Deliverable D8.2 RST2

Restoration Management Tool

DOCUMENT IDENTIFIER	PS_WP08_HSA_D8.2_RestorationManagementTool_v1.1
DATE	14/03/2006
ABSTRACT	Description of Restoration Management Tool in its version 1.1, published after the end of the project.
KEYWORDS	RMT, RSS, Communication Library
WORKPACKAGE / TASK	WP08 – Restoration Management Tool
AUTHOR, COMPANY	HSArt (Plaschzug/Höller), IT Innovation (Stuart Middleton)
NATURE	Other
DISSEMINATION	Confidential

DOCUMENT HISTORY

Release	Date	Reason of change	Status	Distribution
0.9	22/02/2006	Document First Draft	Living	Confidential
1.0	14/03/2006	Delivered	Complete	Confidential
1.1	02/04/2008	Document dissemination level changed to Public	Complete	Public

Table of Contents

1. Document Scope and Organisation

This deliverable describes the Restoration Management Tool (RMT) as it has been implemented in the course of the PrestoSpace project up to PM24.
In variation from the original contract, P23_HSArt has taken major responsibility for the development of the RMT and replaced P04_JRS.

RMT, Version 1.0 as finished and distributed to the project partners inside WA RES is basis for this document. Although the main application has been provided by P23_HSArt, there were major contributions from P04_JRS and P15_ITInnov, who provided the D2 API and communication library respectively. The other partners, especially the user partners have provided valuable comments.

This document first introduces the RMT from an operator's perspective, outlining the user interface and operational procedures. It then follows through to discuss the technical details of the communications layer. The final section reports on the integration activities between the RMT and RSS developers.

2. Executive Summary

RMT in Version 1.0 is a Windows-XP application and considered as the main operator interface for all restoration tasks inside the PS factory. It can be operated on any workstation that has (mounted file) access to the content repository. The RMT communicates, by means of a communication library based on GSOAP, with the various Restoration Subsystems (RSS) available in the PS factory.

Although communication is generic, the main focus in the development has been the integration of 3 types of RSS, namely DIAMANT, BRAVA and AUDIOCUBE. Those RSS are described in D10.1, D10.2 and D10.3.

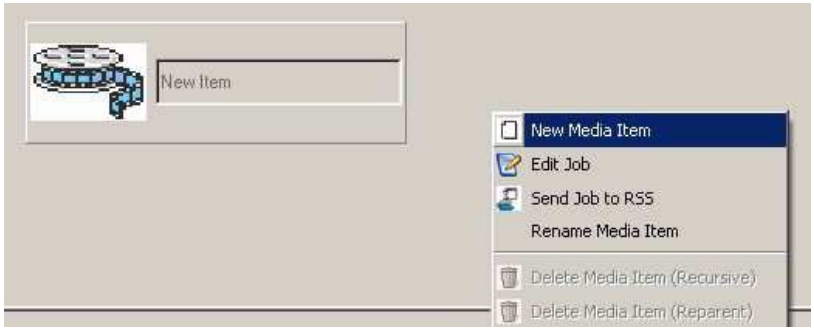
The RMT is presented from an operator's viewpoint, detailing the user interface artefacts they must use and the capabilities that are offered to them. The underlying communications layer is looked at in some detail, explaining how the RSS applications will receive and process tasks provided by the RMT via GSOAP-based web services. Rationales behind design choices are included.

3. Workflow

In the course of the PS project, WA RES partners with involvement of the user partners have determined a workflow and requirements from the perspective of the restorers. As a consequence the RMT-architecture has been presented in a first draft in D3.1.
This chapter gives an overview of the RMT supported workflow as it is perceived by the central restoration operator.

3.1. Content Acquisition

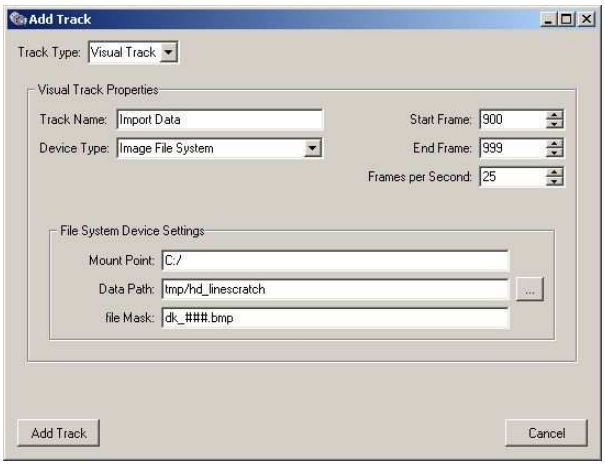
The RMT operator works with media items. An empty media item can be created inside a repository, folder and project to follow a structured approach. To do this the operator can either use the menu “Media Item”->“New” or the context menu “New Media Item”.



In order to fill the empty media item with content, the operator has to add and define data tracks for the media-item. Data tracks have a specific type and the data of the track can be stored on a specific device.
The following table gives an overview of the relation between supported track and devices:

	Visual Track	Audio Track
File System Device	N/a	WAV files
Video File System Device	Packed formats: mpeg, avi, quicktime	N/a
Image File System Device	Single image file formats like: TIF, TGA, BMP, DPX, CIN,…	N/a

The location of the data is selected and – dependent on the device – additional meta-information is extracted (e.g. image icon).



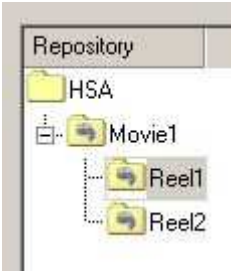
Finally if the media item is selected, the assigned track is visualised in the RMT, additional tracks can be added by the same procedure.

	Type	Label	Device	Info
1	VisualTrack	Import Data	ImageFSDevice	D:/Prestospace/PrestoSpace_RMT_07/RMT_v07/D2_Data/TestImages
2	AudioTrack	MP2-Audio	AudioFSDevice	C:/tmp/Title 01.mpg

Alternatively there is a drag & drop function for some of the devices available; in this case there is no need for creation of an empty media item. Dragging the content from the windows explorer and dropping it in the RMT's main widget will automatically result in creation of a media item and the appropriate data track. Currently this is only possible for the “Image File System Device”.

3.2. Administration

The administration of media items is another important purpose of the RMT. As there is usually some buffering inside the RMT (meaning that not all what is acquired is immediately send to restoration), there is some hierarchy necessary to store and administrate data.
Inside a repository (that is represented by a XML-File) we have folders and projects. Whereas folders cannot contain media items, projects are the place where the data (items) are stored. Folders are visualised by a “classical” Windows folder symbol, whereas projects contain a film-reel symbol.



There is a set of functions available for media items; these are systematically described in chapter 4.3.2.

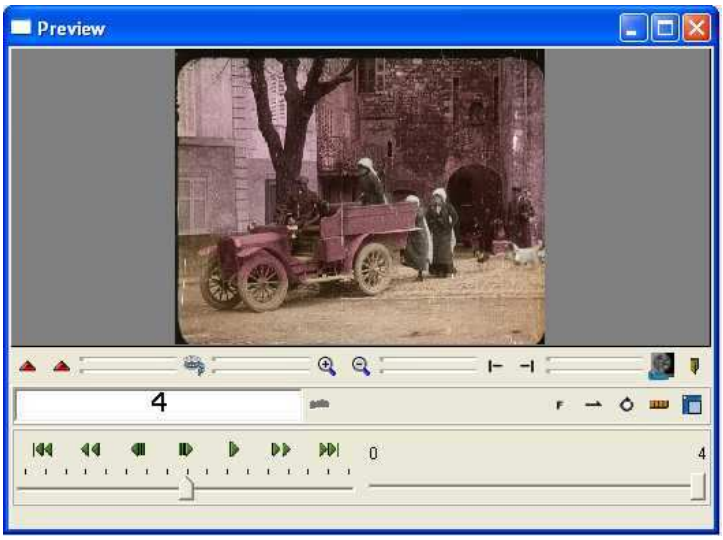
3.3. Restoration Preparation

Once the media item is available in the RMT it can be prepared for restoration. Although the details of such preparation might vary for different purposes, there is a set of common requirements that are supported by the RMT.

Inspection

Inspection of the material is the first step in the restoration preparation. For that purpose the RMT offers an integrated Player and support of external applications. Usually it is not the media item as such that can be inspected, but the data tracks associated with the item. V1.0 of RMT offers a:

- MoviePlayer
  the Movieplayer, if opened on the media item, takes the first visual track of the item and plays it as a sequence. Currently the player does not support avi or mpeg. If opened on a specific visual track the content of the track can be viewed as long as the format is compliant with the player's capabilities (single image file formats, quicktime).
- IrfanView (The image viewer could be configurable)
  IrfanView shows the first single image of the track.
- Standard Application
  Opens the Windows standard application to visualise the selected (or default) track. This is currently the only way to play audio tracks, avi- and mpeg files.



Restoration Plan

Any media item is associated with a specific restoration plan that is empty if the media item has been freshly generated. However, an external restoration plan can be imported and assigned to the media item.

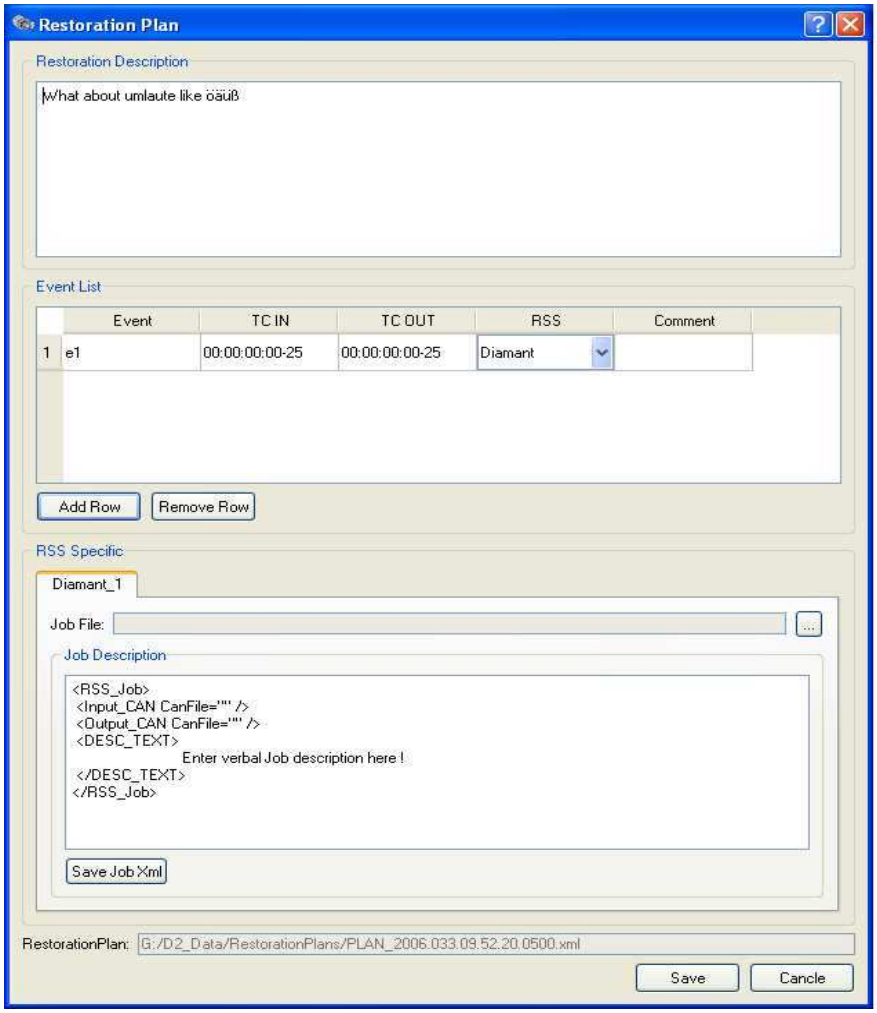
On the other hand the operator can manually create the restoration plan and/or use some information that has already been generated by any Restoration subsystem (RSS).

The restoration plan itself consists of 3 major sections:

- RestorationDescription (free text)
  This is a free word section where the RMT operator can give guidelines and general remarks on the media item and the required restoration planning. This

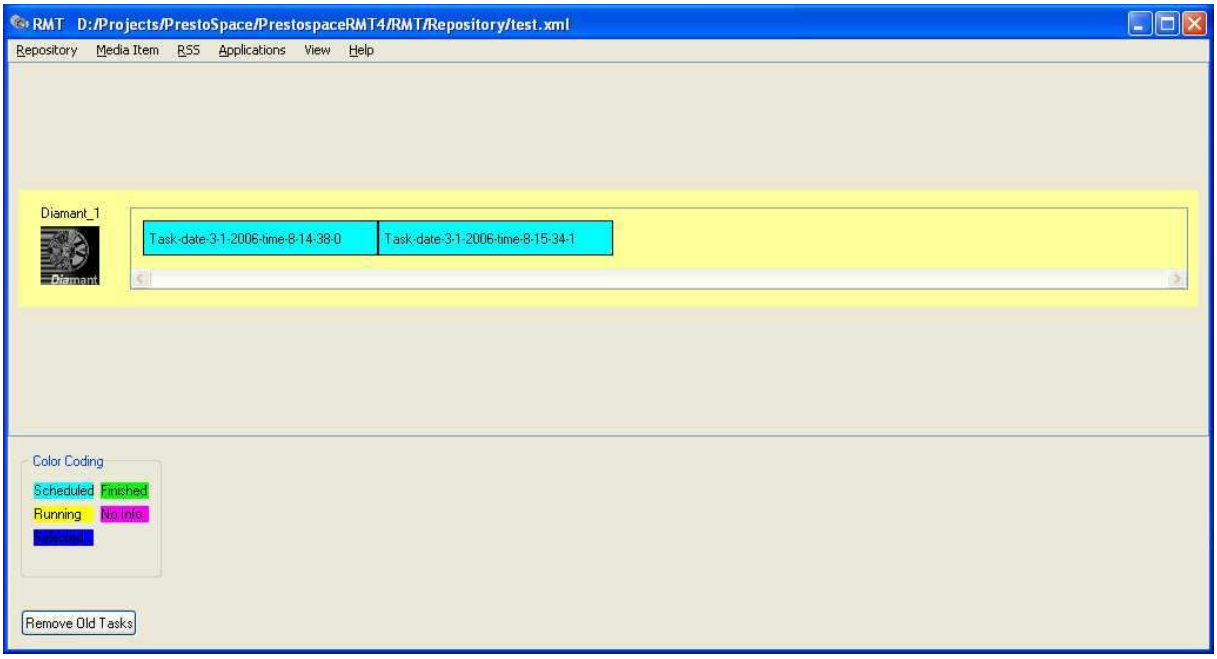
information might be important for any RSS operator, how and to which extend he should work on the restoration job.

- EventList
  This is a formal list of events that might be useful for the restoration process. As the type of event is free, there can be simple information like CUT and SCENE, but also some specific quality hints like SPLICE, BROKEN_IMAGE…
  Such information might come from various sources. Externally, if the restoration plan is imported (e.g. from an EDOB), manually if the RMT operator creates it or from an RSS (e.g.: if a specific RSS like BRAVA reports such an event list back).
- RSS-Specific
  Dependent on the type of RSS, there is a specific section of the restoration plan that might be used. The main purpose is that the RMT operator can assign some RSS type-dependent restoration commands. However in this case the RMT operator either needs not know the syntax of the RSS or he might use predefined default-jobs (e.g. DIAMANT pre-processing).



3.4. Restoration Control

The RMT operator administrates and centralises the utilisation of the various RSS's and the PS factory.
For that purpose the operator can start/stop restoration jobs. A restoration job is a set of 2 media items (original, result) and the extracted part of the restoration plan that is associated with the original media item. A restoration report, associated with the result item is completed by the RSS.
When a restoration job is submitted, some general parameters can be set (start time, end time) and an entry for the log is created in the restoration queue.
The RSS communicates with the RMT and picks up the restoration jobs, assigned to the RSS. Feedback on the progress/status of the jobs in the queue is available in the "Task management view" of the RMT (see below).



3.5. Quality Control

After completion of a restoration job, the RMT operator is verifying the result. He can furthermore decide if the media item should be sent to another RSS or re-sent to the same RSS.
The operator can check 2 things:

- Content
  Similar as in the restoration preparation, the data tracks can be inspected (standard application for both, visual and audio tracks; MoviePlayer for visual tracks). Comments, and if required a second restoration pass, can be done by editing the restoration plan of the resulting media item. If another pass is required, the operator simply creates another job entry in the queue after completion of the restoration plan.
- Restoration report
  The restoration report that is assigned with the resulting item contains all relevant information that has been provided by the RSS. The structure is similar to the structure of the restoration plan and contains a free text-section, an

event-related section and finally a RSS-dependent section. The RMT operator usually summarises and finishes the report by updating the free text-sections and the events.

Minor issues can also be directly solved from the RMT operator. For this purpose the integrated MoviePlayer is currently enhanced, to permit some editing/toolbox functions for manual correction. This is useful if there are a few and minor issues that can be ad hoc fixed and then resubmitted to the RSS. The paint-mode can be activated in the MoviePlayer at any time. In this mode it is possible to “clone” the content from any image of the sequence to specific part in the image to be corrected. This immediate fix is only possible for visual tracks.

Dependent on the size of the facility, there could be different operator roles:

- the person supervising the overall restoration facility, who assigns jobs and does QA after restoration was done and
- the restoration operator itself specialised on e.g. audio/film/video)

4. Operation

This chapter documents the RMT as it is perceived by the operator. It is formulated in a manner that it could directly be used as user-manual.

4.1. Setup

In this subchapter we drive you through the setup process in order to get the RMT principally working.

4.1.1. System requirements

Usually the RMT will be installed on a controller/operator workstation. The minimal requirements are quite moderate. The recommendations are:

- Windows XP (pro) SP2
- Min. 2 GHz CPU (AMD, INTEL)
- Min 1 GB RAM
- 80 GB hard disk
- network access

4.1.2. Specific pre-requisites

What is very important is the connection to central data storage. In order to make sensible work with the tool, the operator needs fast (high bandwidth) access to the stored digital content of the PS factory. As this issue is not part of this deliverable, the formal requirement is a network access (by mounted file system) to the central content storage.
As the RMT is communicating with various restoration devices (we call them from now on RSS), there must be some available RSS in the same network.
It is very important that to know that optimal RSS-integration is based on a same/similar view on the centrally stored content.

Also the RMT requires some underlying PS libraries and functions that come along with the RMT distribution. Currently these are:

- D2 Libraries
  This provides the basic restoration infrastructure that has been developed by P04_JRS in the framework of the PS project.
- Communication Library
  This provides the communication infrastructure that has been developed by P17_ITInnov in the framework of the PS project. It permits the communication between RMT and RSS and task management.

4.1.3. Installation

RMT V1.0 is available as self-extracting ZIP-File with the name “Prestospace_RMT_V10.exe”.
The installation is started by double-click on the zip-file. The recommended destination folder is: “c:\”.

The installation process creates a main directory named “Prestospace_RMT_10” that contains all necessary files and resources to run the RMT.

4.1.4. Configuration

A manual configuration process is necessary if you did not install the RMT into its recommended “c:\” path, but into <INS-DIR>:
The file “<INS-DIR>\Prestospace_RMT_10\readme.txt” contains a description on how to configure it.
Shortly said the following steps need to be done:

1 Edit “<INS-DIR>\Prestospace_RMT_10\startRMT.bat”
Adapt the D2HOME variable, so that its value points to the D2 directory in your installation directory. Default is: “c:\Prestospace_RMT_10\D2”.

2 Edit “<INS-DIR>\Prestospace_RMT_10\\RMT_v10\bin\config\RMT_ExternalApplications.xml”
Adapt the path of the player according to your installation directory. Default is: “c:\Prestospace_RMT_10\MoviePlayer\”.
Do the same procedure if you want to use a viewer for static images. Default (for IrfanView) is: “c:\Program Files\Irfanview\”.

You can also exchange the proposed application in the configuration process. Just the Executable- and Path- Variables need to be correctly filled.

As a last step you should create a Desktop-symbol (i.e.: link) for the application “startRMT.bat”:

1 Click right on the startRMT.bat and move the mouse to your desktop;
2 Select “Create Link”;
3 Rename the linked application to RMT

4.1.5. Testing

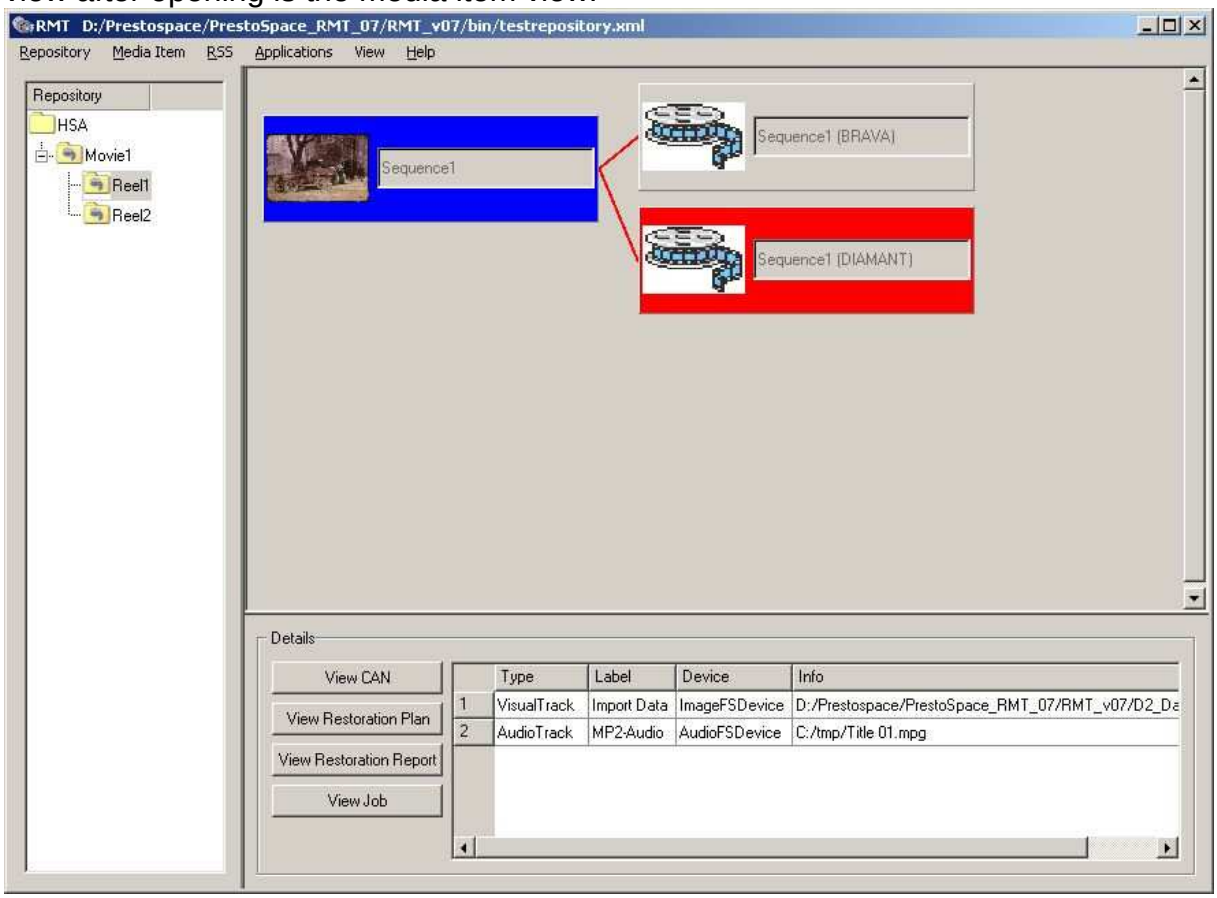
Click on the Desktop-symbol that has been created in the previous step. The RMT application should open itself.
If there is any unrecoverable malfunction please use the following contact to get immediate support:

HS-ART Digital Service GmbH
Dietrichsteinplatz 3

A-8010 Graz
support@hs-art.com
Tel.: +43 316 915998 0

4.2. Interface

When the RMT is started, it offers the functions that are required to manage the workflow, as described in chapter 3.
This is achieved by 2 major views (media item and task management). The default view after opening is the media item view:



Therein the operator can view, manage and administrate media items. In order get an overview of the attached and available RSS, the operator has to change into the task management view that is shown in chapter 3.4.

4.3. Functional Details

4.3.1. Repository

In the context of the RMT, such a content collection is called “Repository” which is the highest hierarchy level. All information about a certain repository is available in a XML-file that defines that universe of data.
The functions available for the operator are:

- *Open*
  Opens a file-dialog for browsing the filesystem and searching for repository files.
- *Save*
  This function simply saves the current state of the repository to the opened repository file. The current, open repository file is displayed in the headline of the main user interface window.
- *New*
  Closes any open repository and opens a new, un-named one. In the saving phase a name has to be given.
- *Save as…*
  The operator is prompted to enter the filename/location of the repository file that should be created.

4.3.2. Media Item

Media items are organised in folders and projects. Both hierarchy levels are available within a repository. The hierarchical view is visualised in the left hand widget of the RMT application, whereas media items itself are in the main widget of the RMT.

- Folder
  A folder in the context of RMT is a structure that is able to hold multiple projects, but no individual media items.
- Project
  A project in the context of RMT is a structure that can hold a collection of media-items that might, but need not, depend on each other.

Media items itself are visualised in the RMT main widget by an icon, if the primary data track is video/image and a name. In case there is no possibility to extract/display the icon than a “default” icon is being displayed.

Dependencies
Media items might depend on each other. Generally, a dependent media item does not necessarily contain all its data directly. Some content (e.g.: tracks or even single samples) might simply be a reference to the media item, that is the master.

Functions/Actions
There is a set of actions/functions on media items available. Those actions are for 3 major purposes (management, operation and restoration).

The management actions on a media item are:

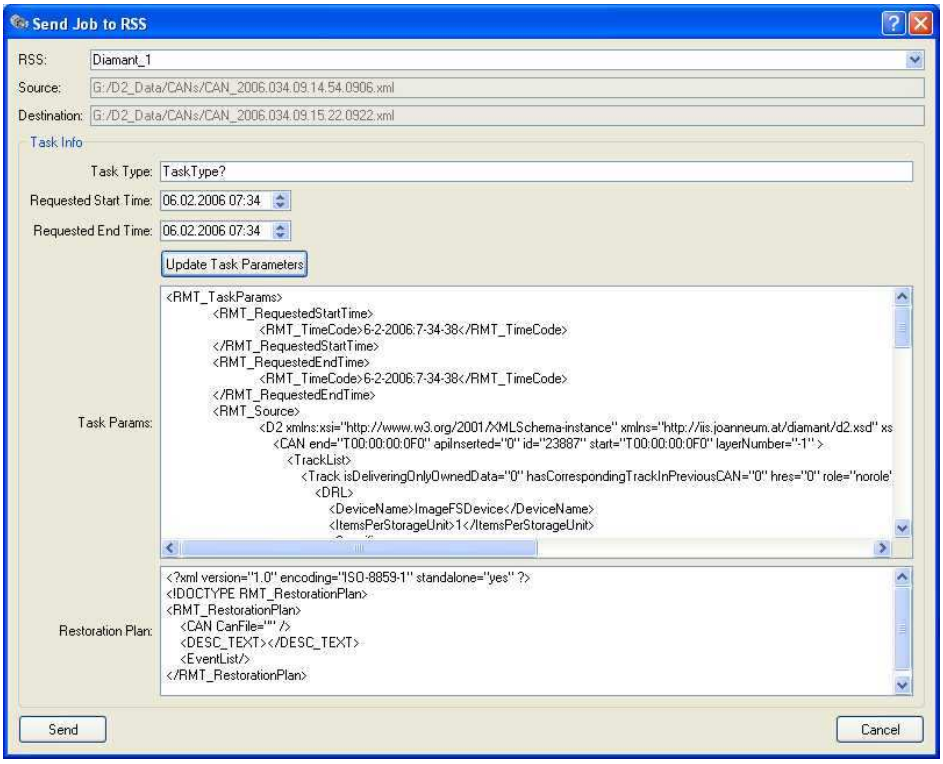
- Select
  A selected media-item is visualised by a highlighted color (e.g.: blue).
- Lock
  A media-item can be manually locked, but is also automatically locked if it is in the restoration queue. Locking prevents access conflicts on the media item. Visualisation of a locked item is by color (red).
- Unlock
  Usually any processed media item is unlocked after the completed restoration process. However, as the operator is the master he can always unlock a media item manually - but he has to decide if this creates any side-effect and conflicts.
- Delete (Recursive)

This action deletes the selected media item and recursively all dependent items.

- Delete (Reparent)
  This action deletes the selected media item, but does not touch the dependent items; reparentisation is applied if the selected item is parent of another item.
- Rename media item
  Simply changes the name of the item.
- Add Track
  This action opens a dialog for addition of another data track into the selected media item. They type of track (visual, audio) has to be selected as well as the location of the track-content (filename, directory…).

The restoration related functions are:

- Edit Restoration Plan
  This is one major function in the restoration workflow. The restoration plan consists of several sections (general, specific and RSS-dependent). This function opens an editor to manually add and change the restoration plan for the selected media item.
- Send Job to RSS
  This function opens a window, where the operator can make the final definitions of a job that is sent to a RSS. Such final definitions include the selection of RSS, partial job selections, start/end time, etc…
  Currently this view is (for RSS debugging feasibility) quite technical, but this will be changed in a productive version.



The operational actions on a media item depend on the configuration and setup of RMT. Operational actions are realised by so called external applications. See chapter 4.3.4 for more details.

4.3.3. RSS

This menu shows the list of available RSS. Currently there are 3 types of RSS available. These are:

- DIAMANT
- BRAVA
- AUDIOCUBE

However, as the library and specification of an RSS is open, other RSS types might follow. There is no restriction, that there is only one RSS from each type. Consequently the RMT supports several RSS per type.

4.3.4. Applications

This menu lists all available external applications that are available inside RMT. This list is configurable and open, as these functions add the real value to the RMT-operator.
Currently the following are configured:

- MoviePlayer
  The MoviePlayer is an application to monitor visual data tracks. Currently we use a player that originates from the DIAMANT restoration system, but has been adapted to fit into the RMT. Soon, this will be extended by a painting application that permits immediate interactive manipulation.
- Irfanview or Photoshop
  This is a sample for any external (command line controlled) application, that can be used directly from inside RMT. In this sample we use an application for single image editing.
- StandardApplication
  The standard application is specific Windows-application that is defined for a certain filetype. Thus, if you want to list a WAV-file the standard application for a WAV-file is being opened with the file. Same is true for any underlying filetype (e.g.: MPEG2, MOV…).

4.3.5. View

The view menu permits to switch between media item view and task management view. As the media item view is the default, this menu is mainly used to see and monitor the status of the restoration jobs that have been submitted/distributed by the RMT operator.

4.3.6. Help

The functions available are:

- Contents
  Not implemented yet, will be available in a later version.
- Index
  Not implemented yet, will be available in a later version.
- About
  Shows version and copyright information.

5. Communication Library

5.1. Overview

The communications infrastructure is based on web services and provides a mechanism for the restoration sub-systems (RSS) to request new restoration tasks and report restoration results. The infrastructure maintains lists of known restoration sub-systems, devices and reports. The operator can control the overall restoration workflow and specify the next restoration activities that will be enacted; new restoration activity will result in new tasks being assigned to the restoration sub-systems.
The restoration management tool communications library is statically linked to the restoration management tool interface code and provides the 'behind the scenes' functionality. The communications library includes a client side library for individual restoration sub-systems developers to statically link to, allowing easy access for legacy applications to the new web service functionality.

5.2. Architecture

The restoration management tool is based on a web services communication platform, implementing a polling-based control protocol. The communications infrastructure is written in C++ and uses the GSOAP web services libraries coupled with Xerces XML libraries. Figure 5.1 shows this architecture and the partners responsible for development of each module.

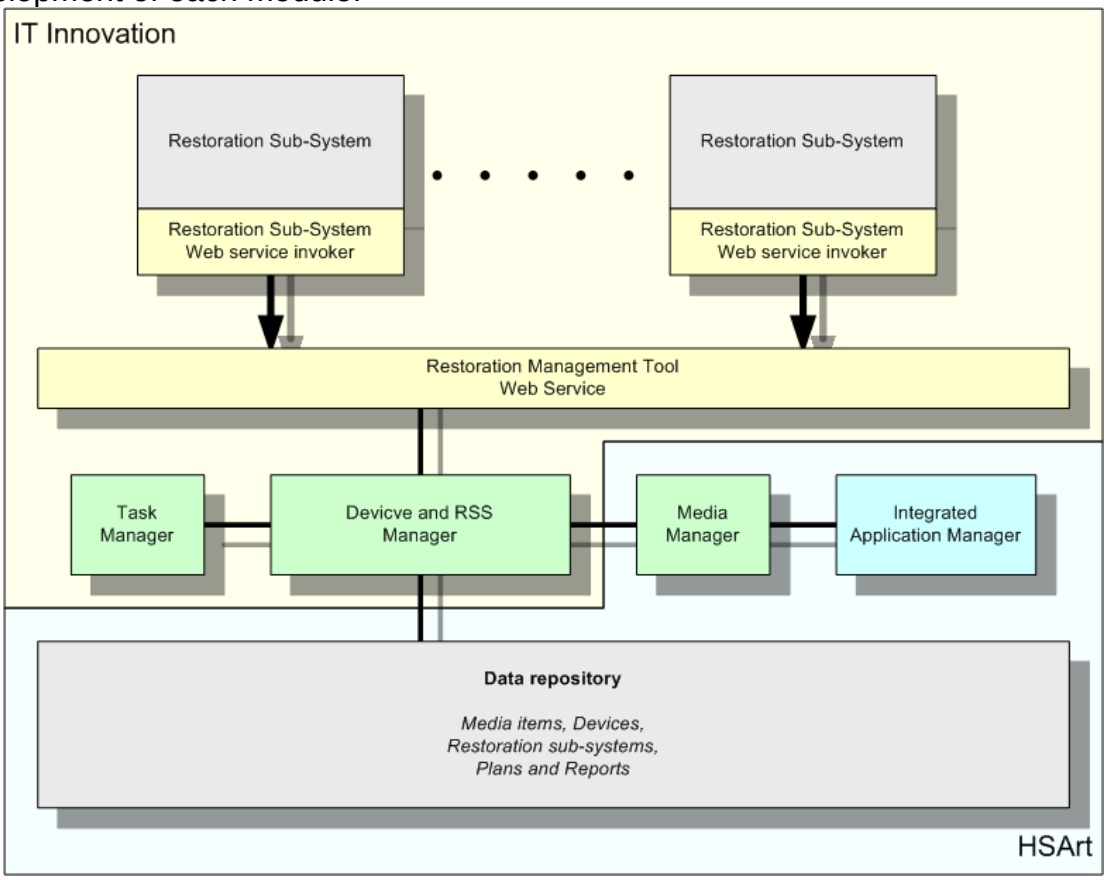


Figure 5.1. Restoration management tool architecture

The communications infrastructure provides a single web service that all restoration sub-systems can invoke. Communication is one way (restoration sub-system to restoration management tool) so the restoration sub-systems must regularly poll to find out what tasks they are expected to do next.
A C++ implementation (Windows platform) of the server side communications library has been created, and a client invocation library (Windows and LINUX platform) shipped to each restoration sub-system provider to invoke it. Example server and client applications are released to show developers how to use the communications code.
We chose GSOAP since it is a lightweight C++ web services implementation that has a very low maintenance and configuration cost. Alternatives such as APACHE AXIS were considered, but rejected since their heavyweight nature would be too costly for realistic exploitation later on in the project. A C++ implementation (as opposed to Java) was chosen to allow easier partner integration with their existing C++ legacy applications. We considered various workflow tools, including soft coded workflow enactors such as FreeFlou, but rejected this technology since the restoration processes are known in advance and often consist of manual operations; fully automated workflow was just not needed when a simpler communication protocol and task queuing system would suffice.

5.2.1. Manager Modules

The restoration management tool managers have the following roles:

- Device and RSS Manager
  Maintains a list of currently installed IO devices and restoration sub-systems. This list will change as time goes on and the restoration apparatus is updated and modified. Each device and sub-system has a set of configuration parameters that are recoded with its unique ID.
- Task manager
  Manages the list of previous and currently active restoration tasks that have been submitted to restoration sub-systems. It provides feedback about status and progress on request, and is used to schedule and unscheduled tasks.
- Media manager
  Manages the individual units within the restoration management tool (MediaItems and associated reports & restoration plans).
- Integrated application manager
  Provides functionality to use integrated applications for media display/edit (e.g.: MoviePlayer), display/edit restoration plans & reports and to provide access to the PrestoSpace factory.

5.2.2. Data Repository

The data repository is a simple file repository, providing persistence and checkpointing functionality. The data repository contains the following data items needed for normal execution of the restoration management tool:

- MediaItems such as D2API CANs and some accompanying information like icons, restoration plans & reports.
- An IO Device list of both active and configured IO Devices
- A RSS list of both active and configured restoration sub-systems
- A task list/queue for each for each restoration sub-systems
- A list of integrated applications (e.g.: MovieManager)

5.2.3. XML Data Structures

Each data item in the data repository is represented using an XML structure that can be serialized and de-serialized to disk. Each restoration sub-system has a task queue, containing both currently active tasks and previous tasks that have been (un)successfully completed. Tasks on the task queue contain configuration information, a restoration sub-system specific restoration plan, used device list and a set of status reports that were generated during the lifetime of the task. A global device list is also maintained so that the restoration management tool operator knows which devices are currently locked for which tasks.
Appendix A contains the XML structures used for reference purposes.

5.3. Software release details

The restoration management tool communications library distribution has a simple structure detailed below. The shipped executables are pre-compiled for Windows 2000 (or better). The distribution contains an ANT build script and will use the Visual C++ compiler (7.1 or later) to build the executables. The restoration management tool will compile using Visual C++. The restoration sub-system example client application will compile with most C++ compilers (Visual C++, GCC etc) under Windows and LINUX.
The restoration management tool communications library release consists of two sub-components, the restoration management tool communications library (and example application) and the restoration sub-system invoker library (and example application). The two sub-components are bundled as a whole software release covered by the terms of the shipped licence file.
The software has been released and uploaded onto the PrestoSpace server for partners to use. Example applications are included as source, along with an ANT build environment configured to compile and link these applications ‘out of the box’.

5.3.1. Distribution structure

rmt-client<version>.zip	Restoration management tool example client application and rmt-invoker library.
rmt-server-<version>.zip	Restoration management tool example server application and communications rmt-server library.

5.3.2. Restoration sub-system distribution (LINUX, Windows)

/licence-prestospace-rmt.doc	Licence file for this software release.
/ipr-registry.doc	Intellectual property rights registry listing the released code and all third party code licensing details.
/build.xml	Simple ANT build script to compile the restoration sub-system example application code.
/bin/runRSSApp.bat	Client ran script. Run with no parameters for some help info.
/doc/RMT-user-manual.doc	Use manual.
/doc/release-notes.txt	Release notes for this version
/lib/…	Third party libs (GSoap, PThead, Xerces) and the RMT invoker library.
/log	Log directory where the example

	application log file will be written.
/tpc-licences/…	Third party licence files required to be shipped with the RMT release.
/src	Source code for the restoration sub-system example client application.

5.3.3. Restoration management tool distribution (Windows)

/licence-prestospace-rmt.doc	Licence file for this software release.
/ipr-registry.doc	Intellectual property rights registry listing the released code and all third party code licensing details.
/build.xml	Simple ANT build script to compile the restoration management tool example application code.
/bin/runRMTServer.bat	Server ran script. Run with no parameters for some help info.
/doc/RMT-user-manual.doc	User manual.
/doc/release-notes.txt	Release notes for this version
/lib/…	Third party libs (GSoap, PThead, Xerces) and the rmt-server library.
/log	Log directory where the example application log file will be written.
/tpc-licences/…	Third party licence files required to be shipped with the RMT release.
/src	Source code for the restoration management tool example application.

5.4. RMT / RSS interaction

The basic polling behaviour of the restoration sub-systems is described in figure 5.2. The restoration sub-system will poll the restoration management tool when it does not have a current task every 1 to 5 seconds, asking the restoration management tool service for the current task queue. Once the task queue is no longer empty the restoration sub-system will start to process the first task on the queue, reporting immediately that the task has been started. When the task is finished the restoration sub-system will report the results back to the restoration management tool service. The reports will appear on the restoration management tool operator's user interface as they become available, keeping the operator informed as to the current restoration situation for the open project.

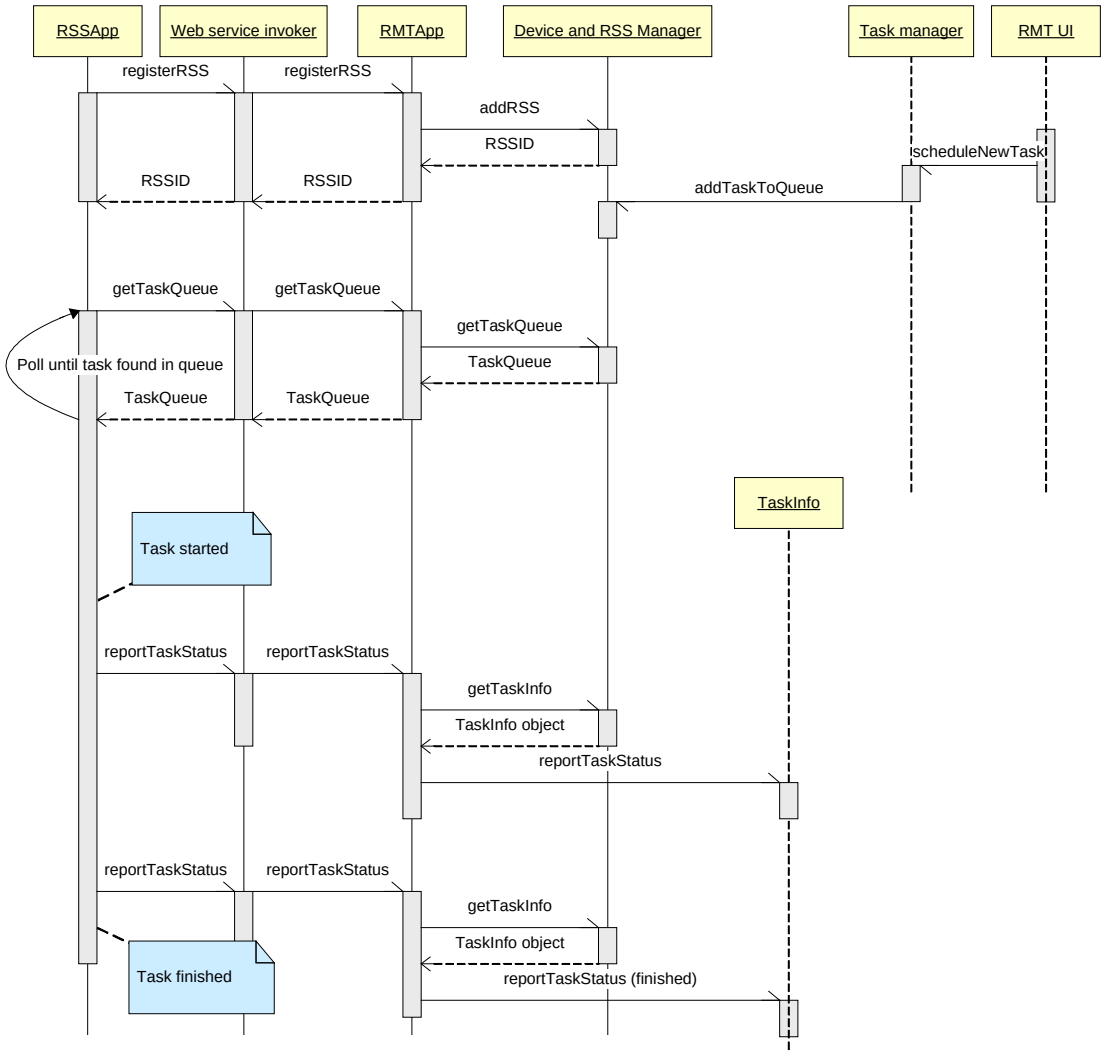


Figure 5.2. Basic restoration sub-system usage

In addition to polling for tasks the restoration sub-system will have to manage the locking of the devices it has been tasked to use. This device usage is described in figure 5.3. When a task is started and requires the device the restoration sub-system should try to lock the device; this might fail so a loop must be performed until the lock has succeeded. Locking a device might fail if another restoration sub-system has locked the device and is performing a task. An example of this is where a video sequence is stored on a disk (the device) and two restoration tasks are needed on the same video footage. The first restoration sub-system to be ready to perform its task will grab the lock first, and the second restoration sub-system will have to wait until this lock has been released before it can perform its own task.

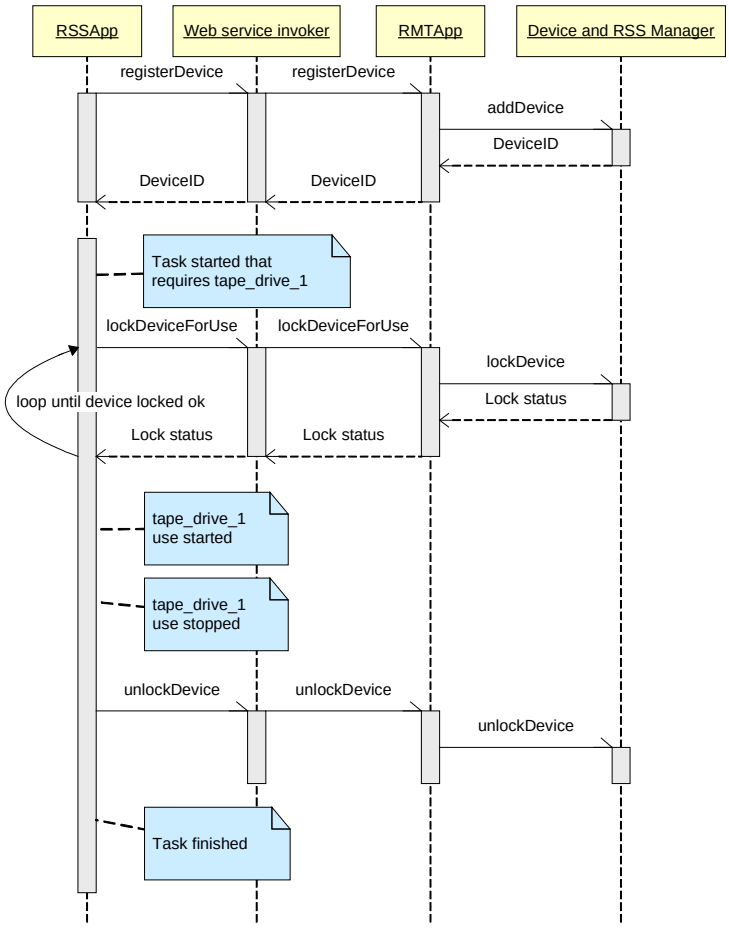


Figure 5.3. Device usage

5.5. Communication performance

Some performance statistics are shown in Figure 5.4, using the below hardware. Please remember that real usage will not execute activities sequentially, so will degrade depending on how many invocations are being processed at the same time. The drop in performance at 128 Mbytes was due to the computer running near the limit of its memory, and needing to start caching etc.
The performance shown should be more than adequate for the restoration management tools role in PrestoSpace.

- Client computer [1.5 GHz, 512Mbyte RAM]
- Server computer [1.8 GHz, 1Gbyte RAM]
- LAN [100 Mbit/s Ethernet]
- 10 iterations run per data size, average computed over these 10 runs

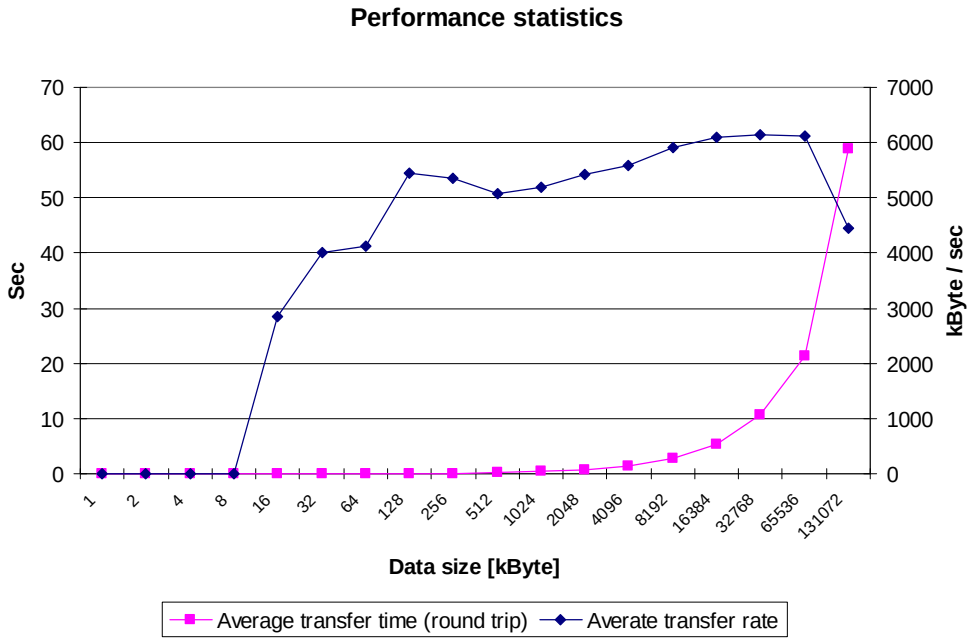


Figure 5.4. RMT performance statistics over a 100Mbit LAN

6. RSS-RMT Integration

6.1. Overview

The restoration management tool design has been chosen to be a flexible, lightweight design with low maintenance costs. C++ was chosen for easy integration with legacy restoration sub-system applications, and XML chosen since there are third party tools available for parsing and it allows us to defer defining the exact XML structure until later in the development process when all the issues were known.
For integration between restoration sub-systems and the restoration management tool we employed a clear strategy, which is outlined below. The development and integration phases were targeted to produce a prototype system that could be demonstrated at the next review.

6.2. Integration strategy

We discussed and developed an initial restoration management tool design based on the requirements of the restoration sub-systems and practical issues involved with distributed systems. The design document was a living document, updated as the project progressed. At this stage we also produced a clear work plan / integration plan so all partners knew when their activities needed to be completed by.
To facilitate early adoption by the restoration sub-system developers, IT Innovation released an early prototype of the communications library. This release contained the basic XML structures needed and some working example applications, both for the restoration management tool server application and the restoration sub-system client applications. Early adoption allowed us to identify issues right from the start, and factor them into the final software releases.
After the release of the first version of the communications library we conducted parallel development threads and parallel integration activities. HS-ART developed the main restoration management tool, including a user interface and data repository support. The restoration sub-systems (BRAVA, AudioCube and DIAMANT) were developed at the same time, providing user interfaces to support the communications library and task management.
Integration was conducted at the pace of each restoration sub-system developer, and will be concluded by bringing the three restoration sub-systems together for final whole system testing (see deliverables D10.1-D10.3).

7. Conclusion

The first version of the RMT as presented in this document fulfils the major requirements that have been formulated in the system architecture documents.
In this sense the available software provides:

- A centralised management and restoration planning tool for the restoration operator;
- Integration of the RSS: DIAMANT, BRAVA and AUDIOCUBE
- A general interface and library for integration of any restoration product into a PrestoSpace compliant RSS.
- An open platform for plug-ins of external tools and software that might vary for different users.

The RSS / RMT integration strategy involved a clear integration plan and schedule, with early release of the communications library. This was followed by parallel developments of the RMT user interface and RSS applications. Integration was conducted at the pace of each restoration sub-system developer, and will be concluded by bringing the three restoration sub-systems together (see deliverables D10.1-D10.3).

In the further course of the PS project further functionality will be added to the RMT. Defect & Quality analysis, quality management as well as communication with other PS components need to be integrated and completed. However the RMT V1.0 is the starting point for user testing and further streamlining.

8. Glossary

Terms	Description
GSOAP	C++ web services software, http://www.cs.fsu.edu/~engelen/soap.html
APACHE AXIS	C++ web services code, http://ws.apache.org/axis/
FreeFlou	Workflow enactor, http://freefluo.sourceforge.net/
Xerces	XML parser, http://xerces.apache.org/
IrfanView	Freeware image viewer, http://www.irfanview.com/
Photoshop	Image painting SW, http://www.adobe.com/photoshop

9. Appendix A – XML data structures

9.1. Repository configuration file

```
<!DOCTYPE RMT_Repository>

<RMT_Repository>
	<Folder Type="0" ID="2005.059.11.36.57.0343" ParentID="ROOT" Label="Repository" />
	<Project Type="1" ID="2005.160.16.22.52.0937" ParentID="2005.059.11.36.57.0343" Label="Films" />
	<MediaItem JobFile="" CanFile="c:/D2/RMT/Repository/CANs/CAN_2005.160.16.23.11.0187.xml" Type="2"
locked="false"		ReportFile="c:/D2/RMT/Repository/RestorationReports/REPORT_2005.160.16.23.11.0187.xml"
	PlanFile="c:/D2/RMT/Repository/RestorationPlans/PLAN_2005.160.16.23.11.0187.xml"
ID="2005.160.16.23.11.0187" ParentID="2005.160.16.22.52.0937"
	IconFile="c:/D2/RMT/Repository/Icons/ICON_2005.160.16.23.11.0187.jpg" Label="Test1" />
	<MediaItem JobFile="" CanFile="c:/D2/RMT/Repository/CANs/CAN_2005.160.17.26.08.0140.xml" Type="2"
locked="false"		ReportFile="c:/D2/RMT/Repository/RestorationReports/REPORT_2005.160.17.26.08.0140.xml"
	PlanFile="c:/D2/RMT/Repository/RestorationPlans/PLAN_2005.160.17.26.08.0140.xml"
ID="2005.160.17.26.08.0140" ParentID="2005.160.16.23.11.0187"
	IconFile="c:/D2/RMT/Repository/Icons/ICON_2005.160.17.26.08.0140.jpg" Label="jpgtest" />
</RMT_Repository>

<RSSList>
	<RMT_RSSInfo>
		<RMT_RSSID>...</RMT_RSSID>
		<RMT_RSSType>...</RMT_RSSType>
		<RMT_ConfigParams>
			... RSS config params ? ...
		</RMT_ConfigParams>
		<RMT_TaskQueue>
			<RMT_TaskInfo> … </RMT_TaskInfo>
			…
		</RMT_TaskQueue>
	</RMT_RSSInfo>
</RSSList>

<DeviceList>
	<RMT_DeviceInfo>
		<RMT_DeviceType>...</RMT_DeviceType>
		<RMT_DeviceID>...</RMT_DeviceID>
		<RMT_ConfigParams>
			... Device config params ? ...
		</RMT_ConfigParams>
	</RMT_DeviceInfo>
</DeviceList>
```

9.2. CAN description file

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<D2 xmlns="http://iis.joanneum.at/diamant/d2.xsd" xmlns:d2="http://iis.joanneum.at/diamant/d2.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://iis.joanneum.at/diamant/d2.xsd D2.xsd
">
<CAN apiInserted="0" id="9681" layerNumber="-1">
  <TemporalExtent>
      <MediaTimePoint>T00:00:01:15F25 </MediaTimePoint>
      <MediaDuration>P0DT0H0M0S4N25F</MediaDuration>
  </TemporalExtent>
  <TrackList>
    <Track apiInserted="205" hasCorrespondingTrackInPreviousCAN="0" isDeliveringOnlyOwnedData="0" label="Import Data"
role="-842150451" xsi:type="VisualTrackType">
     <DRL>
       <DeviceName>ImageFSDevice</DeviceName>
       <ItemsPerStorageUnit>1</ItemsPerStorageUnit>
       <Specific>
         <FSDRLInterface><MountPoint>C:/tmp/</MountPoint>
         <RelativePath>tiftest/</RelativePath><Pattern>L1.####.tif</Pattern>
         </FSDRLInterface>
       </Specific>
     </DRL>
     <OwningInformation>
```

```
      <MediaTime>
            <MediaTimePoint>T00:00:01:15F25</MediaTimePoint>
            <MediaDuration>P0DT0H0M0S4N25F</MediaDuration>
      </MediaTime>
    </OwningInformation>
    <ValidityInformation>
      <MediaTime>
        <MediaTimePoint>T00:00:01:15F25</MediaTimePoint>
        <MediaDuration>P0DT0H0M0S4N25F</MediaDuration>
      </MediaTime>
    </ValidityInformation>
    <HorizontalResolution>0</HorizontalResolution>
    <VerticalResolution>0</VerticalResolution>
    <LowerResolutionTracks/>
   </Track>
  </TrackList>
  <LabelAliases/><TrackDependencies/>
</CAN>
</D2>
```

9.3. Restoration plan description file

```
<!DOCTYPE RMT_RestorationPlan>
<RMT_RestorationPlan>
 <CAN CanFile="Repository/CANs/CAN_2005.154.14.49.55.0687.xml" />
 <DESC_TEXT>
 This sample is heavily damaged and needs to be cleaned up by
 1.) Brava and 2.) DIAMANT.
 Main problems are noise, big blotches and splices.
 Sound quality needs verification!
 </DESC_TEXT>
  <EventList>
 <Event> ID="001" TCFIRST="T00:00:00:1F25" TCLAST="T00:00:25:1F25" DESC="what is there?" ID="ID of the RSS"
Command = "RSS specific command" </Event>
 <Event> ID="002" TCFIRST="T00:00:01:9F25" TCLAST="T00:00:01:9F25" DESC="" ID="Shot" RSSCommand = "" </Event>
 <Event> ID="003" TCFIRST="T00:00:03:15F25" TCLAST="T00:00:03:15F25" DESC="splice on top" ID="DIAMANT_01"
RSSCommand = "interpolate" </Event>
 </EventList>
 <DIAMANT_01>  </DIAMANT_01>
 <BRAVA_01>  </BRAVA_01>
 <AUDIO_CUBE_01>  </AUDIO_CUBE_01>
</RMT_RestorationPlan>
```

9.4. Restoration report description file

```
<!DOCTYPE RMT_RestorationReport>
<RMT_RestorationReport>
 <CAN CanFile="Repository/CANs/CAN_2005.154.14.49.55.0687.xml" />
 <DESC_TEXT>
 This sample has been cleaned up by 1.) Brava and 2.) DIAMANT.
 Main problems werw noise, big blotches and splices.
 Sound quality has been verified!
 </DESC_TEXT>
  <EventList>
 <Event> <TaskID>001</TaskID> TCFIRST="T00:00:00:1F25" TCLAST="T00:00:00:1F25" DESC="what has been done
there?" </Event>
 <Event> <TaskID>002</TaskID> TCFIRST="T00:00:01:9F25" TCLAST="T00:00:10:1F25" DESC="Brava de-noise" </Event>
 <Event> <TaskID>003</TaskID> TCFIRST="T00:00:03:15F25" TCLAST="T00:00:03:15F25" DESC="splice on top" </Event>
 </EventList>
 <DIAMANT_01>  </DIAMANT_01>
 <BRAVA_01>  </BRAVA_01>
 <AUDIO_CUBE_01>  </AUDIO_CUBE_01>
</RMT_RestorationReport>
```

9.5. Task description XML

```
<RMT_TaskInfo>
        <!-- ID's -->
        <RMT_RSSID>...</RMT_RSSID>
        <RMT_TaskID>...</RMT_TaskID>
        <RMT_TaskType>...</RMT_TaskType>

        <!-- completion status -->
        <RMT_Finished> true | false </RMT_Finished>

        <!-- Task specific data needed by the RSS -->
        <RMT_TaskParams>
                <RMT_RequestedStartTime>
                        <RMT_TimeCode>d-m-y:h-m-s</RMT_TimeCode>
                </RMT_RequestedStartTime>
                <RMT_RequestedEndTime>
                        <RMT_TimeCode>d-m-y:h-m-s</RMT_TimeCode>
                </RMT_RequestedEndTime>

                <Source> ... extra RSS tasks specific to the RSS are ok, and will be ignored by RMT code … </Source>

        </RMT_TaskParams>

        <RMT_RestorationPlan>
                ... restoration plan ...
        </RMT_RestorationPlan>

        <RMT_DeviceList>
                <RMT_DeviceID>...</RMT_DeviceID>
                … device list …
        </RMT_DeviceList>

        <!-- History of all task status reports -->
        <RMT_TaskReports>
                <RMT_StatusReport> … see section 6.6 for details … </RMT_StatusReport>
                … status reports …
        </RMT_TaskReports>
</RMT_TaskInfo>
```

9.6. Task status report XML

```
<RMT_StatusReport>
        <RSS_ExpectedCompletionTime>
                <RMT_TimeCode>d-m-y:h-m-s</RMT_TimeCode>
        </RSS_ExpectedCompletionTime>
        <RSS_Status>
                not started | processing | finished | failed
        </RSS_Status>
        <RSS_Error>
                … optional tag to report details on any errors that caused a failed status …
        </RSS_Error>
        <RMT_RestorationPlan>
                ... the restoration plan tag is optional, and is used to update the current restoration plan ...
        </RMT_RestorationPlan>
        <RMT_DeviceList>
                <RMT_DeviceID> ... </RMT_DeviceID>
                ... the device list tag is optional, and is used to update the current device list ...
        </RMT_DeviceList>
        <Details>
                ... RSS developers can add new entries like this details tag (that are ignored by RMT) for their own
operators. …
        </Details>
</RMT_StatusReport>
```