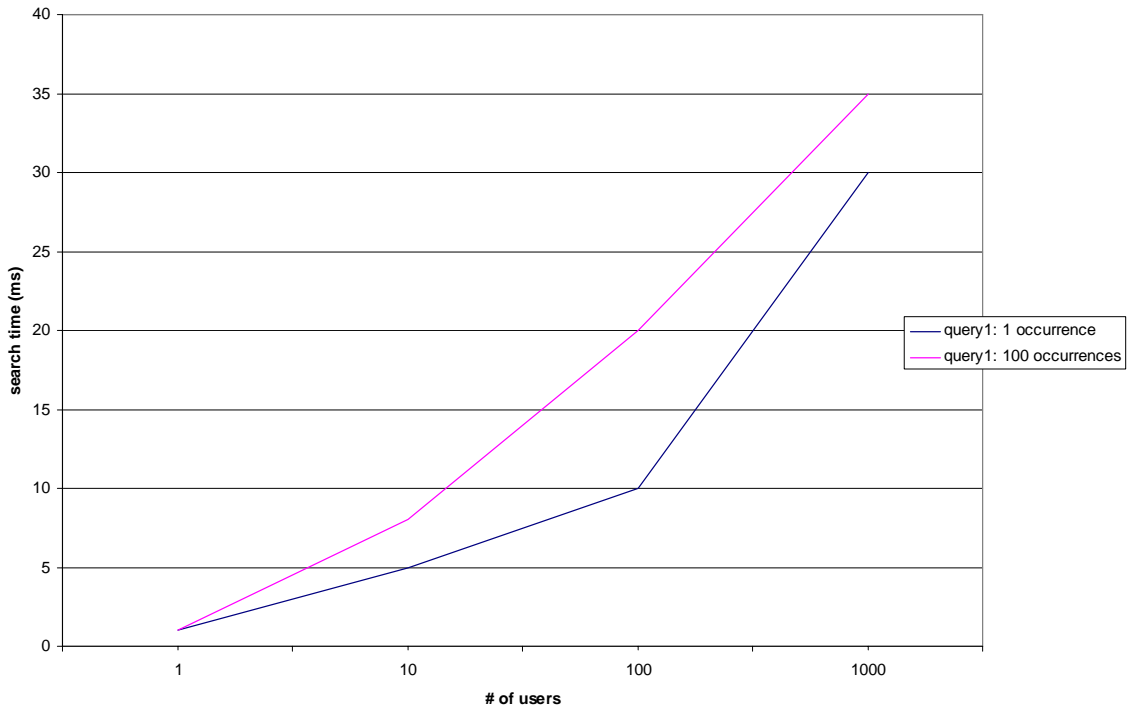
Graphics examples:

AXCSObjectsID Multiuser Query Graphic (1)



AXCSObjectsID Multiuser Query Graphic (2)

Y = search time X = # simultaneous connections – fixed: # of occurrences



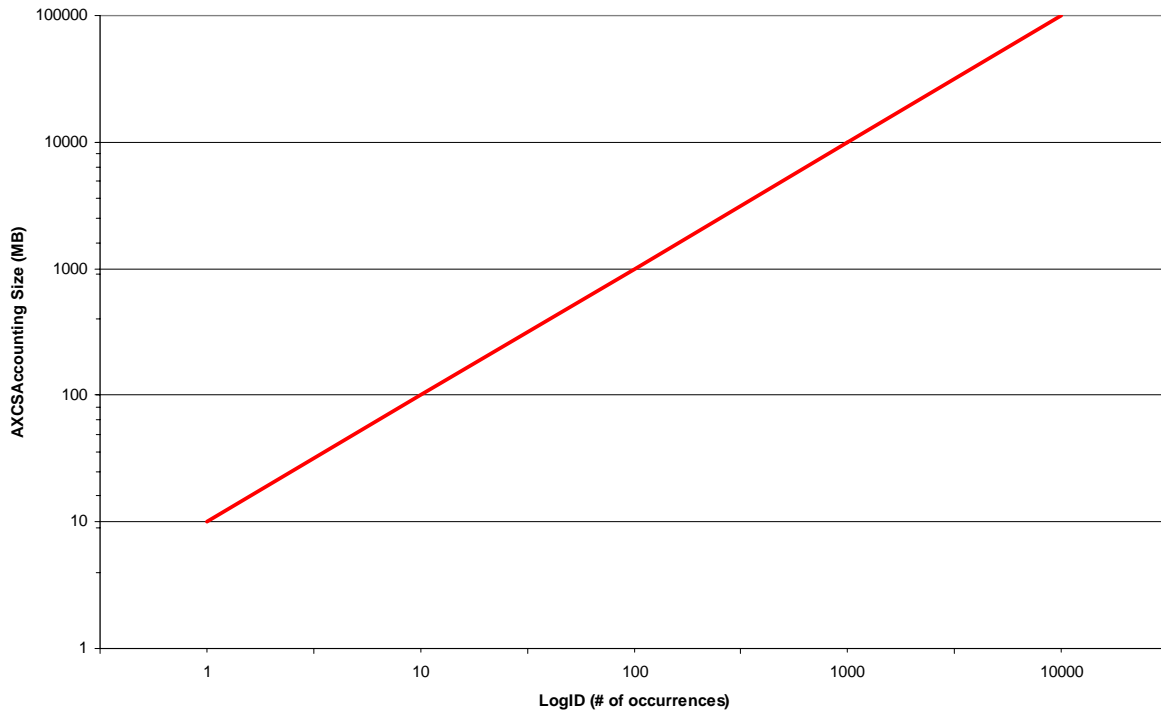
11.3.3 AXCS – Accounting (DSI)

11.3.3.1 Database size / # of tuple

This test scenario checks the relationship about database size and the number of tuples contained. Accounting tables finds their reference field in LogID field. This field is the primary reference for all test realized for this database. To consider that LogID in table actionlog and in table supervisorinputdata are different instances of action log. Thus database size is directly dependent from LogID number of tuple the related graphic will be always of type : $DBsize = A * (\# \text{ of LogID-tuple})$ where A is the average size of a LogID-tuple. The graph of this database will result something like this:

Graphics examples:

LogId graphic:



11.3.3.2 Search Time / # of tuple

This test scenario shows how performances in executing queries decrease when database size growth. These kind of test is useful to determine the critical size of the database, beyond that size performances in query managing could increase even exponentially. In this scenario results may have substantially changes due to the given query used to perform the test. A reasonable thing to do in projecting tests is to apply queries actually used by those web services that uses the database. For AXCSAccounting database these queries are:

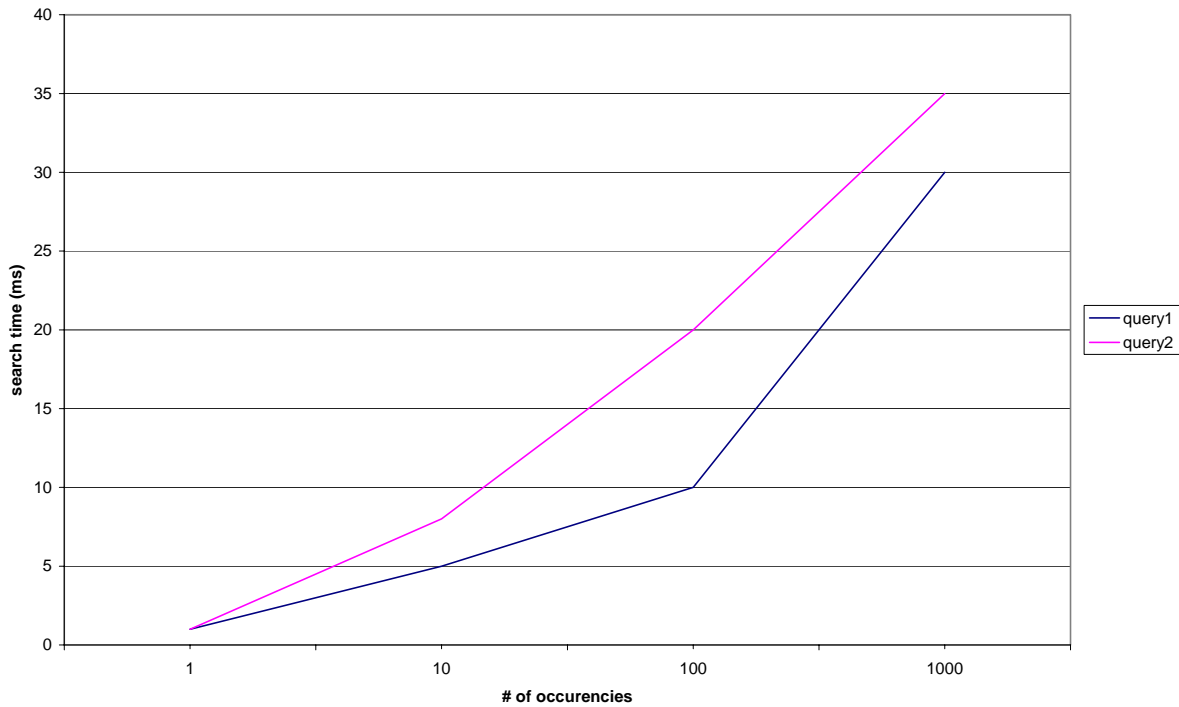
- Insertion of log related data into the actionlog table
- Retrieval of log related data from the actionlog table given a LogID
- Retrieval of log related data from the actionlog table given an AXOID (multiple retrieval)
- Retrieval of log related data from the actionlog table given an AXUID (multiple retrieval)
- Retrieval of log related data from the actionlog table given some criteria (multiple retrieval)
- Retrieval of all data in the actionlog table

For each query presented the diagram is created showing # of occurrences of the target query vs. search time used to retrieve results occurred.

This kind of test gives information not only about behaviours of different queries but also about evolution in search time when database increase its size, since a great number of occurrences as result for a query denote a large size of given database. To link number of occurrences to database size refers to table size/# of tuples section.

Graphics examples:

Query graphic for AXCSAccounting

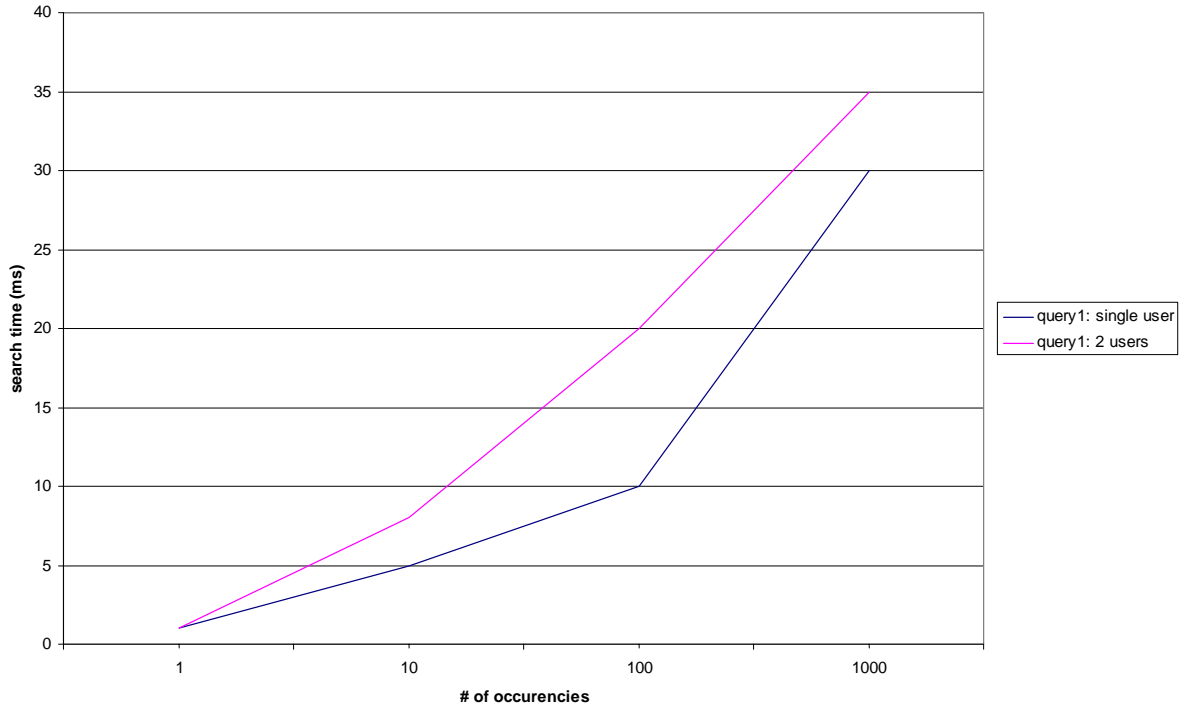


11.3.3.3 Single and Multiple access tests comparison

Another performance scenario that has to be considered is the case of multiple access, indeed in a concurrent access scenario a decrease in DBMS performances is expected. To state how this reduction will affect retrieve time for AXCS database queries is necessary to repeat search time tests in a multiple access scenario, driven by number of concurrent access to the database. One diagram for each query will be created as a result of this kind of measures.

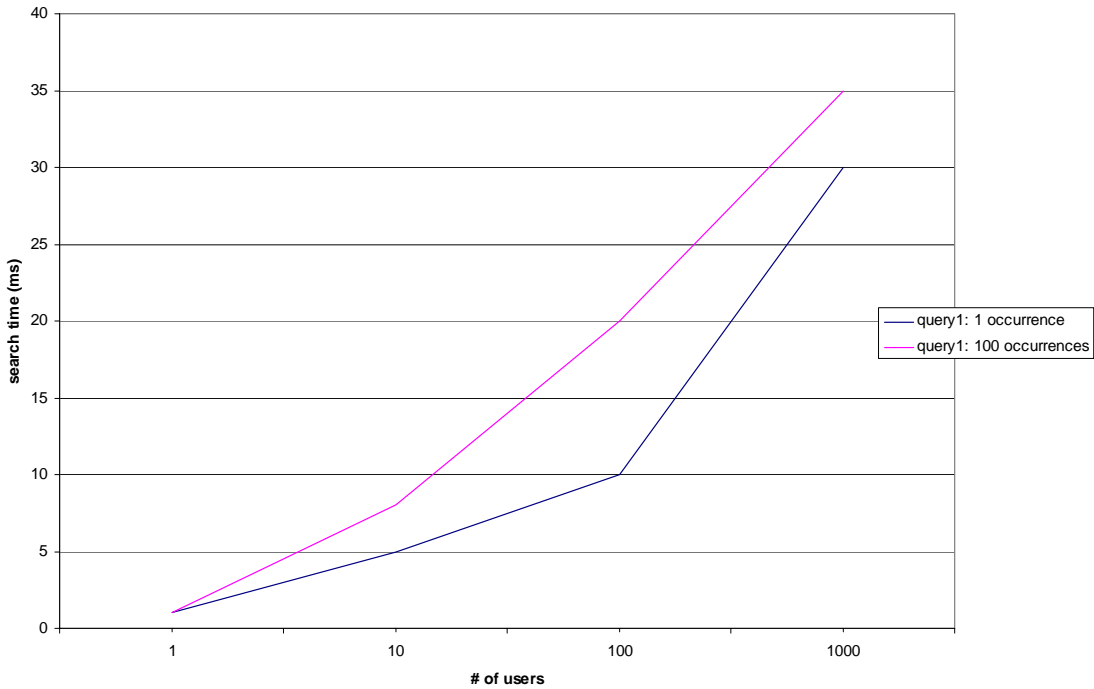
Graphics examples:

AXCSAccounting Multiuser Query Graphic (1)



AXCSAccounting Multiuser Query Graphic (2)

Y = search time X = # simultaneous connections – fixed: # of occurrences

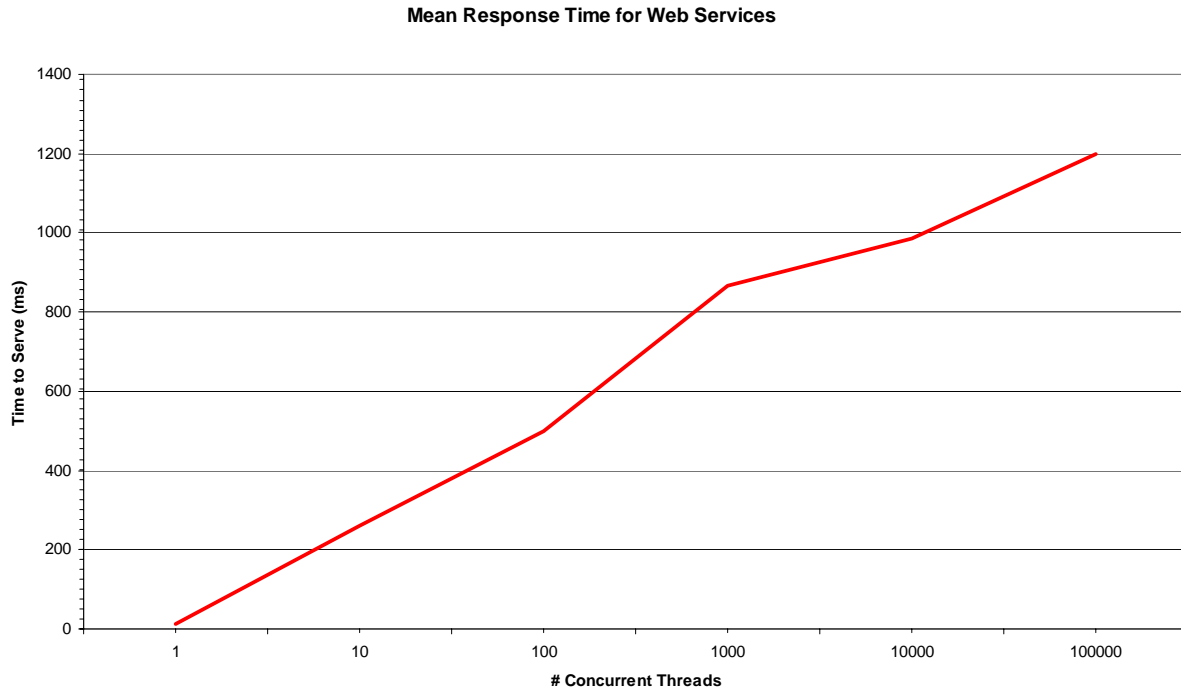


11.3.4 Users Registration Web Service (DSI)

This web services concerns about users registration and is linked to the performances of Registration and Certification database.

11.3.4.1 Time Tests\

Time tests concerns the average times of response per number of threads



To obtain the web service response time, it has to be considered also the performances of the related database, the AXCS Registration and Certification database (please see the pertinent section). Each thread is associated to an instance of the web service which performs one or more queries over the database and some data processing. Web service time response results as:

$$\sum_i (QryTime_i(\text{sizeofDB})) + dpt + nt$$

where

$QryTime(\text{sizeofDB})$ is the query time depending on the size of the db

i is the number of query

dpt is the data processing time

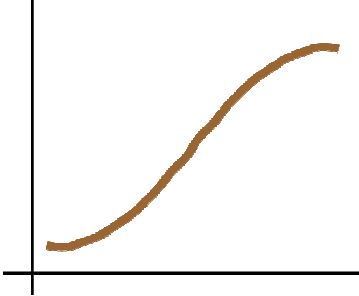
nt is the network time depending on the network infrastructure

It has to be noticed that dpt value will be significantly smaller than $QryTime(\text{sizeofDB})$ value. The nt value depends on the network infrastructure goodness.

11.3.4.2 Capacity Tests

The meaning of the following tests has been introduced at the beginning of the current section.

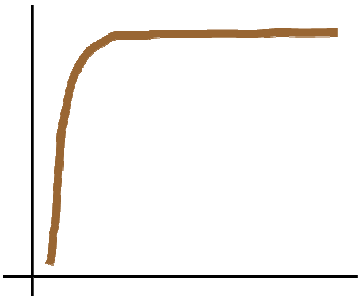
Load test: x = number of request, y = time



Load Test

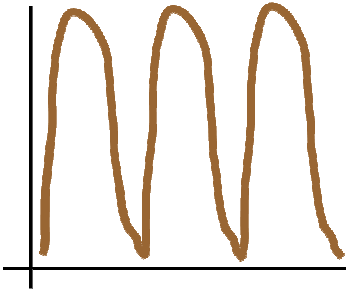
Expected trend for load test (number of requests per time)

Stress test: x = number of request, y = time



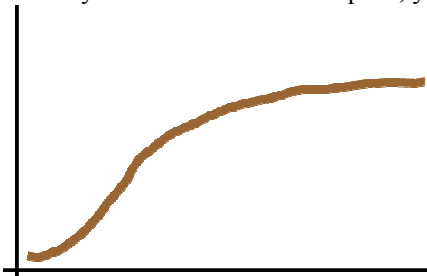
Stress Test

Spike test: x = number of request, y = time



Spike Test

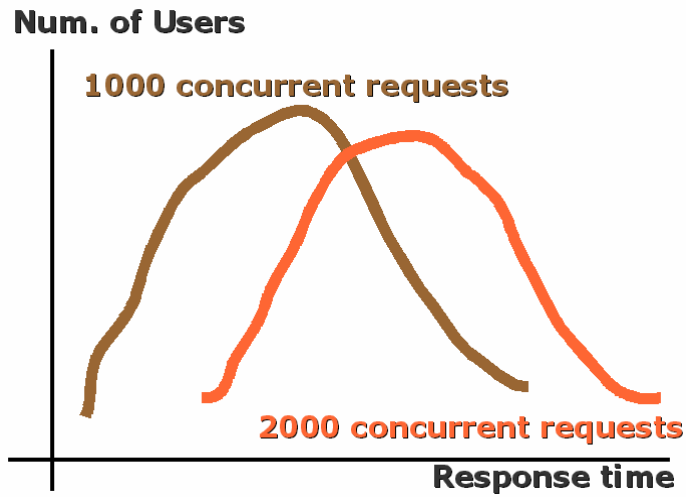
Stability test: x = number of request, y = time



Stability Test

11.3.4.3 Concurrency Tests

X = Response time, Y = number of users for a fixed number of concurrency request.



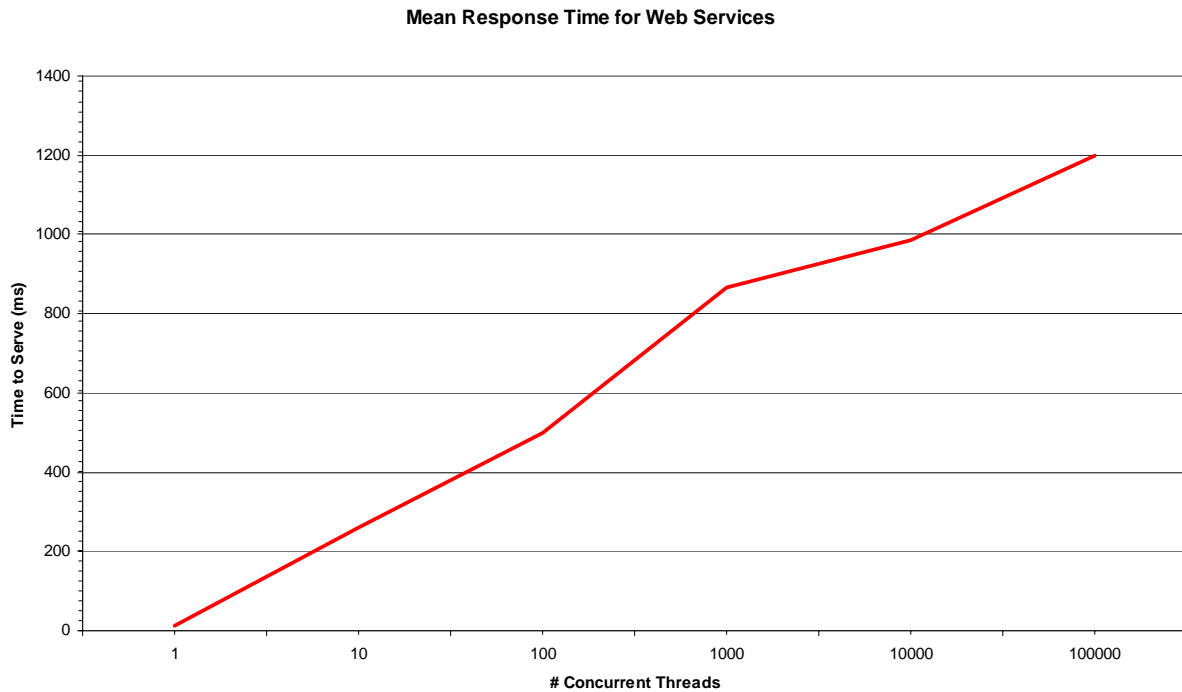
Number of Users per Response Time for fixed number of concurrent requests (1000 req. brown coloured and 2000 req. orange coloured)

11.3.5 Objects Registration Web Service (DSI)

This web services concerns about object metadata registration and is linked to the performances of ObjectID database.

11.3.5.1 Time Tests\

Time tests concerns the average times of response per number of threads



To obtain the web service response time, it has to be considered also the performances of the related database, the Objects ID database (please see the pertinent section). Each thread is associated to an instance of the web service which performs one or more queries over the database and some data processing. Web service time response results as:

$$\sum_i (QryTime_i(\text{sizeofDB})) + dpt + nt$$

where

$QryTime(\text{sizeofDB})$ is the query time depending on the size of the db

i is the number of query

dpt is the data processing time

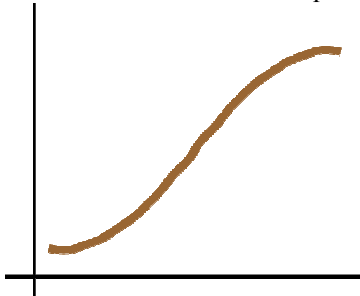
nt is the network time depending on the network infrastructure

It has to be noticed that dpt value will be significantly smaller than $QryTime(\text{sizeofDB})$ value. The nt value depends on the network infrastructure goodness.

11.3.5.2 Capacity Tests

The meaning of the following tests has been introduced at the beginning of the current section.

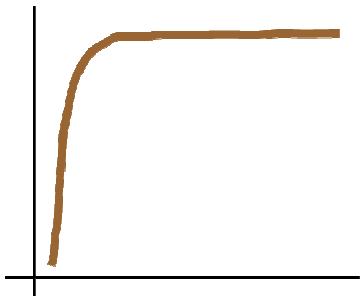
Load test: x = number of request, y = time



Load Test

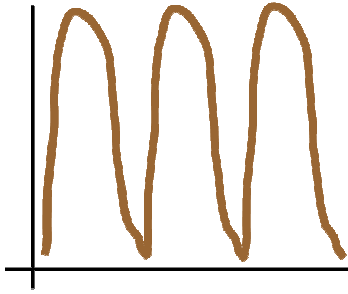
Expected trend for load test (number of requests per time)

Stress test: x = number of request, y = time



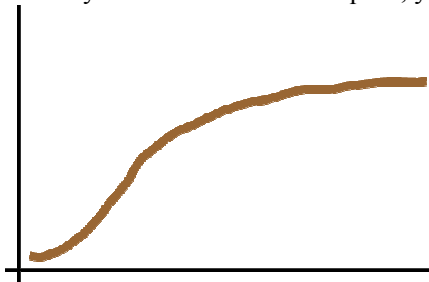
Stress Test

Spike test: x = number of request, y = time



Spike Test

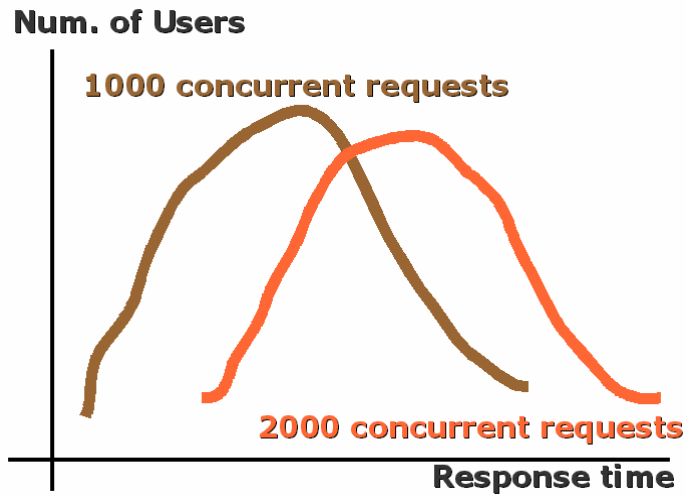
Stability test: x = number of request, y = time



Stability Test

11.3.5.3 Concurrency Tests

X = Response time, Y = number of users for a fixed number of concurrency request.



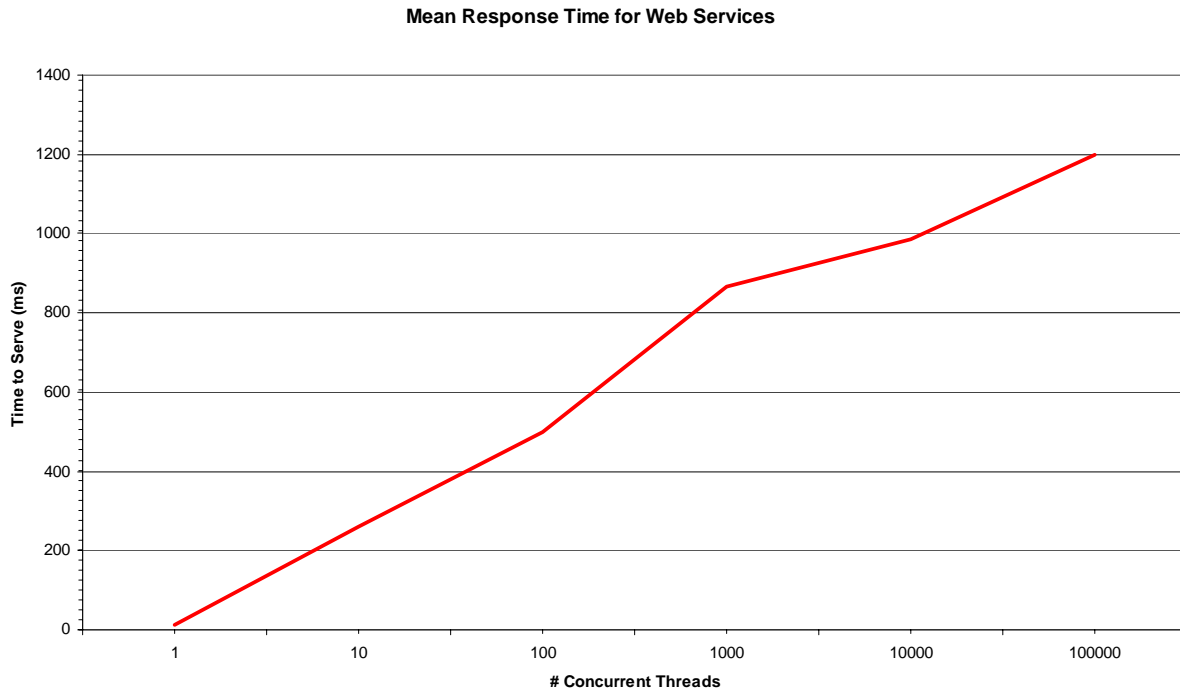
Number of Users per Response Time for fixed number of concurrent requests (1000 req. brown coloured and 2000 req. orange coloured)

11.3.6 Reporting Web Service (DSI)

This web services concerns about object metadata registration and is linked to the performances of Accounting database.

11.3.6.1 Time Tests\

Time tests concerns the average times of response per number of threads



To obtain the web service response time, it has to be considered also the performances of the related database, the AXCS Accounting database (please see the pertinent section). Each thread is associated to an instance of the web service which performs one or more queries over the database and some data processing. Web service time response results as:

$$\sum_i (QryTime_i(\text{sizeofDB})) + dpt + nt$$

where

$QryTime(\text{sizeofDB})$ is the query time depending on the size of the db

i is the number of query

dpt is the data processing time

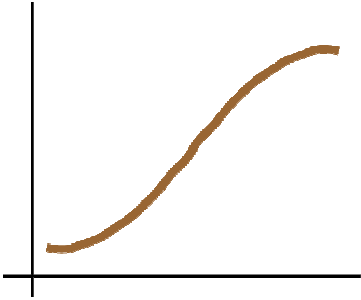
nt is the network time depending on the network infrastructure

It has to be noticed that dpt value will be significantly smaller than $QryTime(\text{sizeofDB})$ value. The nt value depends on the network infrastructure goodness.

11.3.6.2 Capacity Tests

The meaning of the following tests has been introduced at the beginning of the current section.

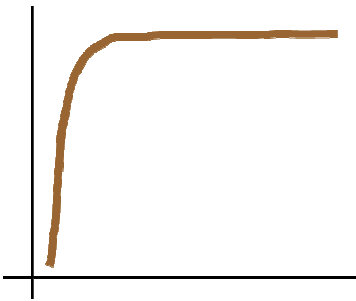
Load test: x = number of request, y = time



Load Test

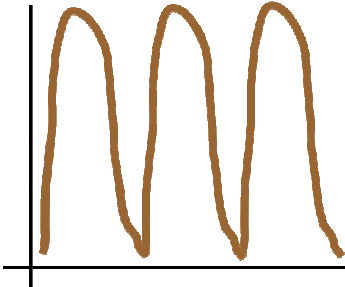
Expected trend for load test (number of requests per time)

Stress test: x = number of request, y = time



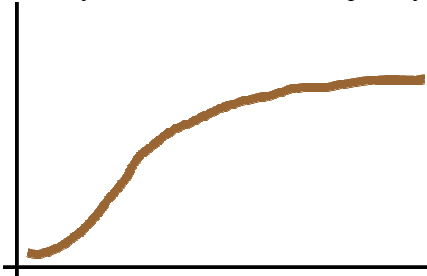
Stress Test

Spike test: x = number of request, y = time



Spike Test

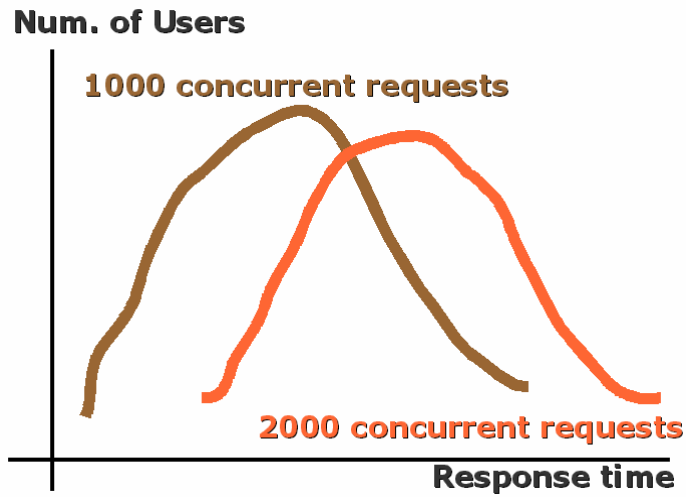
Stability test: x = number of request, y = time



Stability Test

11.3.6.3 Concurrency Tests

X = Response time, Y = number of users for a fixed number of concurrency request.



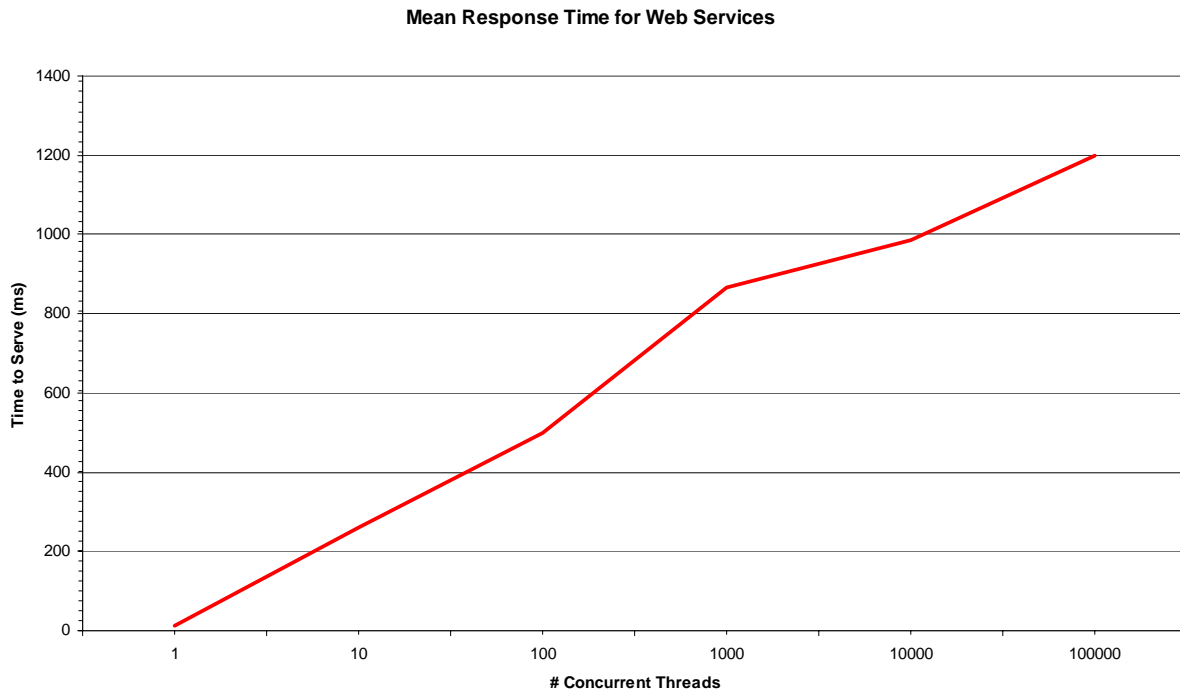
Number of Users per Response Time for fixed number of concurrent requests (1000 req. brown coloured and 2000 req. orange coloured)

11.3.7 Statistics Web Service (DSI)

This web services concerns about object metadata registration and is linked to the performances of Accounting database.

11.3.7.1 Time Tests\

Time tests concerns the average times of response per number of threads



To obtain the web service response time, it has to be considered also the performances of the related database, the AXCS Accounting database (please see the pertinent section). Each thread is associated to an instance of the web service which performs one or more queries over the database and some data processing. Web service time response results as:

$$\sum_i (QryTime_i(\text{sizeofDB})) + dpt + nt$$

where

$QryTime(\text{sizeofDB})$ is the query time depending on the size of the db

i is the number of query

dpt is the data processing time

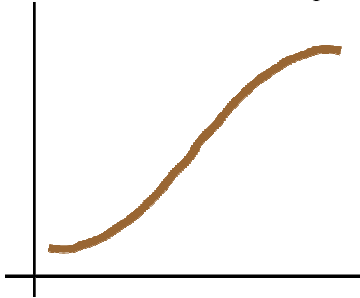
nt is the network time depending on the network infrastructure

It has to be noticed that dpt value will be significantly smaller than $QryTime(\text{sizeofDB})$ value. The nt value depends on the network infrastructure goodness.

11.3.7.2 Capacity Tests

The meaning of the following tests has been introduced at the beginning of the current section.

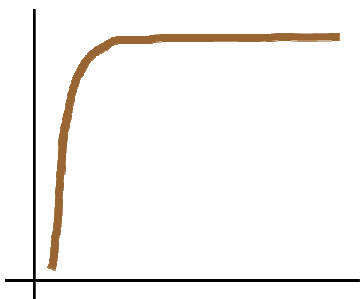
Load test: x = number of request, y = time



Load Test

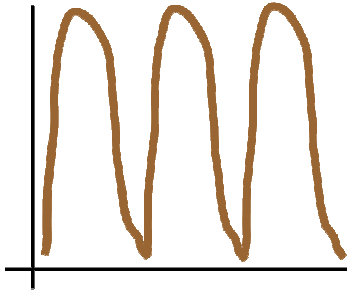
Expected trend for load test (number of requests per time)

Stress test: x = number of request, y = time



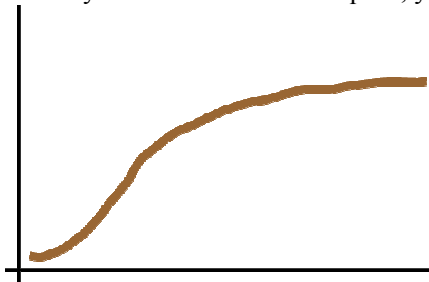
Stress Test

Spike test: x = number of request, y = time



Spike Test

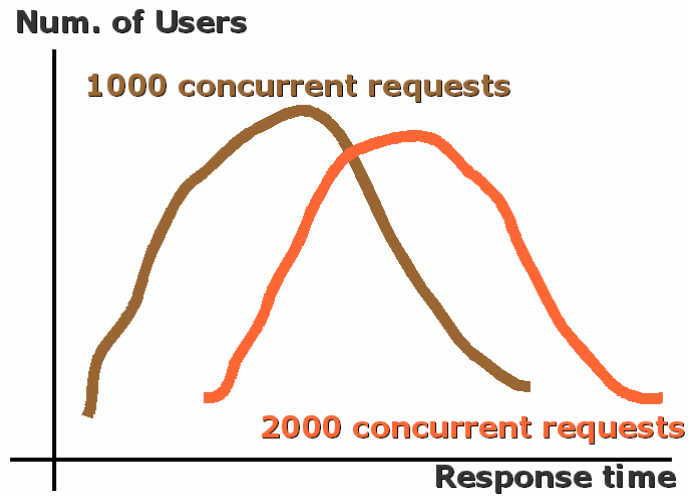
Stability test: x = number of request, y = time



Stability Test

11.3.7.3 Concurrency Tests

X = Response time, Y = number of users for a fixed number of concurrency request.



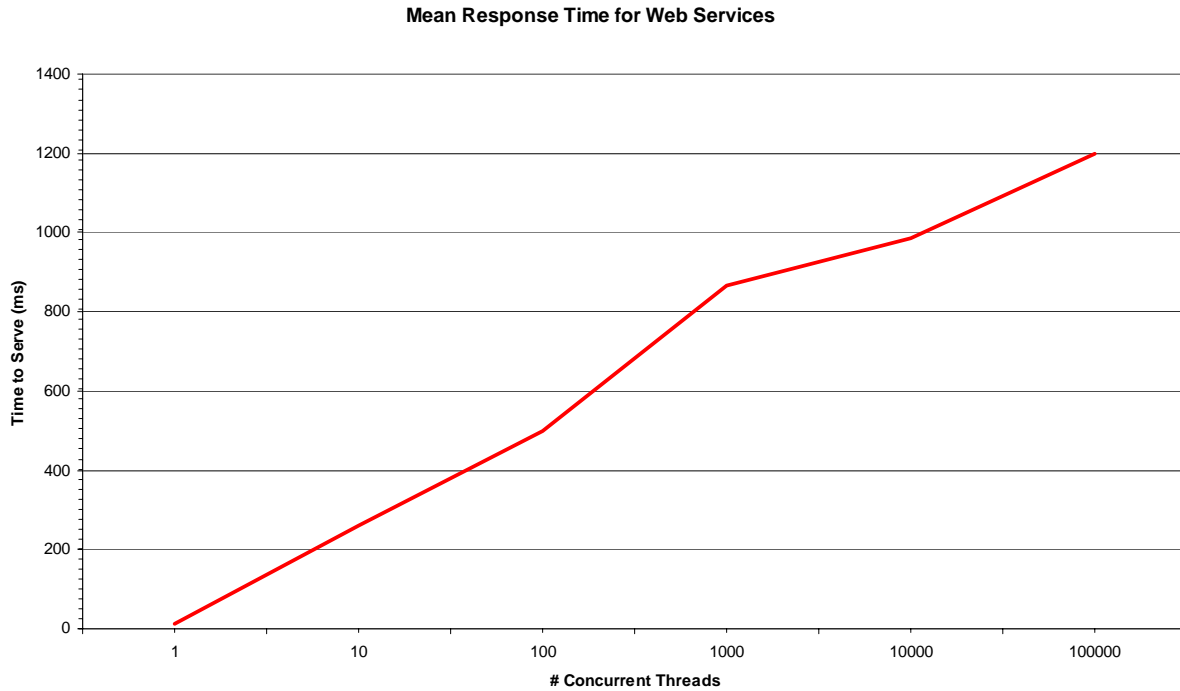
Number of Users per Response Time for fixed number of concurrent requests (1000 req. brown coloured and 2000 req. orange coloured)

11.3.8 AXMEDIS AXCV and AXS Web Services (FUPF)

This web service concerns user and tool certification and verification and retrieval of protection information from AXCS database.

11.3.8.1 Time Tests

Time tests concerns the average times of response per number of threads



To obtain the web service response time, it has to be considered also the performances of the related RegCert and Accounting databases. Each thread is associated to an instance of the web service which performs one or more queries over the database and some data processing. Web service time response results as:

$$\sum_i (QryTime_i(\text{sizeofDB})) + dpt + nt$$

where

$QryTime(\text{sizeofDB})$ is the query time depending on the size of the db

i is the number of query

dpt is the data processing time

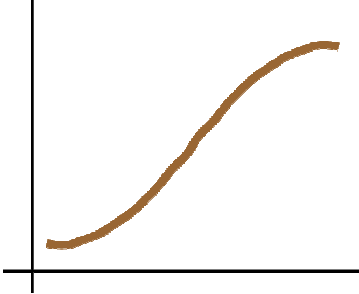
nt is the network time depending on the network infrastructure

It has to be noticed that dpt value will be significantly smaller than $QryTime(\text{sizeofDB})$ value. The nt value depends on the network infrastructure goodness.

11.3.8.2 Capacity Tests

The meaning of the following tests has been introduced at the beginning of the current section.

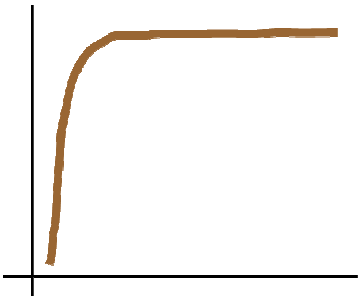
Load test: x = number of request, y = time



Load Test

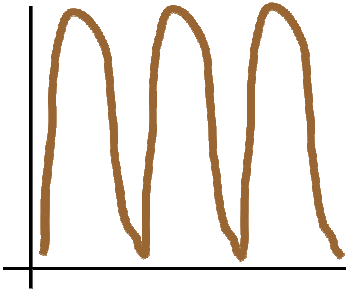
Expected trend for load test (number of requests per time)

Stress test: x = number of request, y = time



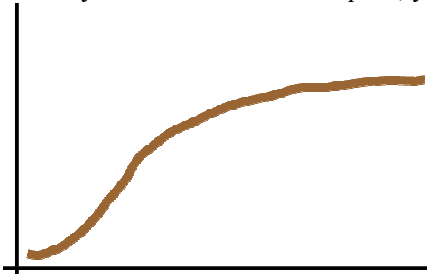
Stress Test

Spike test: x = number of request, y = time



Spike Test

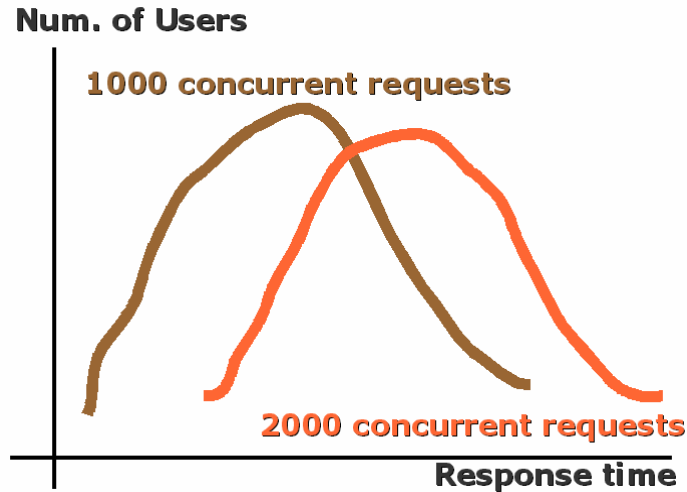
Stability test: x = number of request, y = time



Stability Test

11.3.8.3 Concurrency Tests

X = Response time, Y = number of users for a fixed number of concurrency request.



Number of Users per Response Time for fixed number of concurrent requests (1000 req. brown coloured and 2000 req. orange coloured)

11.4 AXMEDIS PMS (FUPF)

11.4.1 AXMEDIS PMS Server License database (FUPF)

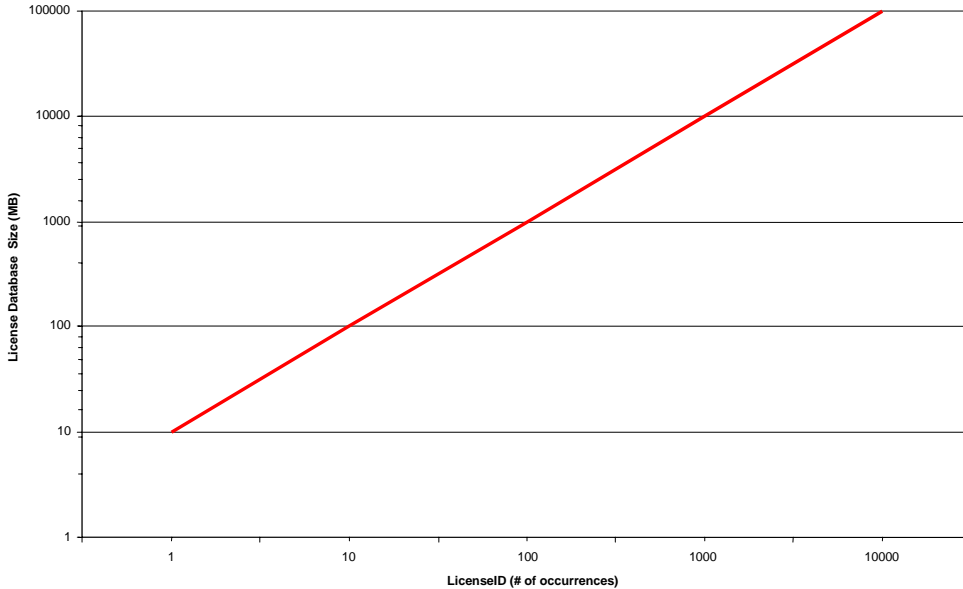
11.4.1.1 Database size / # of tuple

This test scenario checks the relationship about database size and the number of tuples contained. License database tables find their reference field in LicenseID field. This field is the primary reference for all test realized for this database.

Database size is directly dependent from LicenseID number of tuple. The related graphic will be always of type : $DBsize = A * (\# \text{ of LicenseID-tuple})$ where A is the average size of a LicenseID-tuple. The graph of this database will result something like this:

Graphics examples:

LogId graphic:



11.4.1.2 Search Time / # of tuple

This test scenario shows how performance in executing queries decreases when database size grows. These kind of tests are useful to determine the critical size of the database, beyond which size performances in query managing could increase even exponentially. In this scenario, results may experience substantial changes depending on the query used to perform the test. A reasonable thing to do in projecting tests is to apply the same kind of queries that are used by those web services that interact with the database. For License database, these queries are:

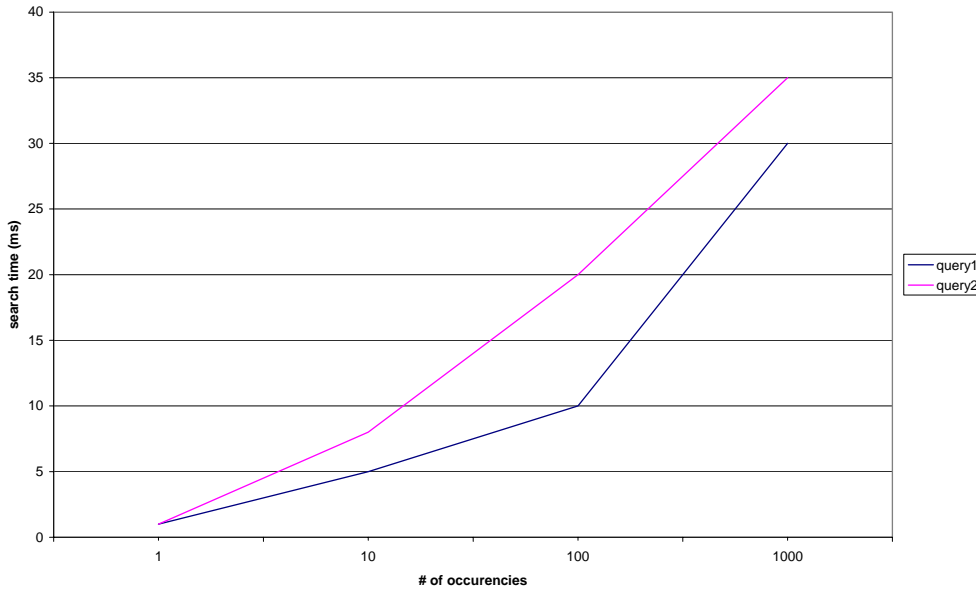
- Insertion of License related data into the License database
- Retrieval of License related data from the License database given a LicenseID
- Retrieval of Licenses related data from the License database given an AXUID, action and AXOID (multiple retrieval)

For each of the previous queries, a diagram will be created showing the number of occurrences of the target query vs. the search time used to retrieve the results.

This kind of test gives information not only about behaviours of different queries but also about evolution in search time when database increases its size, since a great number of occurrences as result for a query denote a large size of given database. To link number of occurrences to database size refer to “table size/# of tuples section”.

Graphics examples:

Query graphic for AXCSAccounting

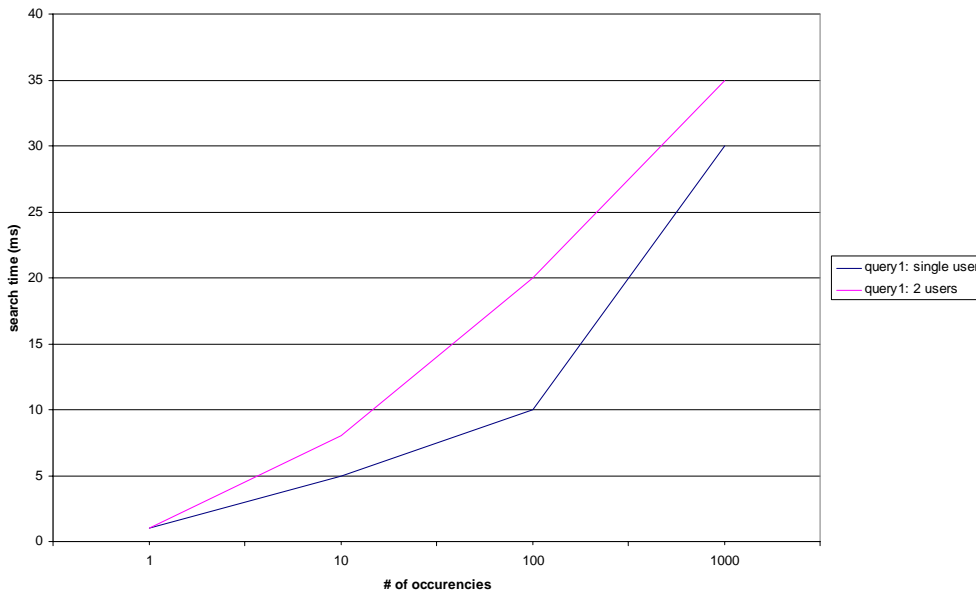


11.4.1.3 Single and Multiple access tests comparison

Another performance scenario that has to be considered is the case of multiple access. In a concurrent access scenario a decrease in DBMS performances is expected. To state how this reduction will affect the retrieval time for License database queries, it is necessary to repeat search time tests in a multiple access scenario, driven by the number of concurrent access to the database. One diagram for each query will be created as a result of this kind of measures.

Graphics examples:

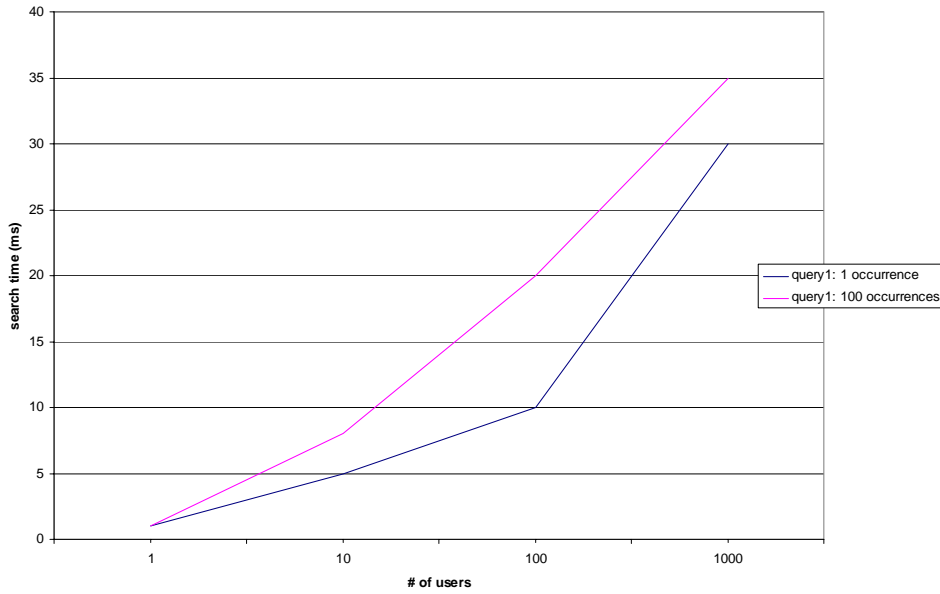
License Database Multiuser Query Graphic (1)



License Database Multiuser Query Graphic (2)

Y = search time

X = # simultaneous connections – fixed: # of occurrences

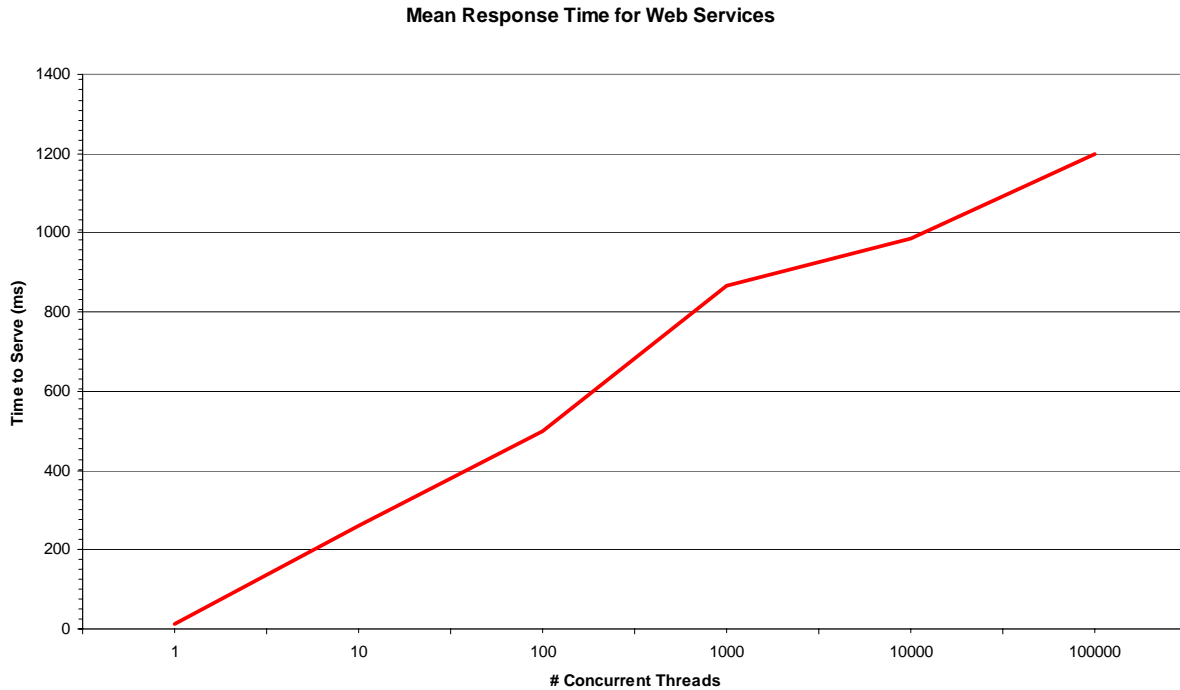


11.4.2 AXMEDIS PMS Server Web Service (FUPF)

This web service concerns user authorisation and protection information retrieval through AXCS.

11.4.2.1 Time Tests

Time tests concerns the average times of response per number of threads



To obtain the web service response time, it has to be considered also the performances of the related License database and AXCS-AXCV and AXCS-AXS web services. Each thread is associated to an instance of the web service which performs one or more queries over the database and some data processing. Web service time response results as:

$$\sum_i (QryTimePMS_i(\text{sizeofPMSDB})) + dptPMS + ntPMS + (QryTimeAXCS_i(\text{sizeofAXCSDB})) + dptAXCS + ntAXCS$$

where

$QryTimePMS(\text{sizeofDB})$ is the query time against the PMS db depending on the size of the db

$QryTimeAXCS(\text{sizeofDB})$ is the query time of AXCV and/or AXS against AXCS db depending on the size of the db

i is the number of query

$dptPMS$ is the PMS data processing time

$dptAXCS$ is the AXCS AXCV and/or AXS data processing time

$ntPMS$ is the PMS network time depending on the network infrastructure

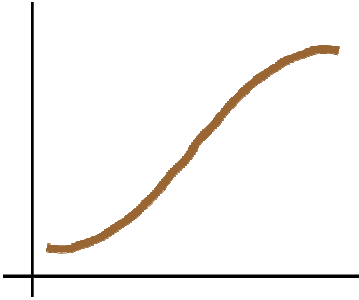
$ntAXCS$ is the AXCS AXCV and/or AXS network time depending on the network infrastructure

It has to be noticed that dpt value will be significantly smaller than $QryTime(\text{sizeofDB})$ value for both PMS and AXCS. The nt value depends on the network infrastructure goodness.

11.4.2.2 Capacity Tests

The meaning of the following tests has been introduced at the beginning of the current section.

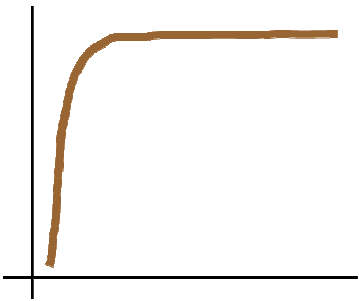
Load test: x = number of request, y = time



Load Test

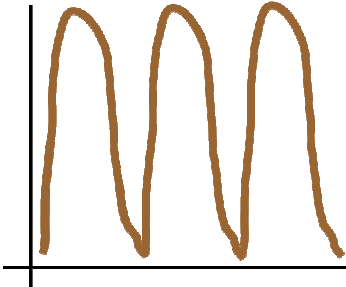
Expected trend for load test (number of requests per time)

Stress test: x = number of request, y = time



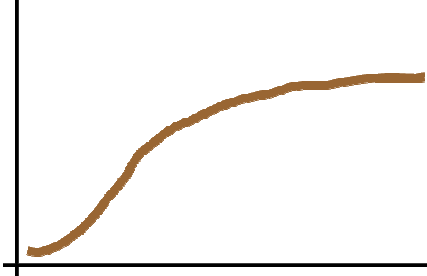
Stress Test

Spike test: x = number of request, y = time



Spike Test

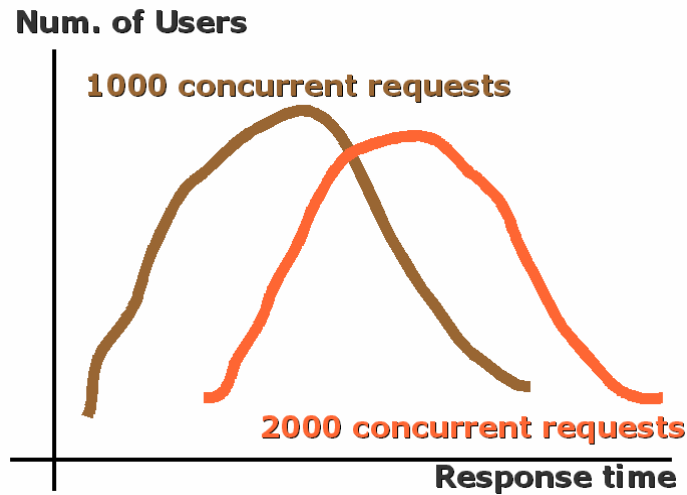
Stability test: x = number of request, y = time



Stability Test

11.4.2.3 Concurrency Tests

X = Response time, Y = number of users for a fixed number of concurrency request.



Number of Users per Response Time for fixed number of concurrent requests (1000 req. brown coloured and 2000 req. orange coloured)

11.5 AXMEDIS Database (EXITECH, FUPF)

In this section some performance figures for the Database related web services will be reported. In particular we will have 3 sections: (i) Loader, (ii) Saver, and (iii) Query.

It should be noted that Loader, Saver and Query are factory services or locally installed services in a Kiosk for example and therefore they have not a large number of concurrent users such as public webservice as Registration and other AXCS services. This consideration will be taken in account in defining the figures.

11.5.1 Loader Web Service [EXITECH]

Loader Web Service is capable to return the URL to be downloaded on the basis of a given AXOID and version. Its performance such as for all the database query are mainly affected by the number of item to be checked. The reference parameter will be the number of record in the VersionHistory Table that collects all the versions of all the objects.

The tests will be performed by fixing the number of records in this table and issuing a concurrent number of requests and registering the mean response time with confidence interval for the requests.

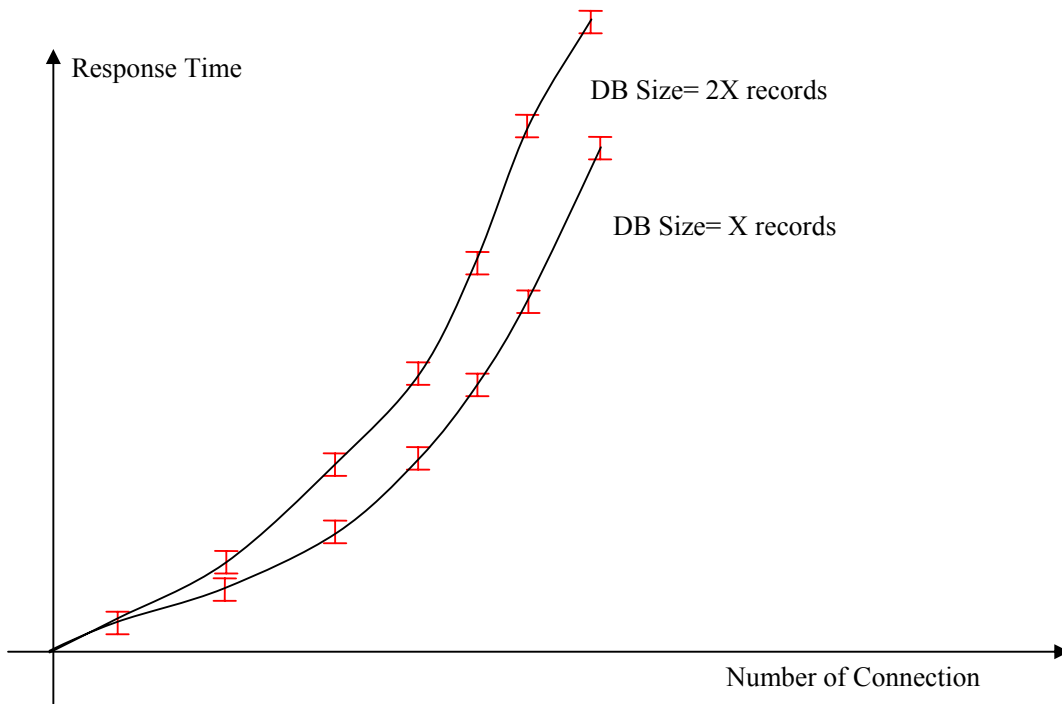
11.5.1.1 Response Time for Number of connection

In this metric the number of records in the Version History is fixed and we will obtain different figures for each number of records:

The execution of the test will be:

- Fill the DB with the fixed number of record
- Execute a test with a fixed number of concurrent connection
- Register the mean response time with respect to the whole number of connection
- Repeat with a different number of concurrent connections
- Repeat all the previous for a different number of records

The aim is to obtain a graph like the following where on the X we have the number of connection and on the Y the mean response time with confidence interval. Different curves for different number of records will be obtained.



For this measure quite all the test reported in the introduction can be executed. Typical range for connection can be from 10 to 200.

11.5.2 Saver Web Service [EXITECH]

Saver Web Service is capable to index in the database an object provided as a file. Its performance such as for all the database software are affected by the number of item to be inserted and read, but in this case the main parameter to be considered is the size of the file in order to verify the impact of the file size on the indexing process. The reference parameter will be therefore the size of the file submitted supposing that the file is already in the filesystem and not remotely located, since a remote connection insert a noise that is not predictable.

The tests will be performed by fixing the size of the files and issuing a concurrent number of requests and registering the mean response time with confidence interval for the requests. It should be noted that each connection should store a different file in order to measure maximum effort.

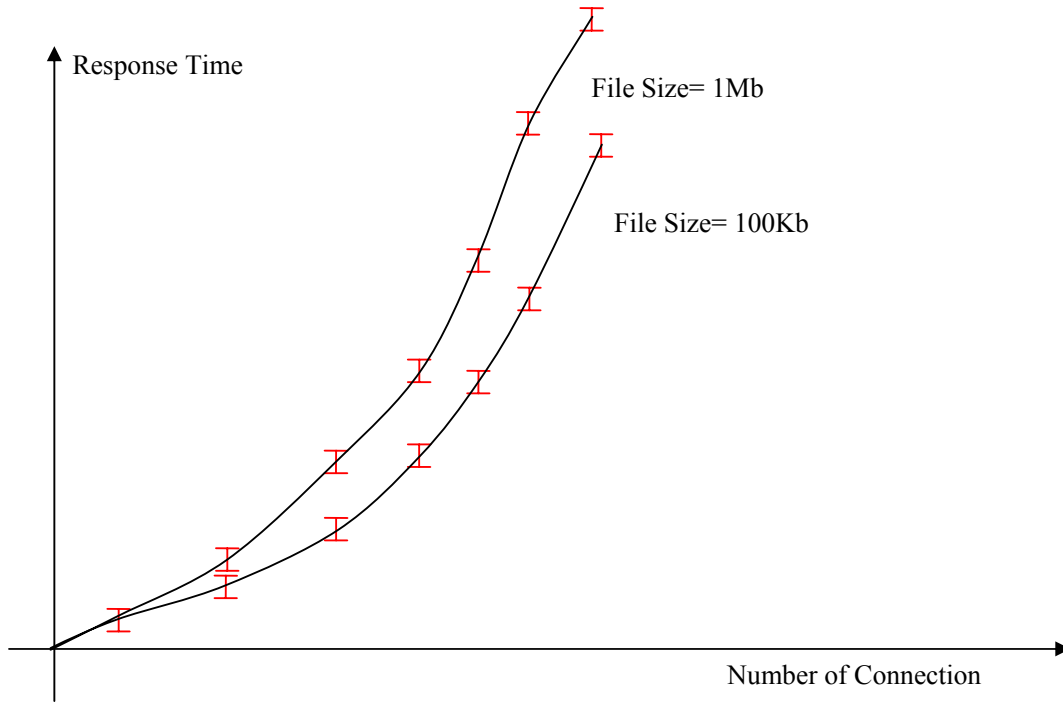
11.5.2.1 Response Time for Number of connection

In this metric the size of the file is fixed and we will obtain different figures for each size:

The execution of the test will be:

- Fill the file size
- Execute a test with a fixed number of concurrent connection
- Register the mean response time with respect to the whole number of connection
- Repeat with a different number of concurrent connections
- Repeat all the previous for a different file size

The aim is to obtain a graph like the following where on the X we have the number of connection and on the Y the mean response time with confidence interval. Different curves for different number of records will be obtained.



For this measure quite all the test reported in the introduction can be executed. Typical range for connection can be from 10 to 100, while file size should be 100K, 1M, 10Mb, 100Mb

11.5.3 Query Support Web Service for Metadata [EXITECH]

In this section the query Support web service for AXDB will be tested without considering the time of distributing the query by the MainQuery Support and collecting results, since these processes are affected by network time can be not predictable. Query Support for AXDB is capable to retrieve the AXODI that satisfy certain requests in terms of metadata and rights. Performance will be measured for metadata only in the section.

The main factors that affects the query are:

- number of record in the database (say number of records in the Did table);
- Type of the query that can be of three different typologies:
 - AXINFO only
 - AXINFO + DDMI
 - AXINFO + DDMI + OPTIONAL FIELDS

The tests will be performed by fixing the number of records in the DID table and issuing a concurrent number of requests and registering the mean response time with confidence interval for the requests. It should be noted that each connection should store a different file in order to measure maximum effort. This process has to be repeated for all the query types identified.

11.5.3.1 Response Time for Number of connection

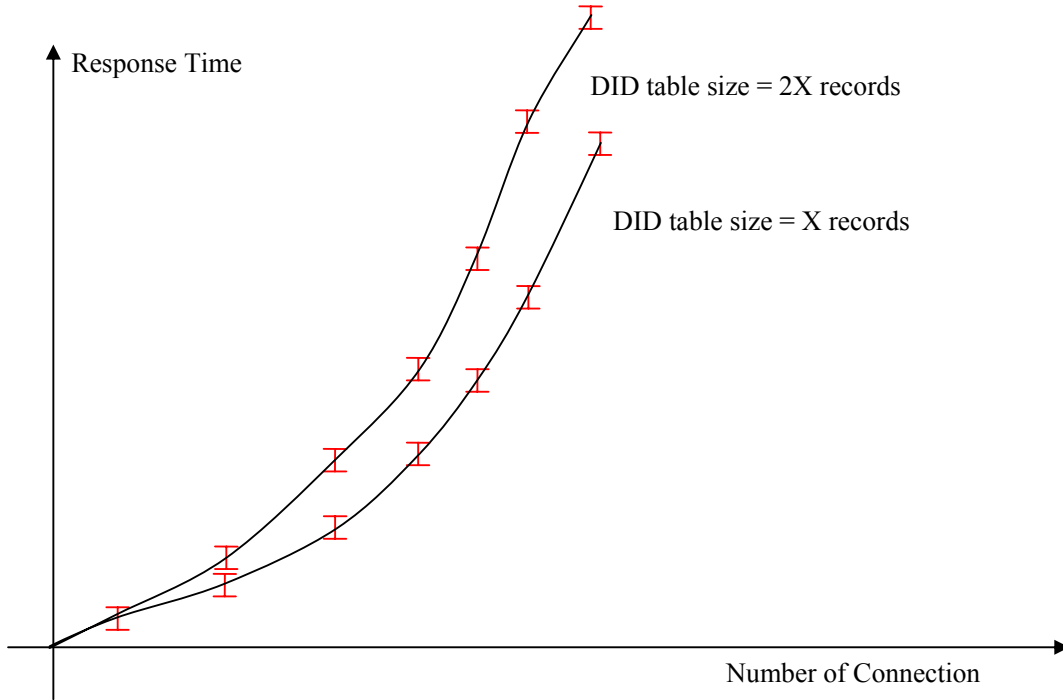
In this metric the number of records in the DID table is fixed and we will obtain different figures for each size. For each size different type of queries are issued:

The execution of the test will be:

- Fill the file size
- Execute a test with a fixed number of concurrent connection
- Register the mean response time with respect to the whole number of connection
- Repeat with a different number of concurrent connections

- Repeat all the previous for a different file size
- Repeat all the previous for a different query type

The aim is to obtain a graph like the following where on the X we have the number of connection and on the Y the mean response time with confidence interval. Different curves for different number of records will be obtained. One graph for each query type will be obtained.



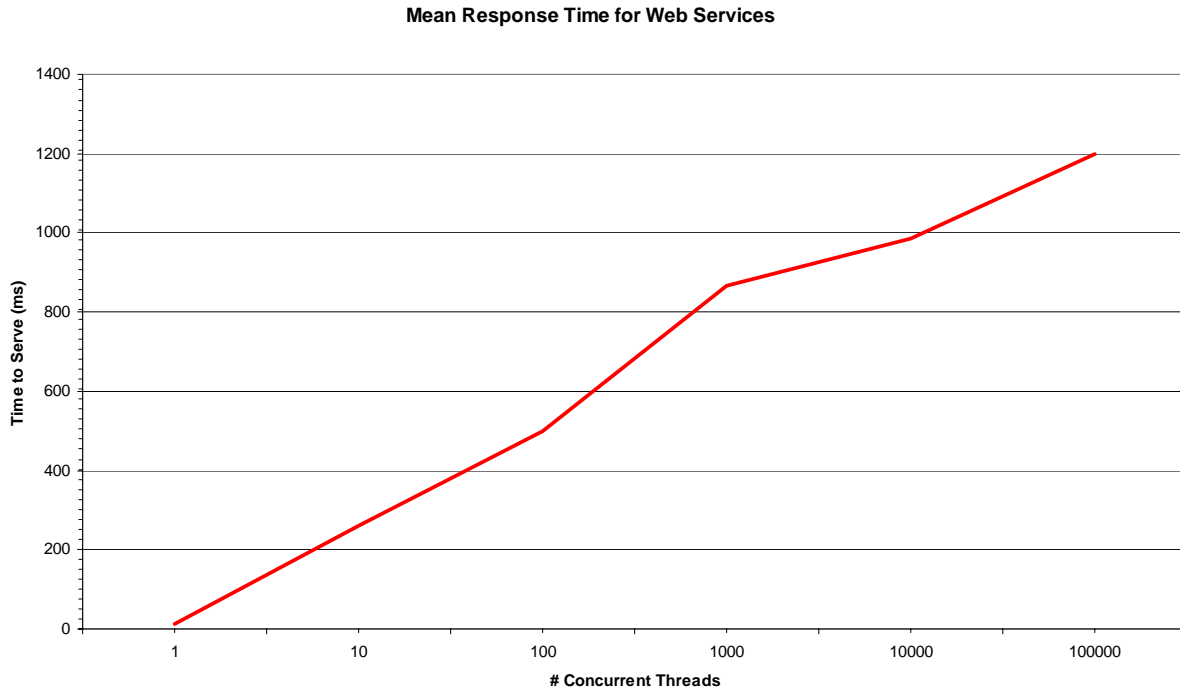
For this measure quite all the test reported in the introduction can be executed. Typical range for connection can be from 10 to 1000.

11.5.4 Query Support Web Service for PAR [FUPF]

This web service concerns query support for PARs. It uses the same License database as PMS Server, so refer to PMS License database section for the database tests.

11.5.4.1 Time Tests

Time tests concern the average times of response per number of threads



To obtain the web service response time, it has to be considered also the performances of the related License database. Each thread is associated to an instance of the web service which performs one or more queries over the database and some data processing. Web service time response results as:

$$\sum_i (QryTimePMS_i(\text{sizeofPMSDB})) + dptPMS + ntPMS$$

where

$QryTimePMS(\text{sizeofDB})$ is the query time against the PMS db depending on the size of the db

i is the number of query

$dptPMS$ is the PMS data processing time

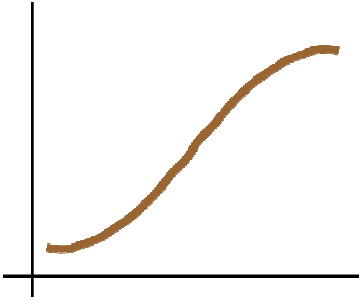
$ntPMS$ is the PMS network time depending on the network infrastructure

It has to be noticed that dpt value will be significantly smaller than $QryTime(\text{sizeofDB})$ value. The nt value depends on the network infrastructure goodness.

11.5.4.2 Capacity Tests

The meaning of the following tests has been introduced at the beginning of the current section.

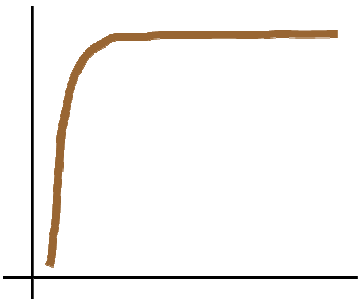
Load test: x = number of request, y = time



Load Test

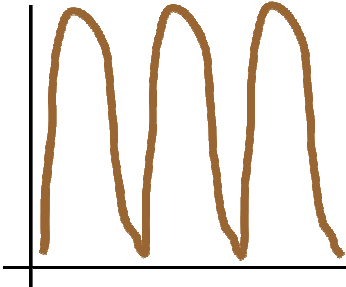
Expected trend for load test (number of requests per time)

Stress test: x = number of request, y = time



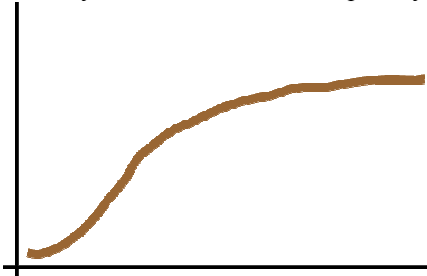
Stress Test

Spike test: x = number of request, y = time



Spike Test

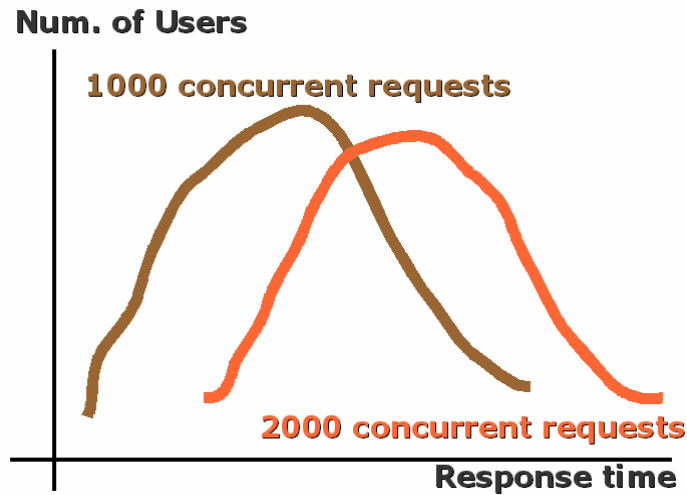
Stability test: x = number of request, y = time



Stability Test

11.5.4.3 Concurrency Tests

X = Response time, Y = number of users for a fixed number of concurrency request.



Number of Users per Response Time for fixed number of concurrent requests (1000 req. brown coloured and 2000 req. orange coloured)

11.6 AXMEDIS Content Processing GRID (DSI)

The performance analysis to be performed on the GRID infrastructure implies a set of tests and measures on different tools:

- AXMEDIS Rule Scheduler
- AXMEDIS Rule Executor, or Engine

Since the Content Processing GRID works as a service provider, it can be analysed in different work conditions by performing:

- **Load Test.** The number of rules to be run grows until the maximum tolerable number is reached. It is useful to understand the behaviour of the web service put through an overload of requests while operating in its normal environment.
- **Stress test.** The GRID is put through load of requests greater than the standard one. It is useful to understand the behaviour when the number of request increase highly and unexpectedly, till it reaches a great value (near the maximum tolerable number).
- **Spike Test.** The GRID is put through cyclic peaks of requests. A repeated great number of concurrency requests is useful to understand the behaviour when it is put through great load of requests.
- **Stability Test.** The GRID is put through a standard load of requests for a long time. This test is useful to detect unexpected problems such as memory leaks.

For each AXCP tool, different analysis can be performed as described in the following sections.

11.6.1 AXMEDIS Rule Scheduler

For each service of the AXMEDIS Rule Scheduler a precise analysis is needed:

- Time to service (from the Web Service) in different conditions of workload, service in terms of considering the rule to be scheduled.
- Reaction time from the Node (alarm or message) in different conditions up to the Maximum number of nodes manageable

- Reliability of connections among nodes and scheduler

11.6.2 AXMEDIS Rule Executor, or Engine

For each service of the Rule Executor/Engine a precise performance analysis is needed:

- Precision in maintaining the promise about the max % of workload
- Precision in maintaining the promise about the max % of the average workload
- Reaction time in leaving the CPU in the occurrence of an extreme need of the user.
- Measuring of the maximum:
 - HD space
 - MIPS accessible
 - Number of nodes
 - Number of rules manageable
 - Size of the rule in terms of code
 - Size of the rule in terms of memory space
- Estimation of precision for a CPU capability model

11.7 AXMEDIS Content Processing Algorithms (DSI, FHGIGD)

For each of the algorithms proposed the characterisation is needed to have a measure of the CPU time needed to execute the algorithm on a given resource. This means that the main parameters that influence the execution time of the algorithms have to be determined:

- Size of the file, digital resource
- Duration of the file: audio, and video
- X-size and Y-size, for images and video
- Sample rate for audio and video
- Bit per pixel, for images and video
- Bit per sample, for audio
- Time to disk access
- CPU performance
- Memory usage
- Etc.

The identification of the main parameters can be considered completed if the trend of the execution time as a function of them is linear. Thus the estimation of the K factor is needed. More complex and elaborated models can be defined in a second phase.

11.8 AXMEDIS P2P Server for queries (EXITECH)

Since in the P2P the AXDB Query support will be adopted, the same metrics reported in section 11.5.3 can be used.

11.9 AXMEDIS P2P AXEPTool and AXMEDIA Tools (DSI, HEXAGLOBE)

For the assessment of the performance capabilities the typical measures of the P2P can be used.

- The download time as a function of the number of sources and of the size of the file and of the population of peers.
- The time to restart the flow of data chunks of an interrupted download
- The time to publishing an Object.

11.9.1 AXMEDIS P2P Tracker (DSI, HEXAGLOBE)

For the assessment of the performance capabilities the typical measures of the P2P can be used.

- Time to provide a bittorrent information as a function of the number of bittorrent files in the Tracker Server
- Time to update a bittorrent information as a function of the number of bittorrent files in the Tracker Server
- Time to remove a bittorrent information as a function of the number of bittorrent files in the Tracker Server
- Time to provide a bittorrent information as a function of the number of simultaneous requests at the Tracker Server
- Time to update a bittorrent information as a function of the number of simultaneous requests at the Tracker Server
- Time to remove a bittorrent information as a function of the number of simultaneous requests at the Tracker Server

12 Guidelines for Video Demonstration acquisition (DSI: Nicola Mitolo)

This chapter specifies all options used to capture the screen using the software *BB FlashBack* that has been selected by the partners to realize the video demonstrations of AXMEDIS tools.

(<http://www.bbsoftware.co.uk/BBFlashBack.aspx>)

12.1 How to record with BBFlashBack Recorder

- 1) After the installation start “BB FlashBack Recorder” (or right-click on the Systray on the *BB FlashBack Recorder* icon, in the right side of the Windows toolbar);
- 2) Click on “Record a new movie”;
- 3) In the next windows select the folder and the filename; select the “Whole screen” option. Untick the “Show recorder toolbar” in the “Recorder Options” box and press the “More Options...” button (Fig1).

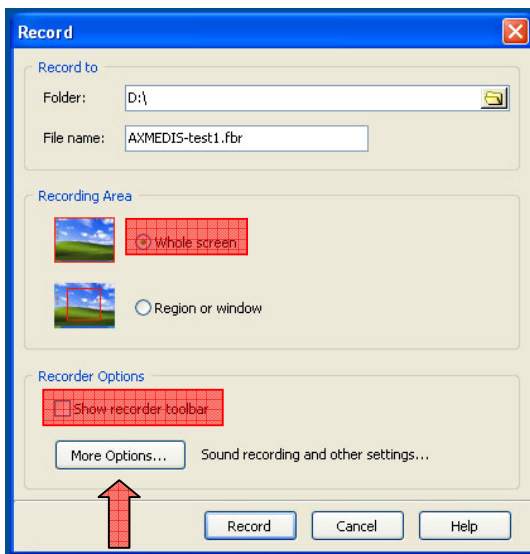


Fig. 1

- 4) In the *BB FlashBack Recorder Options* window, select the “Display” tab and tick the following options (Fig. 2):
 - Change resolution while recording
 - Video Mode: 800x600

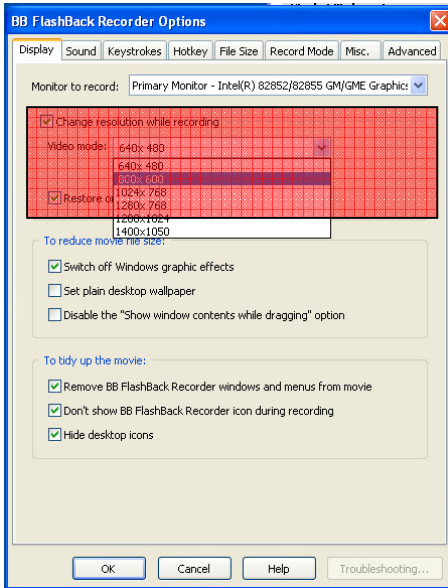


Fig. 2

- 5) In the “*To tidy up the movie*” option box (Fig. 3):
- Remove BB FlashBack Recorder windows and menus from movie
 - Don't show BB FlashRecorder icon during recording
 - Hide desktop icons

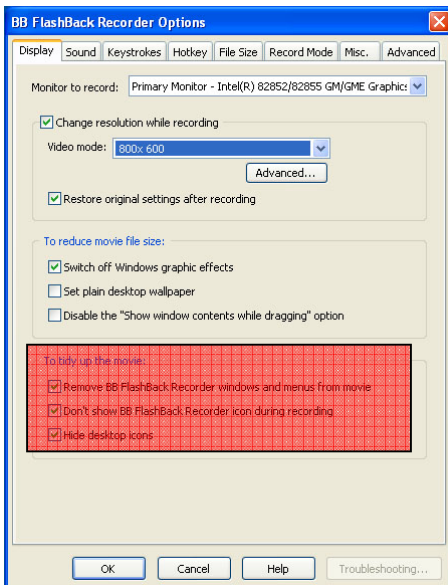


Fig. 3

- 6) Select the Sound tab (Fig. 4) and:

- Select the “Recorded sound” option
- Select “Microphone” as Sound Source (correct the Rec Level if necessary)
- In the Sound Format box select “44.1kHz”, Stereo and Use MP3 Compression “High Quality”

The audio can be also recorded in a second moment, after the screen recording, using an audio recording professional software as Adobe Audition or the freeware Audacity (<http://audacity.sourceforge.net/>). In this case it is very important to have the screen captured video and the audio perfectly synchronized.

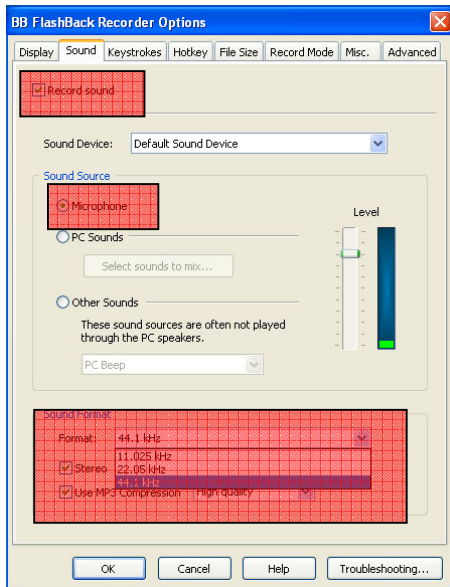


Fig 4

- 7) Select the “Hotkey” tab and set hotkeys for the “Record” and for the “Stop recording” functions (Fig.5);

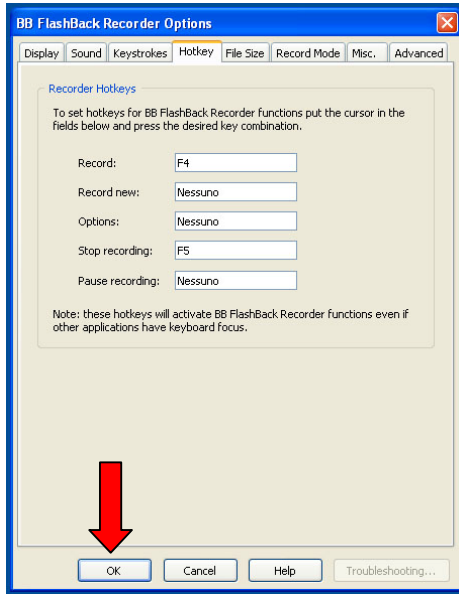


Fig. 5

- 8) Press "OK" in the "Options" windows and press "Record" (Fig. 6) to start the recording; also it is possible to start the recording using the appropriate hotkey.

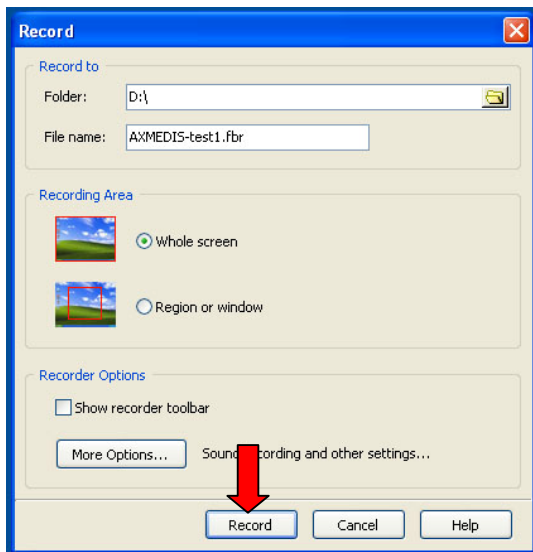


Fig. 6

- 9) During the recording all icons in the desktop will be hid. Press the key you set to stop the recording. The window in Fig. 7 will appear. Press "Yes" and the *BB FlashBach Player* will start immediately. In this manner you can check if the recorded movie is ok.



Fig. 7

Please, send the final movie in .fbr format to mitolo@dsi.unifi.it

12.2 Structure of video presentations

- **AXMEDIS Presentation:**
 - [AXMEDIS General Architecture](#)
 - [Examples of AXMEDIS Framework and tool Exploitation](#)
 - Part 1 - video on [General Overview](#)
 - Part 2 - video on [The Creation of the Object](#)
 - Part 3 - video on [Content Processing, Database and Query Support](#)
 - Part 4 - video on [Licensing](#)
 - Part 5 - video on [Protection aspects and the PandP Editor](#)

- **[AXMEDIS Editors the authoring tools:](#)**
 - [AXMEDIS Factory](#)
 - video on [AXMEDIS Editor](#) (DSI: Bellini)
 - Creating an New Object
 - Creating a nested Object (an object that contains other objects inside, difference layers)
 - Loading and Saving an object
 - Navigating in the object structure (AXMEDIS model)
 - Navigating in the object structure (MPEG-21 model)
 - The AXMEDIS Information, AXInfo
 - Metadata Editing and view
 - Metadata Mapper Editor (UNIVLEEDS)
 - Protection Information editing and view (FHGIGD)
 - Protecting an object and posting data on AXCS (FHGIGD)
 - Observing/interacting with Protected Objects
 - License Editing and view (FUPF)
 - Potentially Available Rights Editing and view (FUPF)
 - Posting License on the PMS Server (FUPF)
 - Application of Content Processing Algorithms
 - Object Behavior definition (EPFL)
 - Object Behavior execution and play (EPFL)
 - Connection to the database, make a query and selection
 - Using Automatic Formatting Facilities
 - Using Fingerprint Facilities
 - Using Adaptation Facilities
 - Using Watermark Facilities (FHGIGD)
 - Using Device Profile Facilities (IRC)
 - Using Ringtone Facilities (IRC)
 - SMIL Editor (EPFL)

- Preprocessor (EPFL)
- Etc.
- **AXMEDIS Players:** (DSI)
 - AXMEDIS Player for Windows (DSI)
 - Video Player
 - Image Player
 - Document and HTML Player (SEJER)
 - Audio Player
 - OSMO player (EPFL)
 - AXMEDIS Player as Active X for Internet Explorer (DSI)
 - AXMEDIS player as Plug in for Mozilla (SEJER)
 - video on [AXMEDIS PDA Player](#) (EPFL)
- **AXMEDIS Protection Tools Area:** (FUPF)
 - video on [License Editor](#)
 - video on [License to Contract and Vice versa](#)
 - video on [Grant Authorization](#)
 - verification of a license against PMS server
 - Protection Information Editor
- **AXMEDIS Content Processing Tools Area, AXCP Area:**
 - video on [AXMEDIS Rule Editor and debugger](#) (DSI: Bruno)
 - Loading and save a Rule
 - Creating a new Rule
 - Executing a Rule
 - Debugging a Rule
 - Using content processing algorithms into rules
 - Accessing to AXMEDIS object elements
 - Manipulating AXMEDIS object metadata
 - Making queries and selections
 - Accessing to AXMEDIS database
 - Moving objects among different AXMEDIS databases
 - Working on Licenses
 - Working on Protection Information
 - Protecting Objects
 - Loading and saving objects on disk
 - Loading and saving objects via FTP
 - Accessing to ODBC databases
 - Accessing to WEBDAV databases
 - Accessing to HTTP information
 - Exploiting Web Services WSDL
 - AXMEDIS GRID
 - AXMEDIS Stand alone nodes
- **AXMEDIS Content Processing Algorithms and AXCP Plugins:**
 - Adaptation Algorithms
 - Video on Audio adaptation (EPFL)
 - Video on Ringtones adaptation (IRC)
 - Video on Video adaptation (FHGIGD)
 - Video on Images adaptation (DSI)
 - Video on Documents adaptation (DIPITA)
 - Video on Metadata adaptation (UNIVLEEDS)

- Video on License adaptation (FUPF)
 - Fingerprint Algorithms
 - video on [Audio fingerprint](#) (EPFL)
 - video on [Video fingerprint](#) (FHGIGD)
 - video on [Images fingerprint](#) (FHGIGD)
 - Video on general data fingerprint (FHGIGD)
 - Extractor of Descriptors Algorithms
 - video on [Document descriptors extractor](#) (DIPITA)
 - Video on Audio descriptors (EPFL)
 - Video on Video descriptors (FHGIGD)
 - Video on Images descriptors (FHGIGD)
 - Watermarking Algorithms
 - Video on Audio watermark (FHGIGD)
 - ...
 - other Algorithms:
 - ...
- **AXMEDIS Database and query Support Area:**
 - video on [AXMEDIS Database Administrative Interface](#)
 - video on [AXMEDIS Query Support](#) (EXITECH)
 - load and save of objects
 - make a query
 - get query results
 - navigation in the database
 - AXMEDIS Selection Editor (DSI)
- **AXMEDIS Publication Area: (UNIVLEEDS)**
 - AXEPTool P2P B2B Tool
 - AXMEDIS Tool, P2P for Consumers
 - [AXMEDIS Program and Publication Tools](#), video on [Program and Publication](#)
- **AXMEDIS Reporting, Accounting (EXITECH):**
 - [Using the Action Logs about the exploitation of rights](#)
 - Video on CAMART:.....
 - video on [AXMEDIS Accounting Information Integrator](#), Administrative collection of data
- **AXMEDIS Certifier and Supervisor DSi: Martini, Chellini):**
 - video on [Administrative User Interface](#)
 -
- **AXMEDIS Workflow Tools area (IRC: Badii):**
 - Open Flow Based Management
 - video on [Workflow integration - 1](#)
 - video on [Workflow integration - 2](#)