RHOZET

Universal Media Transcoding

Documentation

Carbon API

Harmonic - Rhozet products
www.rhozet.com

Version: 3.16.0

Contents

1

Introduction

This document describes the API for the Rhozet Carbon range of products. Rhozet Carbon products consist of Carbon Coder and the Carbon Farm line of products. Carbon Coder is a desktop transcoding product, while Carbon Farm is a distributed transcoding system, using one or multiple Carbon Servers and one or multiple Carbon Agents for distributed transcoding. This API applies to all Rhozet Carbon products, the differences in products are detailed below:

Carbon Coder

- Desktop and automated transcoding on Windows XP
- Support for all broadcast formats (XMF, LXF, GXF, etc.)
- Support for analog and SDI ingest

Carbon Farm (built by Carbon Server and Carbon Agents)

- Automated transcoding on Windows XP or Windows 2003 Server
- Manager for distributed transcoding processing
- Supports unlimited number of engines with load balancing
- Support for all broadcast formats (XMF, LXF, GXF, etc.)
- Support for analog and SDI ingest

1.1 The Carbon engine

The Rhozet Carbon engine is a component of all the Rhozet Carbon products. After installation of any of the Rhozet Carbon products, the Carbon engine runs as a Microsoft Windows service on one or more machines within your network. It offers transcoding services to users and devices through a set of applications and automation features. In addition, it can be controlled via the API described in this document.

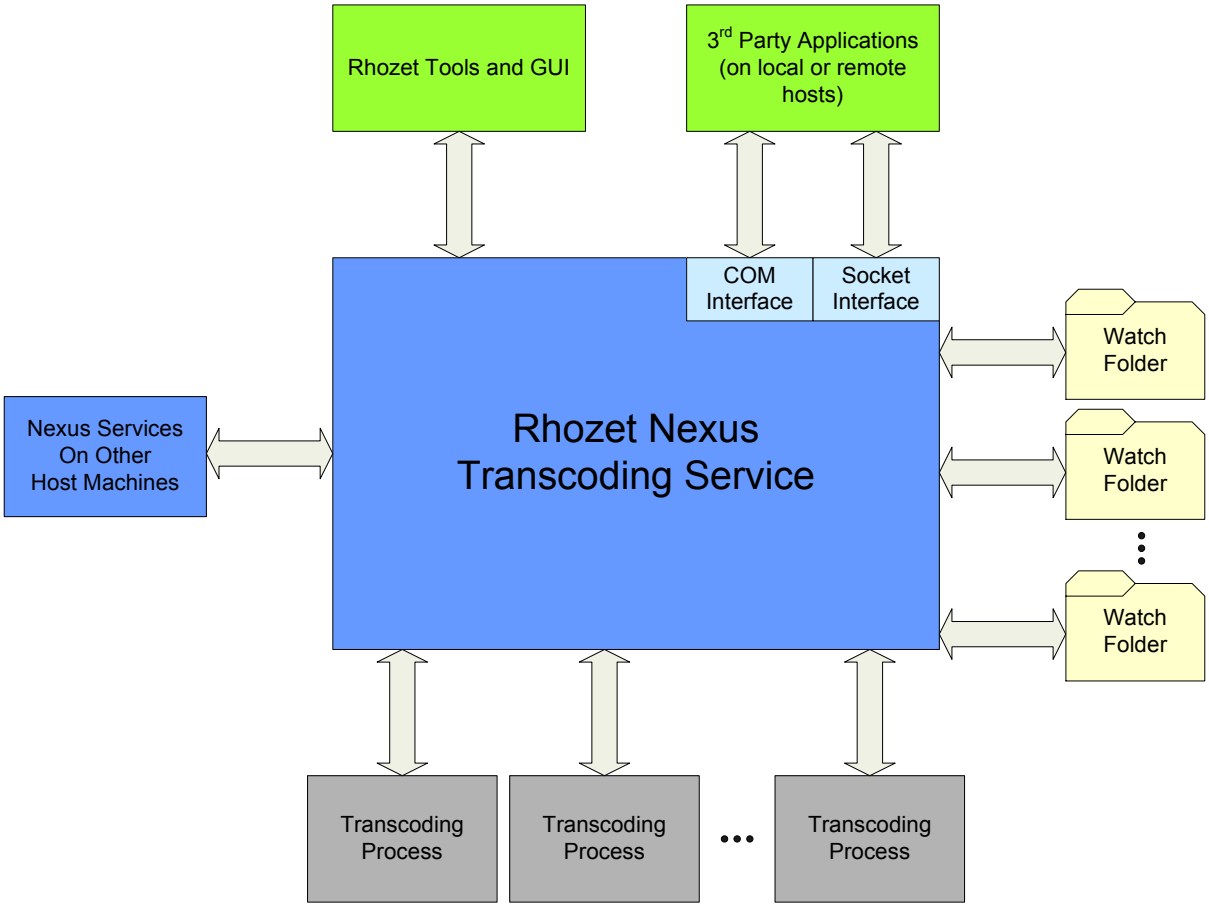
Depending on the type of product, features will differ, but in general Carbon can include GUI and other management tools; the Nexus service which manages dispatching of jobs locally and remotely and can also accept jobs from remote Nexus services on other hosts; and the actual transcoding processes which are called from the command line or from Nexus.

Carbon Coder and Carbon Farm are architecturally similar when installed on a single machine and differ mostly in the applications delivered with the system as well as the inter-machine connectivity.

There are product differentiators between Carbon Coder and Carbon Farm (formats supported, redundancy, load distribution, user interfaces, tools delivered, etc.), but is it very important to note that to a third-party application there is no difference between Rhozet Carbon products from an SDK API perspective. The commands for Carbon Coder are a subset of the commands for Carbon Farm, any development work invested in building a custom application to work with Carbon Coder will automatically scale in performance and redundancy when the custom application is deployed in a Carbon Farm environment.

The following diagram represents a single-host visualization of the components of a Carbon system (some features are version-dependent):

Figure 1.1: API entry points



1.2 SDK Overview

1.2.1 Installation

No SDK installation is required. When a Rhozet Carbon product is installed on a system it includes the installation of the Carbon Engine. Once the Carbon Engine is running, it can accept socket calls and COM commands through Nexus using the API described in this document, and can be controlled with the command-line interface. There is no additional fee for using the Rhozet SDK.

1.2.2 Tools

No specific tools are required. A Rhozet Carbon product installation installs everything necessary to use the API.

1.2.3 API XML Usage

The Rhozet Carbon API uses XML structures for defining and submitting the task to be performed. Example tasks can be:

- submit a transcoding job
- query the status of running jobs
- start/stop running jobs
- modify job parameters (priority, etc.)
- create, enumerate, modify watch folders/drop boxes
- retrieve administrative information about a transcoding system or farm
- manage transcoding agents

The submission of a task for execution can happen in one of the following ways:

- Copying an XML file to a watch folder. The Carbon engine will register the task and either queue it for execution (in the case of transcoding tasks) or simply execute it (for administrative tasks). Note that this is a no feedback system and therefore only suitable for certain tasks.
- Establishing a socket communication to the host on which the Carbon engine is running and submitting the XML data through this socket.
- Calling a command-line program (e.g. from a script or batch file) and passing the name of the command XML file. A response will be sent to stdout and can be piped.
- Using the COM interface to the Carbon engine, the response will be received as a return parameter.

The format of the XML data to be submitted is identical in all cases and will be described in detail along with the task descriptions in the following sections.

1.3 General XML Rules

All XML structures passed must have a payload encapsulated in a <cnpsXML> section. Data outside of this section will be ignored. Note that the entire document must comply with the XML specification as described in http://www.w3.org/TR/REC-xml. In addition, note that only UTF-8 encoded documents are supported.

Note that a valid <cnpsXML> element always specifies the TaskType attribute to indicate the purpose of this XML chunk when submitted to the Carbon Engine. In addition, the version of the Carbon API document used for creating the XML data should be passed to ensure compatibility over different product versions.

Listing 1.1: The <cnpsXML> structure

```
<?xml version="1.0" encoding="utf-8" ?>
<cnpsXML TaskType="xxx" CarbonAPIVer="1.2">
...payload...
</cnpsXML>
```

Although XML supports multiple elements with the same name on the same hierarchy level, the Carbon XML parser does not support this and requires all elements on the same hierarchical level to have unique names. Carbon requires converting multiple identical element names to <elementname_0>, <elementname_1>, etc. This sequence must be strictly followed; no gaps in the sequence numbers are allowed, a sequence must **always** start with _0. Elements with a unique name however can be used without modifications.

Listing 1.2: Incorrect

```
<?xml version="1.0" encoding="utf-8" ?>
<cnpsXML TaskType="xxx" CarbonAPIVer="1.2">
  <Module />
  <Module />
  <Module_5 />
  <Module_7 />
</cnpsXML>
```

Listing 1.3: Correct

```
<?xml version="1.0" encoding="utf-8" ?>
<cnpsXML TaskType="xxx" CarbonAPIVer="1.2">
  <SomeTag />
  <Module_0 />
  <Module_1 />
  <Module_2 />
</cnpsXML>
```

Please note that the Carbon Interface pre-defines which elements are designed for enumeration and which elements are unique, the user cannot deliberately choose this. Elements which are designed for enumeration will be marked as <Elementname_n> in this document, this is an indicator that you can add an arbitrary number of such elements following the numbering scheme in its place.

1.3.1 Example

Listing 1.4: As documented

```
<?xml version="1.0" encoding="utf-8" ?>
<cnpsXML TaskType="xxx" CarbonAPIVer="1.2">
  <SomeElement_n />
</cnpsXML>
```

Listing 1.5: Actual XML structure

```
<?xml version="1.0" encoding="utf-8" ?>
<cnpsXML TaskType="xxx" CarbonAPIVer="1.2">
  <SomeElement_0 />
  <SomeElement_1 />
  <SomeElement_2 />
  ...
</cnpsXML>
```

Part I

Carbon Coder and Carbon Server

2

Carbon Interfaces

2.1 Non-feedback interfaces

Non-feedback interfaces do not provide any reply on job/task submissions therefore they are only suitable for certain tasks. For example, to automatically submit conversion jobs to the Carbon engine the jobs will then be registered and will show up in the job queue list, given that the job was properly described. Issuing informational requests is pointless for example, JobList does not return a list, and while instructional commands (such as reboot agent) can be issued, without feedback there is no control over the actual execution of the task.

2.1.1 XML File Watch Folder Submission

The easiest way to submit jobs and task to the Carbon engine is by generating valid XML files following the specifications in this document and copying them into a folder monitored by the Carbon Engine. After processing, the XML file will be deleted by the engine.

The folder can be specified in the registry:

`HKLM\Software\Rhozet\<your product>\Nexus\JobSubmitFolder`

After a change the system needs to be restarted.

2.2 Feedback interfaces

Feedback interfaces fully comply with the specifications in this document. Submission of data and return structures being received are identical for all feedback interfaces, only the transportation to and from the Carbon engine differs.

2.2.1 Socket Interface

The Socket interface by default listens for connections on port 1120. The port number can also be specified in the registry:

`HKLM\Software\Rhozet\<your product>\Nexus\CarbonAPIPort`

After a change to this registry key the system needs to be restarted to have Carbon use the new value.

The protocol is simple:

1. Connect to the Carbon system using the specified port number.
2. Generate a (multi-byte encoded) string, starting with `"CarbonAPIXML1 "` (note the trailing space character).
3. Add a decimal encoded section with the number of bytes in the XML string following this tag, concluding with a trailing space character. This is both for convenience in dynamic string allocation on the sender/receiver side as well for another checking-level for the completeness/correctness of transmission.
4. Append the XML command string (multi-byte encoded).
5. Send this string to the Carbon system.

Listing 2.1: An example string to be transmitted through a socket is as follows

```
CarbonAPIXML1 68 <?xml version="1.0" encoding="UTF-8" ?><cnpsXML TaskType="
    JobList" />
```

The Carbon system will send a reply string back, following the same convention. The Carbon API will then **disconnect** from the Socket connection (while listening to incoming connections for the next communication attempt). Therefore from the callers perspective, **the socket connection lifetime only lasts for one send/receive command sequence**. In other words, using the socket interface requires a per-command session-based approach.

Listing 2.2: Pseudo-Code for a SendCommand function which also returns the reply

```
SendCommand(const char *command, char *returnbuffer)
{
  opensocket()
  connect()
  Send("CarbonAPIXML1 <string: bytes in XML message> <string: XML message>")
  receive(returnbuffer)
  closesocket()
}
```

2.2.2 Command Line Interface

During installation, Carbon products will install a command line tool, `rzcp.exe`, in

`<Program Files>\Rhozet\<your product>`.

Below is command line help; obtain this by calling rzcp without parameters to get the latest usage and help information.

Listing 2.3: `rzcp.exe` called without parameters

```
Rhozet Carbon Processor
(c) 2005 by Rhozet Corporation

Usage: 1) rzcp.exe [parameters] [presetfile] [projectfile] [source files]
       2) rzcp.exe -queue | -queuewait [-source:<source file>]
                            [-dump:<file>] [-remote:<server>] <XML file>
                            [-queuejob:<file> -queuejob:<file> ...]

Parameters:
     -source:     <source file>
     -target:     <preset GUID>
     -targetdir:  <specify target folder>
                      Note: trailing backslashes are not permitted
     -queuejob:   when queuing, specifies complete job to be queued.
                      Note: multiple projects are possible. Projects must be
                      complete including sources
     -dump:       <1: filename of project file for saving the assembled job>
                  <2: filename of XML file for saving the queue response>
     -remote:     <specify IP or machine name of Carbon Server>

     -raw         only dump the pure response to stdout
                  Note that this can be used for piping an XML response into
                  a well-formed XML file
     -noresponse  do not print queue response XML
     -prf_ft      print processing time
     -prf_nt      print raw processing time (without engine init/de-init

Note: If the -queue/-queueremote parameter is not specified, the conversion
      is executed immediately within rzcp.exe and will be finished
      (or failed) when rzcp.exe terminates. Queue/Queueremote will submit
      it to the Carbon Execution Queue and rzcp will return upon completion
      of the queuing operation.

      For direct execution, multiple sources files are permitted but only
      one project file is accepted. All sources will be stitched
      together to represent one logical source.

      A preset file (.cpf) can be used substituting the project file

      For the -queue command, a complete XML chunk has to be defined
      as specified in the Carbon XML API, other rzcp parameters (besides
      -dump and specifying additional sources) will be ignored when queuing.
```

Even though the tool is mainly designed to simply queue actual jobs for execution, when using the "queue" parameter it will also allow access to the full rich XML command set as described in this document.

2.2.3 COM Interface

The COM interface resides in a DLL, `CarbonCOMDLL.dll`, installed to:

`<Program Files>\Common Files\Rhozet\<your product>\Kernel`

Listing 2.4: For Mircosoft .NET projects, add this DLL as a COM reference to our project and use the 2 commands exposed

```
HRESULT CarbonCommand(    [in] BSTR bstrCommand,
                          [out] BSTR* bstrReturn);

HRESULT CarbonCommandNET( [in] BSTR bstrMachine,
                          [in] BSTR bstrCommand,
                          [out] BSTR* bstrReturn);
```

`CarbonCommand` only talks to the local-machine Carbon kernel. However, `CarbonCommandNET` is able to talk to non-local Carbon kernels as well. `CarbonCommandNET` is only supported for Carbon Farm systems, not for Carbon Coder.

If you intend to use the COM interface in a C++ application, please request sample code from Rhozet.

3

Carbon Tasks

The engine can perform a variety of tasks, depending on the version and product features of your particular Carbon product. The type of task to be executed is encoded in the `TaskType` attribute of the `<cnpsXML>` element, as described in section 5.1 on page 72.

A Carbon task can be either an administrative (most of the administrative tasks are described in chapter 4 starting on page 39) or an actual transcoding task job. In its simplest form, a task always has one command descriptor (the `TaskType` attribute in the `<cnpsXML>` element) and, in the case of queuing a transcoding job, for example, one or more source files and one or more destinations (describing a target file with its encoding parameters).

Listing 3.1: Simple Task (Transcoding Job)

```
<?xml version="1.0" encoding="utf-8" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="JobQueue">
  <Sources>
    <Module_0 Filename="example.avi" />
  </Sources>
  <Destinations>
    <Module_0 PresetGUID="{A7264AEF-FF57-42E0-BBAD-CCF546CD515F}">
  </Destinations>
</cnpsXML>
```

The possible Carbon XML Elements belonging to each task type will be described without particular reference to this type since many elements can be used in various task environments.

3.0.4 Carbon task overview

The following is a rough description of different task types:

JobEvaluate - Conversion Job Evaluation
Evaluate an XML structure, gather information about source files used, etc.

JobQueue - Conversion Job Queuing
Submit a Job for execution or queuing.

JobList - Conversion Job List
This task will return a list of current running or queued jobs.

JobCommand - Conversion Job Administration
The Conversion Job Administration tasks allow a caller to query for information, and to start, stop, delete, etc. a queued, running, or completed conversion jobs.

WatchCreate - Watch Folder Creation
Establish and activate a watch folder.

WatchList - Watch Folder List
This task will return a list of currently active watch folders.

WatchCommand - Watch Folder Administration
The Watch Folder Administration allows the querying for Watch Folder information and to remove/-modify/etc. WatchFolders.

In the following sections, the functions and possible XML elements for each Task Type will be described in detail. The description of the elements themselves can be found in Chapter 5 starting on page 71.

3.1 TaskType "JobEvaluate"

3.1.1 Description

Evaluates a potential transcoding job; the job will be tested (including loading the source files and retrieving source file parameters).

3.1.2 Parameters

Table 3.1: Sub-Elements of <cnpsXML> for TaskType="JobEvaluate"

Element	Type	Description
<Sources>	Element	Describes all source files and logical sources for this transcoding job
<Destinations>	Element	Describes all targets for this job

3.1.3 Return values

Returned is an XML string with a <Reply> root element.

Table 3.2: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description.

A <JobEvaluateResult> element will be attached to a sub-element of <Reply>, containing the modified XML structure. <JobEvaluateResult> contains the traditional <Sources><Module_n> and <Destinations><Module_n> structures, with a <PublicDescription> element added to each <Module_n> element. For more information about <PublicDescription>, see Appendix A JobEvaluate PublicDescription Attributes.

Table 3.3: Attributes of <PublicDescription>

Element	Type	Description
Duration.QWD	Int64, optional	Duration in units of 27MHz clocks (if the duration cannot be retrieved, this attribute is omitted)
RZMETA_VideoBitrate_kbps	Integer, optional	Bitrate of the video stream

Table 3.4: Attributes of <PublicDescription><Video>

Element	Type	Description
Size_X.DWD	Integer	Horizontal # of pixels
Size_Y.DWD	Integer	# of lines
Codec	String	Used Video Codec (user readable, non-standardized)
Interlacing	String	Frame Mode, one of: "Bottom First", "Top First", "Progressive"s
Video_Aspect_X.DWD	Integer	X component of video aspect ratio
Video_Aspect_Y.DWD	Integer	Y component of video aspect ratio
FrameDuration.QWD	Int64, optional	Duration of one Frame in 27MHz clocks (1/framerate). If the source does not have constant frame duration (Windows Media™, QuickTime®), this is omitted.

Table 3.5: Attributes of <PublicDescription><Audio>

Element	Type	Description
SampleRate.DWD	Integer	Samplerate in Hz
SampleSize.DWD	Integer	Bits per sample
Channels.DWD	Integer	Nr of channels
Codec	String	Used Audio Codec (user readable, non-standarized)

3.1.4 Example

Listing 3.2: An example for a Job Evaluation (most attributes and data are omitted only the element structures are shown)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="JobEvaluate" JobName="ThisJob">
  <Sources>
    <Module_n>
    </Module_n>
  </Sources>
  <Destinations>
    <Module_n DestinationName="my destination">
    </Module_n>
  </Destinations>
</cnpsXML>
```

Listing 3.3: An example for a reply from JobEvaluate

```
<?xml version="1.0" encoding="UTF-8" ?>
<Reply Success="TRUE">
  <JobEvaluateResult>
    <Sources>
      <Module_0>
        <ModuleData>
          //   engine internal data, used for advanced analysis
        </ModuleData>
        <PublicDescription Duration.QWD="2414700000">
          <Video Size_X.DWD="160"
                 Size_Y.DWD="120"
                 Codec="Cinepak Codec by Radius"
                 Interlacing="Progressive"
                 Video_Aspect_X.DWD="4"
                 Video_Aspect_Y.DWD="3" />
          <Audio SampleRate.DWD="11025"
                 SampleSize.DWD="16"
                 Channels.DWD="1"
                 Codec="16-bit Big Endian" />
        </PublicDescription>
      </Module_0>
    </Sources>
    <Destinations />
  </JobEvaluateResult>
</Reply>
```

3.2 TaskType "JobQueue"

Submits a transcoding job for queuing/direct execution. The job will be tested for validity (correctness of parameters - and XML structure) at the time of submission. Note however that verification of the existence of sources, target presets, etc. is not performed the job may still fail when it is actually started.

Typically, a job will be queued into the job execution queue and jobs will be executed according to their priority and queuing time. If you need a job executed immediately, please set the Priority number (specified in the <ProjectSetting>) to 255.

3.2.1 Parameters

Table 3.6: Attributes and Sub-Elements of <cnpsXML> for TaskType="JobQueue"

Element	Type	Description
JobName	String	Name of the job
Description	String	Description of the job
User	String	Name of the User
<Sources>	Element	Describes all source files and logical sources for this transcoding job
<Destinations>	Element	Describes all targets for this job. This can be used mutually exclusive with <WatchDestination>
<ProjectSettings>	Element	Describes conversion parameters
<EmailSettings>	Element	Specifies shared email account information
<FTPSettings>	Element	Specifies shared FTP account information

3.2.2 Return values

Returned is an XML string with a <Reply> root element.

Table 3.7: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description
GUID	String	Job GUID for referencing this job in subsequent calls

3.2.3 Example

In addition to the example in the introduction of this chapter, audio and video filters can be added to either sources or targets, general processing options and settings can be specified and pre- and post processing options such as file copy/FTP upload/email notifications/etc. can be added.

Listing 3.4: An example for a full transcoding task description (attributes and data are omitted only the element structures are shown)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="JobQueue" JobName="ThisJob">
  <Notify />
  <Sources>
    <Module_n>
      <Filter_n>
        <Module_n />
      </Filter_n>
      <Notify />
      <PostconversionTasks>
        <FTPUpload_n>
          <Notify />
        </FTPUpload_n>
        <FileCopy_n>
          <Notify />
        </FileCopy_n>
      </PostconversionTasks>
      <ConversionErrorTasks>
        <FTPUpload_n>
          <Notify />
        </FTPUpload_n>
        <FileCopy_n>
          <Notify />
        </FileCopy_n>
      </ConversionErrorTasks>
    </Module_n>
  </Sources>
  <Destinations>
    <Module_n DestinationName="my destination">
      <Filter_n>
        <Module_n />
      </Filter_n>
      <Notify />
      <PostconversionTasks>
        <FTPUpload_n>
          <Notify />
        </FTPUpload_n>
        <FileCopy_n>
          <Notify />
        </FileCopy_n>
      </PostconversionTasks>
    </Module_n>
  </Destinations>
  <ProjectSettings />
  <EmailSettings />
  <FTPSettings />
</cnpsXML>
```

Listing 3.5: An example for a reply from JobQueue

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE" GUID="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}"/>
```

3.3 TaskType "JobList"

Returns a list of Jobs, paired in Name/GUID pair. The application can create and display a list based on this call and then query for detailed job information using the GUIDs as identifiers.

3.3.1 Parameters

Table 3.8: Sub-Elements of <cnpsXML> for TaskType="JobList"

Element	Type	Description
N/A	N/A	This command has no parameters

3.3.2 Return values

Returned is an XML string with a <Reply> root element and a <JobList> sub element.

Table 3.9: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description.
NrOfJobs.DWD	Integer	Number of <job_n> elements in <JobList>

Table 3.10: Attributes of <Job_n>

Element	Type	Description
GUID	String	The GUID of the Job
Name	String	Name of the Job

3.3.3 Example

Listing 3.6: An example for a JobList call

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="JobList" />
```

Listing 3.7: An example for a reply from JobList

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE" NrOfJobs.DWD="3">
  <JobList>
    <Job_0 Name="Job 1" GUID="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}" />
    <Job_1 Name="Job 2" GUID="{32A160F3-49E8-5358-6073-7DCAE4B4ED75}" />
    <Job_2 Name="Job 3" GUID="{324160F3-59E8-6358-7799-435AE4B4ED34}" />
  </JobList>
</Reply>
```

3.4 TaskType "JobStatusList"

Returns a list of Jobs, paired in Name/GUID pair. The application can create and display a list based on this call and then query for detailed job information using the GUIDs as identifiers.

3.4.1 Parameters

Table 3.11: Sub-Elements of <cnpsXML> for TaskType="JobStatusList"

Element	Type	Description
<Filter>	Element	Includes parameters to limit the content of the list
<ListSettings>	Element	Includes parameters to sort and prepare the output of the list

Table 3.12: Sub-Elements of <cnpsXML><Filter> for TaskType="JobStatusList"

Element	Type	Description
<Criteria_n>	Element, optional	Representation of a search criteria. If multiple is used, the AND operator is used to merge the criteria, limiting the output

Table 3.13: Attributes of <cnpsXML><Filter><Criteria_n> for TaskType="JobStatusList"

Element	Type	Description
Parameter	String	A definition of which parameter in a job description to filter on (see in "Remarks" for more info)
Operator	String, optional	One of: EQUAL (default), LIKE, LESSTHAN, GREATERTHAN
Value	String	The value that the Parameter must conform to, for example: If Parameter = "Status", possible values can be: Preparing, Queued, Starting, Started, Stopping, Stopped, Pausing, Paused, Resuming, Completed, Error and Invalid. Similarly if Parameter = "State", possible values can be: NEX_JOB_PREPARING, NEX_JOB_QUEUE, NEX_JOB_STARTING, NEX_JOB_COMPLETED, NEX_JOB_ERROR....Etc.

Table 3.14: Attributes of <cnpsXML><ListSettings> for TaskType="JobStatusList"

Element	Type	Description
OrderBy	String, optional	The value can be any value that can go as the Parameter in a <Criteria_n> element (described in table 3.13)
OrderByDirection	String, optional	One of: ASCENDING, DESCENDING (default)
LimitStart.DWD	Integer, optional	The first element to show (Default is 0, for the first one)
LimitShow.DWD	Integer, optional	The amount of elements to show (-1 or omitted means all elements)

3.4.2 Return values

Returned is an XML string with a <Reply> root element and a <JobStatusList> sub element.

Table 3.15: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description.

Table 3.16: Sub-Elements and Attributes of <Job_n>

Element	Type	Description
Name	String	Name of the job
GUID	String	GUID of the job
State	String	Current state of the job
Status	String	Current status of the job
Progress.DWD	Integer	Current progress of the job
Description	String	Description of the job
User	String	Username of the user that created the job
SourceDescription	String	Description of the source
AgentIP	String	IP address of the agent or unassigned
Guid	String	Guid of the job on agent
Priority.DWD	Integer	Priority of the job
DeleteProcessedSource.DWD	Integer	0/1 whether the source file will be deleted on completion
CheckInTime	String	Time the job was submitted (MM/DD/YY hh:mm:ss)
CheckInTime_CNLT	String	Time the job was submitted (YYYY-MM-DD hh:mm:ss)
CheckInTime_SCM	String	Time(GMT) the job was submitted (YYYY-MM-DD hh:mm:ss)
StartTime	String, optional	Time the job was started (MM/DD/YY hh:mm:ss)
StartTime_CNLT	String, optional	Time the job was started (YYYY-MM-DD hh:mm:ss)
StartTime_SCM	String, optional	Time(GMT) the job was started (YYYY-MM-DD hh:mm:ss)
CompletedTime	String, optional	Time the job was completed (MM/DD/YY hh:mm:ss)
CompletedTime_CNLT	String, optional	Time the job was completed (YYYY-MM-DD hh:mm:ss)
CompletedTime_SCM	String, optional	Time(GMT) the job was completed (YYYY-MM-DD hh:mm:ss)
<Failures>	Element	Contains any warnings or errors the job might have encountered under sub elements <Warnings> and <Errors>

(Note: Some attributes may not be mentioned here because they are not a committed API feature and may disappear over time)

3.4.3 Remarks

3.4.3.1 The Parameter attribute in <Criteria_n> elements

The value of the `Parameter` attribute must be a specification for a single attribute inside the job XML.

3.4.4 Example

Listing 3.8: An example for a JobStatusList call

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="JobStatusList">
  <Filter>
    <Criteria_0 Parameter="CheckInTime_CNLT"
                Operator="GREATERTHAN"
                Value="2008-05-09 14:04:00" />
    <Criteria_1 Parameter="JobName"
                Operator="LIKE"
                Value="Test" />
    <Criteria_2 Parameter="Status"
                Operator="EQUAL"
                Value="COMPLETED" />
  </Filter>
  <ListSettings />
</cnpsXML>
```

Listing 3.9: An example for a reply from JobStatusList

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE">
  <JobStatusList NrOfJobs.DWD="1" TotalNrOfJobs.DWD="11">
    <Job_0 Name="Test Job 1"
           GUID="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}"
           State="NEX_JOB_COMPLETED"
           Status="COMPLETED"
           Progress.DWD="100"
           Description="Test Job"
           User="WatchFolder"
           SourceDescription="test.avi"
           AgentIP="localhost"
           Guid="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}"
           Priority.DWD="5"
           DeleteProcessedSource.DWD="0"
           CheckInTime="05/09/08 14:04:02"
           CheckInTime_CNLT="2008-05-09 14:04:02"
           CheckInTime_SCM="2008-05-09 21:04:02"
           StartTime="05/09/08 14:10:53"
           StartTime_CNLT="2008-05-09 14:10:53"
           StartTime_SCM="2008-05-09 21:10:53"
           CompletedTime="05/09/08 14:11:15"
           CompletedTime_CNLT="2008-05-09 14:11:15"
           CompletedTime_SCM="2008-05-09 21:11:15">
      <Failures>
         <Warnings/>
         <Errors/>
      </Failures>
    </Job_0>
```

```
  </JobList>
</Reply>
```

3.5 TaskType "JobCommand"

JobCommand is the central entry point to query and modify jobs.

3.5.1 Parameters

Table 3.17: Sub-Elements of <cnpsXML> for TaskType="JobCommand"

Element	Type	Description
<JobCommmand>	Element	The element storing the GUID and the command to be applied to the job with the passed GUID

Table 3.18: Attributes of <JobCommand>

Element	Type	Description
GUID	String	The GUID of the job
Command	String	Command String. Described below in table "Values of Command attribute"
Priority.DWD	Integer	Valid for Command="SetPriority". Specifies the priority (0 through 9, or 255 for immediate execution)

Table 3.19: Values of Command attribute

Element	Type	Description
QueryInfo	String	Returns an informational structure for this job in the element <JobInfo>
Query	String	Returns the entire job structure in <Job>. The job can be stored and used for re-submission
Stop	String	Stops the job if it is currently being executed, no effect otherwise. The job will not be destroyed, it can be re-queued.
Pause	String	Pauses the job. Can be used, for example, to execute high priority jobs without discarding the so-far encoded result of lower priority jobs. No effect if a job is not running.
Resume	String	Resumes a paused job. No effect if a job is not running.
SetPriority	String	Changes the job priority.
Requeue	String	Re-submits a job with the status "Failed", "Stopped" or "Completed" to the execution queue.
Remove	String	Removes the job from the queue, regardless of status

3.5.2 Return values

Returned is an XML string with a <Reply> root element and a <JobInfo> sub element.

Table 3.20: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description.

Table 3.21: Attributes of <JobInfo>

Element	Type	Description
Name	String	Name of the job
GUID	String	GUID of the job
State	String	Current state of the job
Status	String	Current status of the job
Progress.DWD	Integer	Current progress of the job
Description	String	Description of the job
User	String	Username of the user that created the job
SourceDescription	String	Description of the source
AgentIP	String	IP address of the agent or unassigned
Guid	String	Guid of the job on agent
Priority.DWD	Integer	Priority of the job
DeleteProcessedSource.DWD	Integer	0/1 whether the source file will be deleted on completion
CheckInTime	String	Time the job was submitted (MM/DD/YY hh:mm:ss)
CheckInTime_CNLT	String	Time the job was submitted (YYYY-MM-DD hh:mm:ss)
CheckInTime_SCM	String	Time(GMT) the job was submitted (YYYY-MM-DD hh:mm:ss)
StartTime	String, optional	Time the job was started (MM/DD/YY hh:mm:ss)
StartTime_CNLT	String, optional	Time the job was started (YYYY-MM-DD hh:mm:ss)
StartTime_SCM	String, optional	Time(GMT) the job was started (YYYY-MM-DD hh:mm:ss)
CompletedTime	String, optional	Time the job was completed (MM/DD/YY hh:mm:ss)
CompletedTime_CNLT	String, optional	Time the job was completed (YYYY-MM-DD hh:mm:ss)
CompletedTime_SCM	String, optional	Time(GMT) the job was completed (YYYY-MM-DD hh:mm:ss)
<Failures>	Element	Contains any warnings or errors the job might have encountered under sub elements <Warnings> and <Errors>

3.5.3 Remarks

The <Job> Element returned by Command="Query" is identical to the <cnpsXML> element from a Job as submitted by TaskType="JobQueue".

3.5.4 Example

Listing 3.10: An example for a JobCommand call

```
<?xml version="1.0" encoding="UTF-8"?>
<cnpsXML CarbonAPIVer="1.2" TaskType="JobCommand">
  <JobCommand Command="QueryInfo" GUID="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}"
      />
</cnpsXML>
```

Listing 3.11: An example for a reply JobCommand

```
<?xml version="1.0" encoding="UTF-8"?>
        <Reply Success="TRUE">
        <JobInfo Name="Job 1"
                 GUID="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}"
                 State="NEX_JOB_STARTED"
                 Status="STARTED"
                 Progress.DWD="32"
                 Description="Example Job"
                 User="WatchFolder"
                 SourceDescription="C:\_test.mp4"
                 AgentIP="192.168.62.133"
                 Guid="{B33E52F4-9185-4C31-BD93-E28737746C33}"
                 Priority.DWD="5"
                 DeleteProcessedSource.DWD="1"
                 CheckInTime="05/15/08 12:23:29"
                 CheckInTime_CNLT="2008-05-15 12:23:29"
                 CheckInTime_SCM="2008-05-15 19:23:29"
                 StartTime="05/15/08 12:23:41"
                 StartTime_CNLT="2008-05-15 12:23:41"
                 StartTime_SCM="2008-05-15 19:23:41" />
   </Reply>
```

3.6 TaskType "WatchCreate"

Establishes a Watch Folder on the Carbon System. The submission of a Watch Folder is identical to the submission of a Job with the exception that the <Sources> element is not valid here, a <WatchSource> element must exist instead. The Watch folder will be established immediately and will remain active until deleted. If watches need to be "deactivated", then query for the Watch, store the info, delete the watch and re-submit it when needed.

The Watch information will be tested for validity (correctness of parameters - and XML structure) at the time of submission. Note however that no checks will be performed regarding existence of target presets, etc. the job may still fail when it is actually started by an incoming source.

3.6.1 Parameters

Table 3.22: Sub-Elements of <cnpsXML> for TaskType="WatchCreate"

Element	Type	Description
<WatchSource>	Element	Describes all parameters associated with the Watch (Section 5.3 on page 82)
<Destinations>	Element	Describes all targets for this Watch
<ProjectSettings>	Element	Describes conversion parameters
<EmailSettings>	Element	Specify shared email account information
<FTPSettings>	Element	Specify shared FTP account information

3.6.2 Return values

Returned is an XML string with a <Reply> root element.

Table 3.23: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description
GUID	String	Watch GUID for referencing this Watch Folder in subsequent calls

3.6.3 Example

Listing 3.12: Query

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="WatchCreate">
  <Notify />
  <WatchSource WatchPath="G:\example"
               WatchSubFolder.DWD="0"
               AppendSubFolder.DWD="0"
               OverWrite.DWD="0"
               Priority.DWD="5"
```

```
              WatchDeleteProcessedSource.DWD="0" >
  </WatchSource>
  <Destinations>
    <Module_n DestinationName="my destination">
      <Filter_n>
        <Module_n />
      </Filter_n>
      <Notify />
      <PostconversionTasks>
        <FTPUpload_n>
          <Notify />
        </FTPUpload_n>
        <FileCopy_n>
          <Notify />
        </FileCopy_n>
      </PostconversionTasks>
    </Module_n>
  </Destinations>
  <ProjectSettings />
  <EmailSettings />
  <FTPSettings />
</cnpsXML>
```

Listing 3.13: Reply

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE" GUID="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}"/>
```

3.7 TaskType "WatchList"

Returns a list of Watch Folders, paired in Watch Path/GUID pairs. The application can create and display a list based on this call and then query for detailed watch information using the GUIDs as identifiers.

3.7.1 Parameters

Table 3.24: Sub-Elements of <cnpsXML> for TaskType="WatchList"

Element	Type	Description
N/A	N/A	This command has no parameters

3.7.2 Return values

Returned is an XML string with a <Reply> root element and a <WatchList> sub element.

Table 3.25: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description

Table 3.26: Attributes of <WatchList>

Element	Type	Description
NrOfWatches.DWD	Integer	Number of <Watch_n> elements in <WatchList>

Table 3.27: Attributes of <Watch_n>

Element	Type	Description
GUID	String	The GUID of the job
WatchPath	String	Path being watched
Name	String	Name passed at creation time

3.7.3 Example

Listing 3.14: An example for a WatchList call

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="WatchList" />
```

Listing 3.15: An example for a reply from WatchList

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Reply Success="TRUE">
  <WatchList NrOfWatches.DWD="3">
    <Watch_0 Name="Watch1"
             WatchPath="\\server\share1"
             GUID="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}" />
    <Watch_1 Name="Watch2"
             WatchPath="\\server\share2"
             GUID="{32A160F3-49E8-5358-6073-7DCAE4B4ED75}" />
    <Watch_2 Name="Watch3"
             WatchPath="\\server\share3"
             GUID="{324160F3-59E8-6358-7799-435AE4B4ED34}" />
  </WatchList>
</Reply>
```

3.8 TaskType "WatchCommand"

WatchCommand is the central entry point to query and modify watches.

3.8.1 Parameters

Table 3.28: Sub-Elements of <cnpsXML> for TaskType="WatchCommand"

Element	Type	Description
<WatchCommand>	Element	The element storing the GUID and the command to be applied to the watch with the passed GUID

Table 3.29: Attributes of <WatchCommand>

Element	Type	Description
GUID	String	The GUID of the job
Command	String	Command String. Described below in table "Values of Command attribute"
Priority.DWD	Integer	Valid for Command="SetPriority". Specifies the priority (0 through 9, or 255 for immediate execution)

Table 3.30: Values of Command attribute

Element	Type	Description
QueryInfo	String	Returns an informational structure for this watch in the element <WatchInfo>
Query	String	Returns the entire Watch structure in <Watch>. Can be stored and used for re-submission
SetPriority	String	Sets the Priority of the jobs which will be created by this Watch
Remove	String	Removes/Deletes the Watch
Disable	String	Disables the watch, but keeps it in the list
Enable	String	Enables a previously disabled watch
SubmitSources	String	Submits source files to this watch without physically moving them. A sub-element <Sources> as described in section 5.2 on page 73 must be added. Note that source filters will be ignored when submitting sources this way

3.8.2 Return values

Returned is an XML string with a <Reply> root element and, depending on the Watch Command, a <Watch> sub-element.

The <Watch> Element returned by Command="Query" is identical to the <cnpsXML> element from a Watch as submitted by TaskType="WatchCreate".

Table 3.31: Attributes of <Reply>:

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description

3.8.3 Example

Listing 3.16: An example for a WatchCommand call

```
<?xml version="1.0" encoding="UTF-8"?>
<cnpsXML CarbonAPIVer="1.2" TaskType="WatchCommand">
  <WatchCommand Command="SetPriority"
                GUID="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}"
                Priority.DWD="3" />
</cnpsXML>
```

Listing 3.17: An example for a reply from WatchCommand

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE" />
```

4

Carbon Advanced Tasks

In this chapter, all the advanced tasks, primarily focused on administrative purposes, are described in detail. The general syntax and common elements are the same as for the tasks described in chapter 3.

4.0.4 Carbon advanced task overview

The following is a rough description of different task types:

ProfileSave - Create a new profile
This creates (or overwrites) a Profile

ProfileList - Obtain Profile List
This returns a list of all profiles

ProfileCommand - Profile administration
Allows to get profile details and delete profiles

TagList - Get a list of tags
Function to return a list of all the machine tags in a Carbon environment

DirCreate - Create a new folder
Function to create a new local or remote folder

DirList - Directory listing function
List the files and/or folders for a local or remote folder

ServerConnectionTest - Test a remote server connection
Function to test the connection and user rights for a remote server connection

Version - Get the current version
Function to get the current version of a Carbon machine

CarbonRole - Get the role of Carbon
Function to get the role of Carbon to determine if it's a Coder, Server or an Agent

FormPreset* - Manage form presets
Functions to manage form presets for easy remote data fill-out

NodeList - Get a list of all the nodes/agents in a farm
This returns a list of all the nodes in a Carbon Server Farm

NodeCommand - Node Administration
Allows to query and modify Node settings

4.1 TaskType "ProfileSave"

Saves a Profile for Carbon. The creation of a Profile Folder is similar to the submission of a Job with the exception that the <Sources> element is not valid here, instead a profile attributes section must be attached.

In addition to target/destination profiles, this command is also used to manage audio- and video-filter profiles and configuration settings.

4.1.1 Parameters

Table 4.1: Sub-Elements of <cnpsXML> for TaskType="ProfileSave"

Element	Type	Description
<ProfileAttributes>	Element	Describes all parameters associated with the Profile
<Destinations>	Element, opt.	Describes the target and filters for this Profile (note: only one target is permitted)
<Filter_Audio>	Element, opt.	Describes the audio filter for this Profile (note: only one filter is permitted)
<Filter_Video>	Element, opt.	Describes the video filter for this Profile (note: only one filter is permitted)
<Setting>	Element, opt.	Describes the settings for this Profile (note: only one settings group is permitted)

Table 4.2: Attributes of <ProfileAttributes>

Element	Type	Description
ProfileType	String	One of: Destination, Audio_Filter, Video_Filter, Settings
Name	String	Readable name of the profile
Category	String	Category of the Profile
Description	String	Description of the Profile
GUID	GUID (String), optional	When set, uses this GUID. If another profile with this GUID exists, it will be replaced. Can be used to update profiles

4.1.2 Return values

Returned is an XML string with a <Reply> root element.

Table 4.3: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String, optional	If (Success==FALSE) then "Error" contains a human readable error description
GUID	GUID (String)	Profile GUID for referencing this Profile in subsequent calls

4.1.3 Remarks

Please study the creation of Destinations in section 5.4 for details about how to set up the destination parameters.

4.1.4 Example

Listing 4.1: An example for a ProfileSave call (actual profile data are omitted)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="ProfileSave">
  <ProfileAttributes Name="My New Profile"
                     Category="My Profiles"
                     Description="Profile I made today"
                     ProfileType="Destination" />
  <Destinations>
    <Module_0 DestinationName="my destination">
      ...
    </Module_0>
  </Destinations>
</cnpsXML>
```

Listing 4.2: An example for a reply from ProfileSave

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE" GUID="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}" />
```

4.2 TaskType "ProfileList"

Returns a list of Profile attributes. For further information about particular profiles, use `ProfileCommand`, using the obtained GUIDs as identifiers.

4.2.1 Parameters

Table 4.4: Sub-Elements of <cnpsXML> for TaskType="ProfileList"

Element	Type	Description
<ProfileAttributes>	Element	Defining what profiles to obtain

Table 4.5: Attributes of <ProfileAttributes>

Element	Type	Description
ProfileType	String	One of: Destination, Filter_Audio, Filter_Video, Setting, Connection

4.2.2 Return values

Returned is an XML string with a <Reply> root element and a <ProfileList> sub-element.

Table 4.6: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String, optional	If (Success==FALSE) then "Error" contains a human readable error description

Table 4.7: Elements and attributes of <ProfileList>

Element	Type	Description
<Profile_n>	Elements	One or more elements, described below
NrOfProfiles.DWD	Integer	Number of <Profile_n> elements in <ProfileList>

Table 4.8: Elements and attributes of <Profile_n>

Element	Type	Description
GUID	GUID (String)	The GUID of the profile
Name	String	Readable name of the profile
Category	String	Category of the Profile
Description	String	Description of the Profile
ProfileFlags.DWD	Integer	Overall scope and protection for the preset (see section 4.2.3.1 for more info)

4.2.3 Remarks

4.2.3.1 ProfileFlags.DWD

The `ProfileFlags.DWD` attribute holds information about the type of preset (`SYSTEM` or `USER`) and whether the preset is read only or can be edited.

Table 4.9 shows what the different bits for this value covers.

Table 4.9: Values for `ProfileFlags.DWD`

Bit (in HEX)	Value	Description
0x00000001	SET	The preset can be **Queued**
0x00000001	NOT SET	The preset can **not** be **Queued**
0x00000002	SET	The preset is a **SYSTEM** preset
0x00000002	NOT SET	The preset is a **USER** preset
0x00000004	SET	The preset is a **readonly** preset
0x00000004	NOT SET	The preset is an **editable** preset
0x00000008	SET	The preset can use **macro gridding**
0x00000008	NOT SET	The preset can **not** use **macro gridding**
0x00000010	SET	The preset is a **complex** preset having multiple targets defined
0x00000010	NOT SET	The preset is an **simple** preset having only one target defined
0x00000020	SET	Only applicable for `ProfileType="Connection"`. This connection can be used as a delivery option
0x00000020	NOT SET	Only applicable for `ProfileType="Connection"`. This connection can **not** be used as a delivery option
0x00000040	SET	Only applicable for `ProfileType="Connection"`. This connection can be used as an acquisition option for a watch
0x00000040	NOT SET	Only applicable for `ProfileType="Connection"`. This connection can **not** be used as an acquisition option for a watch

4.2.4 Example

Listing 4.3: An example for a ProfileList call

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="ProfileList">
  <ProfileAttributes ProfileType="Destination" />
</cnpsXML>
```

Listing 4.4: An example for a reply from ProfileList

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE">
  <ProfileList NrOfProfiles.DWD= 2 >
    <Profile_0 Name="MyProfile"
               Description="Profile Description 1"
               Category="My Profiles"
               ProfileFlags.DWD="1"
               GUID="{62A160F3-79E8-4358-9073-EDCAE4B4ED75}" />
    <Profile_1 Name="Old Profile"
               Description="Profile Description 2"
               Category="System"
               ProfileFlags.DWD="1"
               GUID="{42A160F3-19E8-2358-5073-ADCAE4B4ED75}" />
  </ProfileList>
</Reply>
```

4.3 TaskType "ProfileCommand"

ProfileCommand is the central entry point to query and remove profiles.

4.3.1 Parameters

Table 4.10: Sub-Elements of <cnpsXML> for TaskType="ProfileCommand"

Element	Type	Description
<ProfileCommand>	Element	The element storing the GUID and the command to be applied to the profile with the passed GUID.

Table 4.11: Attributes of <ProfileCommand>

Element	Type	Description
GUID	GUID (String)	The GUID of the profile
Command	String	Command String. Described below in table "Values of Command attribute"

Table 4.12: Values of Command attribute

Element	Type	Description
Query	String	Returns the entire Profile structure in <Profile>. Can be stored and used for re-submission
QueryConfigInfo	String	Returns a structure allowing to build a configuration for this profile, describing all parameters
Remove	String	Removes/Deletes the Profile. Only valid for the following profile types: Destination, Filter_Video, Filter_Audio

4.3.2 Return values

Returned is an XML string with a <Reply> root element and, depending on the Command, a <Profile> sub-element.

Table 4.13: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String, optional	If (Success==FALSE) then "Error" contains a human readable error description
<ProfileData>	Element, optional	Included for Command="Query"
<ProfileParameters>	Element, optional	Included for Command="QueryConfigInfo". See section 5.10 on page 108 for in depth information

4.3.3 Remarks

4.3.3.1 General

The <ProfileData> element returned by Command="Query" is identical to the <cnpsXML> element from a Profile as submitted by TaskType="ProfileSave".

4.3.3.2 Specialties for ProfileType="Connection"

The Command="Query" will return empty values for every connection type. You cannot store a connection profile with the ProfileSave call. Connections are only included here to provide templates for XML inclusion in eg. destination definitions (as a delivery option) or watch definition (as an aquisition watch).

4.3.4 Example

Listing 4.5: An example for a ProfileCommand call

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="ProfileCommand">
  <ProfileCommand Command="Query"
                  GUID="{9FDE86B4-A457-41A8-B6FB-572765C23296}"/>
</cnpsXML>
```

Listing 4.6: An example for a reply from ProfileCommand

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE">
  <ProfileData>
    <Sources/>
    <Destinations>
      <Module_0 ModuleGUID="{9FDE86B4-A457-41A8-B6FB-572765C23296}"
                PresetGUID="{9FDE86B4-A457-41A8-B6FB-572765C23296}">
        <ModuleData CML_P_BaseFileName="%s_HDV"
                    CML_P_Path="..."
                    ...
                    />
        <Filter_0/>
        <Filter_1/>
      </Module_0>
    </Destinations>
    <ProjectSettings />
  </ProfileData>
</Reply>
```

4.4 TaskType "TagList"

4.4.1 Parameters

Table 4.14: Sub-Elements of <cnpsXML> for TaskType="TagList"

Element	Type	Description
N/A	N/A	This command has no parameters

4.4.2 Return values

Returned is an XML string with a <Reply> root element.

Table 4.15: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE

Table 4.16: Attributes of <Reply><Preset_n>

Element	Type	Description
Name	String	The actual tag string
Description	String, optional	A description of the tag (for use in lists)

4.4.3 Example

Listing 4.7: Query

```
<?xml version="1.0" encoding="UTF-8" ?>
  <cnpsXML CarbonAPIVer="1.2" TaskType="TagList">
</cnpsXML>
```

Listing 4.8: Reply

```
<?xml version="1.0" encoding="UTF-8" ?>
<Reply Success="TRUE">
  <Preset_0
    Name="{name of the tag}"
    Description="{description of the tag}"
    />
  <Preset_1
    Name="{name of the tag}"
    Description="{description of the tag}"
    />
</Reply>
```

4.5 TaskType "DirCreate"

4.5.1 Parameters

Table 4.17: Sub-Elements of <cnpsXML> for TaskType="DirCreate"

Element	Type	Description
<DirCreate>	Element	Description of the new directory

Table 4.18: Attributes in <DirCreate>

Element	Type	Description
Path	String	The full path of the new directory including the name of the directory

4.5.2 Return values

Returned is an XML string with a <Reply> root element.

Table 4.19: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description

4.5.3 Example

Listing 4.9: Query

```
<?xml version="1.0" encoding="UTF-8" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="DirCreate">
  <DirCreate Path="{full path of directory}" />
</cnpsXML>
```

Listing 4.10: Reply

```
<?xml version="1.0" encoding="UTF-8" ?>
<Reply Success="TRUE" />
```

4.6 TaskType "DirList"

4.6.1 Parameters

Table 4.20: Sub-Elements of <cnpsXML> for TaskType="DirList"

Element	Type	Description
<DirList>	Element	Description of the directory to list

Table 4.21: Attribute in <DirList>

Element	Type	Description
Path	String	The full path to the directory for which content should be listed
ListType	String	The type of content to return (for details, see table 4.22 below)
ShowDetails.DWD	Integer, optional	A specification of how many details to include about each element (Showing more info may make the query slower) **0** Only the file or folder name **1** Include basic info about files (size, date, etc.)

Table 4.22: Values for the ListType attribute in <DirList>

Element	Type	Description
DIRS	String	The reply will only show directories (Dir_n elements)
FILES	String	The reply will only show files (File_n elements)
DIRS|FILES	String	The reply will show both directories and files (Dir_n and File_n elements)

4.6.2 Return values

Returned is an XML string with a <Reply> root element.

Table 4.23: Attributes of <Reply>

Element	Type	Description
<Dir_0>	Element	The root element (the directory for which listing is requested)
<Dir_0><Dir_n>	Element, optional	Sub directory elements (for ListType="DIRS" or ListType="DIRS|FILES")
<Dir_0><File_n>	Element, optional	File elements (for ListType="FILES" or ListType="DIRS|FILES")

Table 4.24: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description

Table 4.25: Attributes of <Reply><Dir_0><Dir_n

Element	Type	Description
Name	String	The name of the Directory

Table 4.26: Attributes of <Reply><Dir_0><File_n

Element	Type	Description
Name	String	The name of the file
Attributes	String, optional	Semicolon separated list of attributes (Archive, ReadOnly, etc.)
CreateTime	String, optional	The time the file was created (Format: YYYY-MM-DD HH:mm:ss)
LastWriteTime	String, optional	The time the file was last updated (Format: YYYY-MM-DD HH:mm:ss)
LastAccessTime	String, optional	The time the file was last accessed (Format: YYYY-MM-DD HH:mm:ss)
Size.QWD	Integer, optional	The size of the file in bytes

4.6.3 Example

Listing 4.11: Query

```
<?xml version="1.0" encoding="UTF-8" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="DirList">
  <DirList Path="C:\" ListType="DIRS|FILES" ShowDetails.DWD="1" />
</cnpsXML>
```

Listing 4.12: Reply

```
<?xml version="1.0" encoding="UTF-8" ?>
<Reply Success="TRUE">
  <Dir_0 Name="C:\">
    <Dir_0 Name="folder name 1" />
    <Dir_1 Name="folder name 2" />
    <File_0 Name="file1.txt"
              Attributes="Archive"
              CreateTime="2004-03-05 15:33:19"
              LastWriteTime="2004-03-05 15:33:20"
              LastAccessTime="2004-06-22 15:15:26"
              Size.DWD="106036" />
    <File_1 Name="file2.txt"
              Attributes="Archive"
              CreateTime="2004-02-25 16:21:11"
              LastWriteTime="2004-02-25 16:21:11"
              LastAccessTime="2004-02-25 16:21:11"
              Size.DWD="0" />
  </Dir_0>
</Reply>
```

4.7 TaskType "ServerConnectionTest"

4.7.1 Parameters

Table 4.27: Sub-Elements of <cnpsXML> for TaskType="ServerConnectionTest"

Element	Type	Description
<ServerUpload>	Element	A description of the server to test for **upload** access (eg. access to create a file)
<ServerDownload>	Element, opt.	A description of the server to test for **download** access (eg. access to read a file)

Table 4.28: Sub-elements and attributes for the <ServerUpload> element

Element	Type	Description
RemoteServerFolder	String	The folder path on the remote server
ModuleData	Element	The server description including ServerType. The structure of this element is retrieved using a ProfileCommand call on a Connection profile

Table 4.29: Sub-elements and attributes for the <ServerDownload> element

Element	Type	Description
RemoteServerFolder	String	The folder path on the remote server
ModuleData	Element	The server description including ServerType. The structure of this element is retrieved using a ProfileCommand call on a Connection profile

4.7.2 Return values

Returned is an XML string with a <Reply> root element.

Table 4.30: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description

4.7.3 Example

Listing 4.13: Query

```
<?xml version="1.0" encoding="UTF-8" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="ServerConnectionTest">
  <ServerDownload RemoteServerFolder="">
    <ModuleData ServerType="GrassValley Servers"
```

```
                ServerPrimaryAddress="192.168.3.138"
                ServerSecondaryAddress="192.168.3.139"
                ServerSecondaryLoginName="K2admin"
                ServerSecondaryLoginPassword="K2admin"
                BasePath="default" />
  </ServerDownload>
</cnpsXML>
```

Listing 4.14: Reply

```
<?xml version="1.0" encoding="UTF-8" ?>
<Reply Success="TRUE" />
```

Listing 4.15: Reply (on error)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Reply Success="FALSE" Error="Username/password incorrect"/>
```

4.8 TaskType "Version"

4.8.1 Parameters

Table 4.31: Sub-Elements of <cnpsXML> for TaskType="Version"

Element	Type	Description
N/A	N/A	This command has no parameters

4.8.2 Return values

Returned is an XML string with a <Reply> root element.

Table 4.32: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description.
Version	String	The full version number of the Carbon product

4.8.3 Example

Listing 4.16: An example for a Version call

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="Version" />
```

Listing 4.17: An example for a reply from Version

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE" Version="CarbonServer Farm ver. 03.00.32.00" />
```

4.9 TaskType "CarbonRoles"

4.9.1 Parameters

Table 4.33: Sub-Elements of <cnpsXML> for TaskType="CarbonRoles"

Element	Type	Description
N/A	N/A	This command has no parameters

4.9.2 Return values

Returned is an XML string with a <Reply> root element.

Table 4.34: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description.
CarbonRole	String	CLIENT, SERVER or AGENT depending on the role of Carbon whether it's a coder, server or an agent, respectively.

4.9.3 Example

Listing 4.18: An example for a CarbonRole call

```
<?xml version="1.0" encoding="UTF-8" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="CarbonRoles">
</cnpsXML>
```

Listing 4.19: An example for a reply from CarbonRole

```
<?xml version="1.0"?>
<Reply CarbonRole="SERVER" Success="TRUE"/>
```

4.10 TaskType "FormPresetCreate"

This saves a set of values for a form on a web page or in a desktop application. These presets can then later on be retrieved to quickly restore a set of parameters.

4.10.1 Parameters

Table 4.35: Sub-Elements of <cnpsXML> for TaskType="FormPresetCreate"

Element	Type	Description
<Preset>	Element	An element holding all the preset information that is to be stored

Table 4.36: Attributes for <Preset>

Element	Type	Description
Name	String	The pretty name of the new preset (shown in drop down dialog box)
FormName	String	The form name (a-z, A-Z, 0-9, _ (underscore) and no spaces)

Table 4.37: Attributes for <Field_n> sub elements of <Preset>

Element	Type	Description
FieldName	String	The name of the field inside the form (a-z, A-Z, 0-9, _ (underscore) and no spaces)
FieldValue	String	The value of the field

4.10.2 Return values

Returned is an XML string with a <Reply> root element.

Table 4.38: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description.

4.10.3 Example

Listing 4.20: Query

```
<?xml version="1.0" encoding="UTF-8" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="FormPresetCreate">
  <Preset Name="{name of the preset}" FormName="{name of the form}">
```

```
    <Field_0 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    <Field_1 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    <Field_2 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    ...
  </Preset>
</cnpsXML>
```

Listing 4.21: Reply

```
<?xml version="1.0" encoding="UTF-8" ?>
<Reply Success="TRUE" />
```

4.11 TaskType "FormPresetList"

This call retrieves all saved presets for a given form. They can then be listed for easy selection and form fill-out.

4.11.1 Parameters

Table 4.39: Sub-Elements of <cnpsXML> for TaskType="FormPresetList"

Element	Type	Description
<Preset>	Element	An element specifying which form to get presets for

Table 4.40: Attributes for <Preset>

Element	Type	Description
FormName	String	The name of the form, for which to retrieve presets

4.11.2 Return values

Returned is an XML string with a <Reply> root element and Preset_n subelements.

Table 4.41: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description.

Table 4.42: Attributes of Preset_n

Element	Type	Description
Name	String	The pretty name of the preset (shown in drop down dialog box)
FormName	String	The form name (a-z, A-Z, 0-9, _ (underscore) and no spaces)

Table 4.43: Attributes for <Field_n> sub elements of <Preset_n>

Element	Type	Description
FieldName	String	The name of the field inside the form (a-z, A-Z, 0-9, _ (underscore) and no spaces)
FieldValue	String	The value of the field

4.11.3 Example

Listing 4.22: Query

```
<?xml version="1.0" encoding="UTF-8" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="FormPresetList">
  <Preset FormName="{name of the form}" />
</cnpsXML>
```

Listing 4.23: Reply

```
<?xml version="1.0" encoding="UTF-8" ?>
<Reply Success="TRUE">
  <Preset_0 Name="{name of the preset}" FormName="{name of the form}">
    <Field_0 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    <Field_1 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    <Field_2 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    ...
  </Preset_0>
  <Preset_1 Name="{name of the preset}" FormName="{name of the form}">
    <Field_0 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    <Field_1 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    <Field_2 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    ...
  </Preset_1>
  <Preset_2 Name="{name of the preset}" FormName="{name of the form}">
    <Field_0 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    <Field_1 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    <Field_2 FieldName="{name of the field}"
             FieldValue="{value of the field}" />
    ...
  </Preset_2>
  ...
</Reply>
```

4.12 TaskType "FormPresetCommand"

This call is used to do operations on a single preset. A combination of `FormName` and the name of the preset is used to uniquely identify the preset to do operations on.

4.12.1 Parameters

Table 4.44: Sub-Elements of `<cnpsXML>` for `TaskType="FormPresetCommand"`

Element	Type	Description
`<FormPresetCommand>`	Element	An element specifying which presets to remove

Table 4.45: Attributes for `<FormPresetCommand>`

Element	Type	Description
`Command`	String	Command string. Described below in "Values for `Command` attribute"
`Name`	String	The pretty name of the preset (shown in drop down dialog box)
`FormName`	String	The name of the form, for which to retrieve presets

Table 4.46: Values for `Command` attribute

Element	Type	Description
`Remove`	String	Removes the form preset specified

4.12.2 Return values

Returned is an XML string with a `<Reply>` root element.

Table 4.47: Attributes of `<Reply>`

Element	Type	Description
`Success`	String	TRUE or FALSE
`Error`	String	If (Success==FALSE) then "Error" contains a human readable error description.

4.12.3 Example

Listing 4.24: Query

```
<?xml version="1.0" encoding="UTF-8" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="FormPresetCommand">
  <FormPresetCommand Command="Remove" Name="{name of the preset}" FormName="{
      name of the form}" />
</cnpsXML>
```

Listing 4.25: Reply

```
<?xml version="1.0" encoding="UTF-8" ?>
<Reply Success="TRUE" />
```

4.13 TaskType "NodeList"

Returns a list of all the nodes(agents) in a Carbon Server Farm. For further information about a particular agent, use NodeCommand, using the obtained IP address as identifier.

4.13.1 Parameters

Table 4.48: Sub-Elements of <cnpsXML> for TaskType="NodeList"

Element	Type	Description
N/A	N/A	This command has no parameters

4.13.2 Return values

Returned is an XML string with a <Reply> root element and a <NodeList> sub-element.

Table 4.49: Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String, optional	If (Success==FALSE) then "Error" contains a human readable error description

Table 4.50: Elements and attributes of <NodeList>

Element	Type	Description
Node_n	String	IP Address of node n

4.13.3 Remarks

Another TaskType very similar to NodeList worth mentioning here is "FarmList" which returns one additional attribute under the sub-element <FarmList> called "NrOfAgents.DWD". This is the count for the total number of agents in the farm.

4.13.4 Example

Listing 4.26: Query

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="NodeList" />
```

Listing 4.27: Reply

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE">
<NodeList
   Node_0="192.168.3.94"
```

```
   Node_1="192.168.3.107"
   Node_2="192.168.3.115"
   Node_3="192.168.3.109"
   Node_4="192.168.3.91"
   Node_5="192.168.3.82"
   Node_6="192.168.3.123"
   Node_7="192.168.3.89"
   Node_8="192.168.3.83"
   Node_9="192.168.3.118"
   Node_10="192.168.3.111"
   Node_11="192.168.3.117"
   Node_12="192.168.3.68"/>
</Reply>
```

4.14 TaskType "NodeCommand"

NodeCommand is the central entry point to query and modify nodes/agents.

4.14.1 Parameters

Table 4.51: Sub-Elements of <cnpsXML> for TaskType="NodeCommand"

Element	Type	Description
<NodeCommand>	Element	The element storing the NodeIP and the command to be applied to the node with the passed NodeIP

Table 4.52: Attributes of <NodeCommand>

Element	Type	Description
NodeIP	String	The IP Address of the node
Command	String	Command String. Described below in table "Values of Command attribute"

Table 4.53: Values of Command attribute

Element	Type	Description
GetProperties	String	Returns the properties of the node including the version and tag information in the element <NodeProperties>
GetServerNodeStatus	String	Returns Server Settings for the Node including Enabled.DWD, Priority.DWD, Slots.DWD and FreeSlots.DWD in the element <ServerNodeStatus>
GetNodeStatus	String	Returns Global Node Settings including Enabled.DWD, Priority.DWD and Slots.DWD in the element <NodeStatus>
SetServerNodeStatus	String	Sets Server Settings for the Node including Enabled.DWD, Priority.DWD, Slots.DWD and FreeSlots.DWD in the element <ServerNodeStatus>
SetNodeStatus	String	Sets Global Node Settings for the Node including Enabled.DWD, Priority.DWD and Slots.DWD in the element <NodeStatus>

4.14.2 Return values

Returned is an XML string with a <Reply> root element and, depending on the Node Command, a sub-element.

Table 4.54: Elements and Attributes of <Reply>

Element	Type	Description
Success	String	TRUE or FALSE
Error	String	If (Success==FALSE) then "Error" contains a human readable error description
<NodeProperties>	Element,optional	Included for Command = "GetProperties"
<ServerNodeStatus>	Element,optional	Included for Command = "GetServerNodeStatus"
<NodeStatus>	Element,optional	Included for Command = "GetNodeStatus"

4.14.3 Example

Listing 4.28: An example for a NodeCommand call

```
<?xml version="1.0" encoding="UTF-8"?>
<cnpsXML CarbonAPIVer="1.2" TaskType="NodeCommand">
  <NodeCommand Command="GetNodeStatus" NodeIP="192.168.3.82" />
</cnpsXML>
```

Listing 4.29: An example for a reply from NodeCommand

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE">
 <NodeStatus Enabled.DWD="0"
             Priority.DWD="6"
             Slots.DWD="10"/>
</Reply>
```

5

Description of valid Carbon Elements

5.1 <cnpsXML>

This is the element encapsulating the entire job. You must specify the nature of the job (basically the "command" to the Carbon Engine) here. In addition, the XML API version is declared and a name for the job can be given in an attribute you can then use this name, for example, in email notifiers.

5.1.1 Attributes

Table 5.1 below shows the standard attributes for the <cnpsXML> element.

Table 5.1: Attributes of <cnpsXML>

Element	Type	Description
CarbonAPIVer	String	Major and Minor version (string formated as: {major}.{minor} where both are integers)
TaskType	String	The name of the task. All tasks are described in chapter 3 and 4
JobName	String, optional	Name of the job or task. This can be used in notifiers, etc.
GUID	GUID (String)	In some cases a TaskType only requires a GUID
AgentIP	String, optional	The IP Address of the agent in case the command is meant for an agent in a Carbon Server Farm.

The TaskType attribute effectively determines the type of XML data submitted and the command executed by the Carbon Engine.

The CarbonAPIVer element ensures compatibility between XML structures. Only the major part of the minor number is considered for example, this document would require XML structures marked as "1.x".

5.2 <Sources>

The <Sources> element encapsulates all the sources of a project.

Multiple Sources can be handled in 2 different ways, controlled in the <ProjectSettings> element (as described in section 5.5):

1. stitched together to one logical source
2. process each source by itself, as a batch process

Each file source listed is described by the element <Module_n>

5.2.1 Valid for

The <Sources> element is valid for TaskType being one of the following:

- JobQueue
- WatchCommand with Command="SubmitSources"

5.2.2 <Sources><Module_n>

<Module_n> describes one file source, specified by its attributes.

Table 5.2: Attributes of <Sources><Module_n>

Element	Type	Description
Filename	String, optional	Full path to a Source File
RemoteFilename	String, optional	Full UNC path to a Source File located on a network share
RemoteFTPFilename	String, optional	Full path to a Source File located on a FTP site
MultiSource.DWD	Integer, optional	If the value is set to 1, this is a logical source with multiple source files for different data types.
ComplexSource.DWD	Integer, optional	If the value is set to 1, this is a complex source with one video and multiple audios. 5.2.2.2
TransitionSource.DWD	Integer, optional	If the value is set to 1, this is a transition source. 5.2.2.3

The Filename, RemoteFilename and RemoteFTPFilename attributes are mutually exclusive.

Note: If the MultiSource.DWD attribute is set to 1, only "Filename" is allowed (see 5.2.2.1).

RemoteFilename and RemoteFTPFilename will copy the source file to a local folder before attempting conversion.

If a RemoteFTPFilename is specified, an <FTPSettings> element as specified in 5.7 may be added as a sub-element of <Module_n>. If this element is not present, the global <FTPSettings> element will be referenced.

Listing 5.1: Sources element example

```
<Sources>
  <Module_0 Filename = "\\Server\Share\Folder\Example.avi"/>
</Sources>

- or -

<Sources>
  <Module_0 RemoteFTPFilename = "Entrydir\Example.avi">
   <FTPSettings FTPServer="upload.server.com"
                FTPUser="username">
                FTPPassword="Userpass" />
 </Module0>
</Sources>
```

5.2.2.1 Multi-Sources (Separate A/V files)

The Carbon engine can handle sources with separate files for each media track. Because separate files can contain multiple tracks themselves, it is required to specify what file is mapped with which data/media-type to a Carbon source file track.

To use a "multi-source", the `MultiSource.DWD` attribute of `<Sources><Module_n>` must be set to 1. In this case, the following elements/attributes are valid:

Table 5.3: Sub-Elements of of `<Module_n><ModuleData>` for `MultiSource.DWD="1"`

Element	Type	Description
`<StreamTypeTable>`	Element	Describes the assignment of source file tracks to a multi-source track
`<MultiSrcModule_n>`	Element	Describes a single source FILE - carrying attributes and sub-elements identical to the sub-element `<Module_n>` of a regular, single-source-file, `<Source>` element. See section 5.2

General concept `MultiSrcModule_0` is always a master, typically the file which was loaded first with this project before alternate media files were specified.

Audio or Video can be

- used from the "original" (`StreamType_0/1.DWD="0"`) which is the initially loaded master
- can be replaced with a separate file (`StreamType_0/1.DWD="2"`)
- can be muted (`StreamType_0/1.DWD="1"`)

If the selection is "2", the `StreamPtr.DWD` attribute must be set to the `MultiSrcModule_n` element containing the file information.

Listing 5.2: MultiSource with separate Audio and Video example

```
<Sources>
  <Module_0 MultiSource.DWD="1">
    <ModuleData>
      <StreamTypeTable StreamType_0.DWD="0"
                       StreamPtr_0.DWD="0"
                       StreamType_1.DWD="2"
                       StreamPtr_1.DWD="1"/>
      <SourceModules>
        <MultiSrcModule_0 Filename="c:\video.avi" />
        <MultiSrcModule_1 Filename="c:\audio.avi" />
      </SourceModules>
    </ModuleData>
  </Module_0>
</Sources>
```

Table 5.4: Attributes of <StreamTypeTable>

Element	Type	Description
StreamType_n.DWD	Integer	Specification of what-to-select for a particular media stream. **n=0** select stream/file for the video track **n=1** select stream/file for the audio track Value: **0** use original media (from MultiSourceModule_0) **1** use no media **2** use alternate media (from a track specified in StreamPtr_n)
StreamPtr_n.DWD	Integer	Points to a MultiSourceModule_n subelement for a mediatype n **n=0** value specifies the MultiSourceModule_n for video **n=1** value specifies the MultiSourceModule_n for audio

5.2.2.2 Complex Source

Complex Source is an API only extension of Carbon engine. It allows for the combination of one video source and multiple audio sources into a logical input source for a transcoding job. Each source can have its own In/Out points. In/Out Points for the logical source may also be specified like usual sources as described in section 5.2.3.

To use a complex source, the "ComplexSource.DWD" attribute of <Sources><Module_n> must be set to "1".

In this case the following attributes are valid for <Sources><Module_n><ModuleData><Module_n>:

Table 5.5: Attributes of of <Sources><Module_n><ModuleData><Module_n> for ComplexSource.DWD="1"

Attribute	Type	Description
ElementaryStream	String	"AUDIO" or "VIDEO"
SourceTrack.DWD	Integer	Track number in the original source. It is the pointer into the original source specifying which track to use.
ComplexSourceTrack.DWD	Integer	Track number in the logical Carbon source. It is the identifier for the logical source specifying which track to map the source track to.
Filename	String	Full path of the file to be loaded
Inpoint.QWD	64 bit Integer	In point measured in 1/27MHz units
Outpoint.QWD	64 bit Integer	Out point measured in 1/27MHz units

There are certain limitations for the use of complex source:

- Only one video track can be specified
- Maximum of 8 audio tracks can be specified
- Inpoint and Outpoint can be specified per track and also for the logical complex source itself. Therefore, it is possible to take any segment from a source file and map it to the logical source from the beginning. (In example 5.3, a 5 sec to 10 sec segment will be selected from the video file as the logical video track to be transcoded while the audio tracks will be taken completely.)
- Different tracks must have identical duration

Listing 5.3: Complex Source Example

```
<?xml version="1.0" encoding="utf-8"?>
<cnpsXML>
       <Sources>
         <Module_0 ComplexSource.DWD="1" Inpoint.QWD ="-1">
             <ModuleData>
                  <Module_0 ElementaryStream="VIDEO"
                            SourceTrack.DWD="0"
                            ComplexSourceTrack.DWD="0"
                            Filename="C:\ComplexSourceExample\10SecVideo1.m2p"
                            Inpoint.QWD="135000000"
                            Outpoint.QWD="270000000"/>
                  <Module_1 ElementaryStream="AUDIO"
                            SourceTrack.DWD="0"
                            ComplexSourceTrack.DWD="0"
```

```
                            Filename="C:\ComplexSourceExample\5Secaudio1.wav"/>
                  <Module_2 ElementaryStream="AUDIO"
                            SourceTrack.DWD="0"
                            ComplexSourceTrack.DWD="1"
                            Filename="C:\ComplexSourceExample\5Secaudio2.wav"/>
            </ModuleData>
        </Module_0>
      </Sources>
      <Destinations>
        <Module_0 ModuleGUID="{70B3CA7C-5FD2-46E1-9CCA-D82E3DEEBAB1}"
                  PresetGUID="{70B3CA7C-5FD2-46E1-9CCA-D82E3DEEBAB1}">
            <ModuleData CML_P_BaseFileName="%s_demo1_BASIC"
                        CML_P_Path="C:\ComplexSourceExample\out" DVType.DWD="1"
                        Standard.DWD="1"/>
        </Module_0>
      </Destinations>
</cnpsXML>
```

From the above example, the complex source will acquire:

- Video from 10SecVideo1.m2p
- The first audio track from the file 5SecVideo1.m2p, using the first track in that file
- The second audio track from the file 5SecAudio2.m2p, using the first track in that file

5.2.2.3 Transition Source

Transition Source is also an API only extension of Carbon engine which allows for a gradual transition from one clip to another using a distinct "wipe" across the screen as defined by the Society of Motion Picture and Television Engineers (SMPTE).

To use a TransitionSource, the "TransitionSource.DWD" attribute of `<Sources><Module_n>` must be set to "1".

In this case, the following elements/attributes are valid for `<Sources><Module_n><ModuleData>`:

Table 5.6: Attributes of `<Sources><Module_n><ModuleData>` for `TransitionSource.DWD="1"`

Attribute	Type	Description
`SMPTETransitionID.DWD`	Integer	Code of the wipe pattern
`StreamDuration.QWD`	64 bit Integer	Duration of transition measured in 1/27MHz units

A Complex Source may be used as an input to a Transition Source and vice versa. Currently, the following transitions are supported by Carbon:

- 0[1], 1, 2, 3, 4, 5, 6, 7, 8
- 41, 42, 45, 46, 101, 119, 201, 301, 302

[1] "0" is not a Wipe transition but a Dissolve from one source to another.

Listing 5.4: TransitionSource Example

```
<?xml version="1.0" encoding="utf-8"?>
<cnpsXML>
  <Sources>
    <Module_0 TransitionSource.DWD="1">
      <ModuleData SMPTETransitionID.DWD="8" StreamDuration.QWD="648000000">
        <Module_0 Filename="C:\TransitionSourceExample\Video1.mpg"
                  Inpoint.QWD="162000000" />
        <Module_1 Filename="C:\TransitionSourceExample\Video2.mpg"
                  Inpoint.QWD="162000000" />
      </ModuleData>
    </Module_0>
  </Sources>
  <Destinations>
        <Module_0 ModuleGUID="{70B3CA7C-5FD2-46E1-9CCA-D82E3DEEBAB1}" >
          <ModuleData CML_P_BaseFileName="%s_DV" CML_P_Path="C:\Out" />
        </Module_0>
  </Destinations>
  <ProjectSettings Stitching.DWD="1" />
</cnpsXML>
```

Listing 5.5: ComplexSource inside a TransitionSource Example

```
<?xml version="1.0" encoding="utf-8"?>
<cnpsXML>
  <Sources>
     <Module_0 Filename="D:\CarbonXML\Source1.mpg" />

      <Module_1 TransitionSource.DWD="1">
        <ModuleData SMPTETransitionID.DWD="1" StreamDuration.QWD="135000000">
             <Module_0 ComplexSource.DWD="1" Inpoint.QWD="135000000">
                  <ModuleData>
                        <Module_0 ElementaryStream="AUDIO"
                                  SourceTrack.DWD="0"
                                  ComplexSourceTrack.DWD="0"
                                  Filename="D:\CarbonXML\Source2-A_a_t0.m2a"/>
                        <Module_1 ElementaryStream="AUDIO"
                                  SourceTrack.DWD="1"
                                  ComplexSourceTrack.DWD="1"
                                  Filename="D:\CarbonXML\Source2-A_a_t1.m2a"/>
                        <Module_2 ElementaryStream="VIDEO"
                                  SourceTrack.DWD="0"
                                  ComplexSourceTrack.DWD="0"
                                  Filename="D:\CarbonXML\Source2-A_v.m2v"/>
                   </ModuleData>
             </Module_0>
               <Module_1 Filename="D:\CarbonXML\combiner\Source2-B.mpg"
                         Inpoint.QWD="0"/>
        </ModuleData>
      </Module_1>

    <Module_2 Filename="D:\CarbonXML\Source3.mpg"
              Inpoint.QWD="135000000" Outpoint.QWD="270000000"/>
  </Sources>
  <Destinations />
  <ProjectSettings Stitching.DWD="1" />
</cnpsXML>
```

5.2.3 <Sources><Module_n><InOutPoints>

The <InOutPoints> element contains in and out points measured in 1/27MHz units. This is used to encode only a segment from the source as specified by in and out points. Note that multiple in and out points can also be specified.

Table 5.7: Attributes of <InOutPoints>

Element	Type	Description
Inpoint_n.QWD	64 bit Integer	In point measured in 1/27MHz units
OutPoint_n.QWD	64 bit Integer	Out point measured in 1/27MHz units

Listing 5.6: In/Out Points Example

```
<Sources>
  <Module_0 Filename="c:\video.avi">
          <InOutPoints Inpoint_0.QWD="27027000" Outpoint_0.QWD="54054000"
                       Inpoint_1.QWD="66666600" Outpoint_1.QWD="86486400"/>
  </Module_0>
</Sources>
```

The legacy way of creating in and outpoints is also still supported.

Listing 5.7: Legacy In/Out Points Example

```
<Sources>
  <Module_0 Filename="\\Server\Share\Folder\Example.avi"
            Inpoint.QWD = "621621000"
            Outpoint.QWD = "675675000" />

</Sources>
```

5.2.4 <Sources><Module_n><Filter_n>

In the case of filter descriptions, the numbering (Filter_n) has nothing to do with an arbitrary enumeration Filter_0 stands for video filters, Filter_1 for Audio filters.

This numbering scheme is used because data types (such as video, audio, etc.) are represented by numbers in the Carbon engine it is thinkable that Filter_4 will contain filters for data type 4 and the engine doesnt need to know whether "4" stands for audio, video, metadata etc.

To improve readability for the API user, the attribute names <Filter_Video> and <Filter_Audio> may be used instead of <Filter_0> and <Filter_1> respectively. Usage of the former element names should not be mixed with usage of the standard (<Filter_n>)element names.

Note that in presets generated by Rhozet applications, <Filter_n> will always be used.

5.2.4.1 <Sources><Module_n><Filter_Video>

The <Filter_Video> element contains all the video filters which shall be applied to the <Module_n> source.

Note that for the filter element, alternatively but mutual exclusively to the <Filter_Video> element, <Filter_0> may be used (and will be used in preset generated by Rhozet applications). In this (and only this) case the

numbering has nothing to do with an arbitrary enumeration Filter_0 stands for video filters, Filter_1 for audio filters. This numbering scheme is used because data types (such as video, audio, etc.) are represented by numbers in the Carbon engine it is thinkable that Filter_4 will contain filters for data type 4 and the engine doesnt need to know whether "4" stands for audio, video, metadata, etc.

The <Filter_Video> element name was added purely for better readability for SDK users.

5.2.4.2 <Sources><Module_n><Filter_Video><Module_n>

<Filter_Video><Module_n> describes one video filter. The filter itself is specified in an <Module_n> attribute, Filter parameters are in the attributes of the <ModuleData> element within <Module_n>.

Table 5.8: Attributes of <Module_n>

Element	Type	Description
PresetGUID	GUID (String)	GUID of filter preset
ModuleGUID	GUID (String)	GUID of base filter module

The specification of a preset GUID is risky it relies on a prior generated preset which may not exist on, for example, a freshly installed system. If a module GUID is specified, this filter can always be loaded. Module GUID's can be determined through Carbon Admin.

Listing 5.8: Video filter example

```
<Module_0>
  <Filter_Video>
    <Module_0 PresetGUID="{68F4119B-2E96-45B5-A7B2-1DD28F107424}"/>
  </Filter_Video>
</Module_0>

- or -

<Module_0>
  <Filter_Video>
    <Module_0 ModuleGUID="{68F4119B-2E96-45B5-A7B2-1DD28F107424}"/>
  </Filter_Video>
</Module_0>
```

5.2.4.3 <Sources><Module_n><Filter_Video><Module_n><ModuleData>

The <ModuleData> element contains all the configuration parameters for one particular filter implementation in its attributes.

The filter will be initialized with its default parameter set (if ModuleGUID is used) or the preset values (if PresetGUID is used). The parameters in <ModuleData> will modify the default or preset parameters only values that differ from the default need to be specified.

Filters, as well as features and options to filters, are added to Carbon on a regular basis it is impractical to exhaustively document here each filter with all its parameters. In addition, some settings have to be encoded as binary chunks (Base64 encoded) and are not within Rhozets control (such as 3rd party codec and filter configuration data). For those reasons, it is advisable to create a project using Carbon Coder containing the filter desired, save the project file and copy the data from the proper section of that project file into your custom job description.

Listing 5.9: Temporal Noise Reducer Filter example

```
<Module_0>
  <Filter_Video>
    <Module_0 ModuleGUID="{68F4119B-2E96-45B5-A7B2-1DD28F107424}"
              PresetGUID="{68F4119B-2E96-45B5-A7B2-1DD28F107424}">
      <ModuleData Radius.DBL="7.000000"
                  LuminanceLockingThreshold.DBL="7.000000"
                  ChrominanceLockingThreshold.DBL="10.000000"
                  RefreshRate.DBL="15.000000"
                  SceneChangeDetection.DBL="30.000000" />
    </Module_0>
  </Filter_Video>
</Module_0>
```

5.2.4.4 <Sources><Module_n><Filter_Audio>

The <Filter_Audio> element contains all the audio filters that should be applied to the <Module_n> source. The attributes and sub-sections for audio filters are identical to those found in the video filters described above in section 5.2.4.1, 5.2.4.2 and 5.2.4.3.

5.2.5 <Sources><Module_n><Markers>

Each Source module can have Markers attached to it such markers can be used as chapter-points for VOB creation, for the Thumbnail extractor target or for example for WindowsMedia named markers.

Markers are specified in units of a 27 MHz clock 1 seconds for example equals to the number 27000000, similar to the in/out point and duration specification.

Each marker must be in its own sub-element and can be arbitrary (but uniquely in this hierarchy level) named. The actual time is passed in the MarkerTime_27MHz.QWD attribute of the <Markers> element. Optionally, a name can be added in the MarkerName attribute.

Ordering of the defined markers by time or name is not required. If a source has marker points included and the newly defined markers are defined for the same timestamp, the original source makers will be discarded and replaced by the user defined marker.

Listing 5.10: Source Marker example

```
<Sources>
  <Module_0>
    <Markers>
      <Marker_1 MarkerTime_27MHz.QWD="27000000" />
      <Marker_2 MarkerTime_27MHz.QWD="98000000"
                MarkerName="Marker2"/>
      <MyMarker MarkerTime_27MHz.QWD="132000000"
                MarkerName="MyMarker_Example" />
    </Markers>
  </Module_0>
</Sources>
```

5.3 <WatchSource>

The <WatchSource> element describes the properties of a Watch to be created.

5.3.1 Valid for

The <WatchSource> element is valid for TaskType being one of the following:

- WatchCreate

5.3.2 Element description

Table 5.9: Attributes of <WatchSource>

Element	Type	Description
WatchPath	String	Path of the Watches. It is heavily advised to use UNC path naming.
LeadingSource	String, optional	Path to a file that should be used as a leading clip. It is heavily advised to use UNC path naming.
TrailingSource	String, optional	Path to a file that should be used as a leading clip. It is heavily advised to use UNC path naming.
Name	String	A pretty name to show in lists, etc.
Description	String, optional	A short description of the watch
WatchSubFolder.DWD	Integer	0/1
AppendSubFolder.DWD	Integer	0/1, append the watched sub-folder to the target file name?
OverWrite.DWD	Integer	0/1, overwrite target if exists?
Priority.DWD	Integer	0 through 9, 255, Defines the Job priority when it will be established.
WatchDeleteProcessedSource.DWD	Integer	0/1, Delete the source file when finished?
Enabled.DWD	Integer	Specifies the current state of the watch folder **-1** Watch folder **incomplete** **0** Watch folder **disabled** **1** Watch folder **enabled**
GUID	GUID (String), optional	When set, uses this GUID. If another watch with this GUID exists, it will be replaced. Can be used to update watches

5.4 <Destinations>

The <Destinations> element encapsulates all the destinations of a project. Each destination is described in detail by the child element <Module_n>

5.4.1 Valid for

The <Destinations> element is valid for TaskType being one of the following:

- JobQueue
- WatchCreate

5.4.2 <Destinations><Module_n>

<Module_n> describes one destination. The base exporter to be used (AVI, MPEG, Windows Media, etc.) is specified by 1 or 2 GUIDs in its attributes, specific parameters are described in child elements.

Table 5.10: Attributes of <Destinations><Module_n>

Element	Type	Description
PresetGUID	GUID (String)	GUID of destination preset
ModuleGUID	GUID (String)	GUID of destination module
DestinationName	String	User-assigned name, serves identification when destination-specific notifiers are to be sent.
AssignedAgentIPs	String, optional	If set, assignable to these agents only. Multiple Agents are separated by ";"
AssignedTags	String, optional	If set, assignable only to an agent who exposes all of the property tags listed. Multiple Tags are separated by ";".

The specification of a preset GUID is risky it relies on a previously generated preset which may not exist on, for example, a freshly installed system. If a module GUID is specified, this exporter can always be loaded.

The exporter will be initialized with its default parameter set (if ModuleGUID is used) or the preset values (if PresetGUID is used). The parameters described in section 5.4.3 will override those parameters. For this reason, only parameters which are supposed to be different from the default or preset used need to be specified in section 5.4.3.

Listing 5.11: Destinations example

```
<Destinations>
  <Module_0 ModuleGUID="{A7264AEF-FF57-42E0-BBAD-CCF546CD515F}" />
</Destinations>

- or

<Destinations>
  <Module_0 PresetGUID="{A7264AEF-FF57-42E0-BBAD-CCF546CD515F}" />
</Destinations>
```

5.4.2.1 Mpeg-2 Transport Stream Multiplexer

This feature is an API only extension of Carbon engine. It allows to multiplex together a video stream with multiple audio and DVB/EBU-STL subtitle streams. It is intended to create DVB compliant files with multiple audio programs and DVB subtitles.

To use this function, the "MP2TSMuxer.DWD" attribute of <Destinations><Module_n> must be set to "1". This and the KernelGUID, "AFC43F6A-64E8-41fb-BB9A-2947A9A0AAD1" specified in <ProjectSettings> identify it as a special TS multiplexing job. The output is a Mpeg-2 Transport Stream, whose parameters can be set in <Destinations><Module_n><ModuleData>.

Please note that the sources for this TS multiplexing job should always be elementary streams.

In this case the following attributes are valid:

Table 5.11: Attributes of <Sources><Module_n> for MP2TSMuxer.DWD="1"

Attribute	Type	Description
Filename	String	Full path of the source file (Elementary Stream)
ElementaryStream	String	One of the following depending on the type of elementary stream: • VIDEO • VIDEO_AVC • AUDIO_AC3 • AUDIO_AC3_SYSTEM_B • AUDIO_SMPTE_302M • AUDIO_MP1L2 • AUDIO_EAC3 • AUDIO_EAC3_SYSTEM_B • AUDIO_AAC_ADTS • AUDIO_AAC_LATM • SUBTITLE_DVB • SUBTITLE_STL
FrameRateCode.DWD	Integer	This attribute is only required when the audio elementary stream is AUDIO_SMPTE_302M. Possible values are 0, 1 and 2 corresponding to video frame rates: 29.970, 25.000 and 23.976 fps respectively. The video elementary stream needs to be of the above supported frame rates only in this case.
PID.DWD	Integer	This sets the PID which is going to be used in the target.
ISO639Desc	String	ISO 639 Code e.g. "eng", "swe" etc
use_std_descriptor.DWD	Integer	Use STD Descriptor for video elementary stream. (Only valid for Mpeg-2 video elementary stream)
use_data_stream_alignment_descriptor.DWD	Integer	Use use_data_stream_alignment_descriptor for video elementary stream. (Only valid for Mpeg-2 video elementary stream)

The API supports multiplexing of following elementary streams to MPEG-2 Transport stream :

- Video: MPEG-2 or H.264 elementary streams.
- Audio: Dolby Digital (AC-3) (System A and System B), Dolby Digital Plus (E-AC-3), MPEG-1 Layer II, MP3, MPEG-2 Layer II, PCM/302M elementary streams.
- Captions: EBU-STL or DVB subtitle streams.

Table 5.12: Attributes of <Destinations><Module_n> for MP2TSMuxer.DWD="1"

Attribute	Type	Description
OutputFile	String	Full path of the target file (Transport Stream)

Table 5.13: Attributes of <Destinations><Module_n><ModuleData>

Attribute	Type	Description
pcr_pid.DWD	Integer	Elementary PID value for PCR stream. Usually, the application which uses the transport stream defines this value.
pmt_pid.DWD	Integer	PID value for PMT. Usually, the application which uses the transport stream defines this value.
use_fixed_mux_rate.DWD	Integer	Use fixed mux rate by inserting null packets/padding to avoid jitter of stream reference time. Set this option to 1 only when the application(mainly hardware) requires it.
transport_rate.DWD	Integer	Specify the transport stream multiplexing rate here. Use 0 when you want to use the default value based on the current video/audio bitrate.
pat_interval.DWD	Integer	Specify PAT interval in milliseconds
pmt_interval.DWD	Integer	Specify PMT interval in milliseconds
pcr_interval.DWD	Integer	Specify PCR interval in milliseconds
RemoveNullPackets.DWD	Integer	Use this flag to remove null packet from the output stream. The output stream may not be compliant with the ISO/IEC 13818-1. It is typically used for archiving/downloading purpose.
InsertAuInformation.DWD	Integer	Use this flag to insert AU information specified in ETSI TS 101 154 D.2.2. (Only valid for H264 video elementary stream)
UseFieldPairPerPESPacket.DWD	Integer	Use this flag to store field pair into a PES instead of storing two field pictures separately. (Only valid for H264 video elementary stream)
program_number.DWD	Integer	Program Number
transport_stream_id.DWD	Integer	Transport Stream ID

Listing 5.12: MPEG-2 TS Multiplexer Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="JobQueue" JobName="MP2TSMuxer_job">
 <Sources>
        <Module_0 Filename="c:\ElementaryVideo_MP2.m2v"
                  ElementaryStream="VIDEO"
                  PID.DWD="4096"
                  use_std_descriptor.DWD="0"
                  use_data_stream_alignment_descriptor.DWD="1" />
        <Module_1 Filename="c:\ElementaryAudio_MP2_Swedish.m2a"
                  ElementaryStream="AUDIO_MP1L2"
                  ISO639Desc="swe"
                  PID.DWD="4097" />
```

```
        <Module_2 Filename = "c:\ElementaryAudio_MP2_English.m2a"
                  ElementaryStream="AUDIO_MP1L2"
                  ISO639Desc="eng"
                  PID.DWD="4098" />
        <Module_3 Filename="C:\ElementarySubtitle_Polish.dvb"
                  ElementaryStream="SUBTITLE_DVB"
                  ISO639Desc="pol"
                  PID.DWD="4099" />
 </Sources>
 <Destinations>
        <Module_0 MP2TSMuxer.DWD="1"
                  OutputFile="C:\CarbonOut\MP2TS.m2t">
               <ModuleData pcr_pid.DWD="4096" pmt_pid.DWD="48"
                    use_fixed_mux_rate.DWD="1" transport_rate.DWD="0"
                    pat_interval.DWD="125" pmt_interval.DWD="125"
                    pcr_interval.DWD="37" RemoveNullPackets.DWD="1"
                    program_number.DWD="1990" transport_stream_id.DWD="2114"  />
        </Module_0>
 </Destinations>
 <ProjectSettings KernelGUID="{AFC43F6A-64E8-41fb-BB9A-2947A9A0AAD1}" />
</cnpsXML>
```

5.4.2.2 Mpeg-2 Transport Stream Demultiplexer

This feature is an API only extension of Carbon engine. It allows to demultiplex a transport stream into distinct elementary streams including video, audio and DVB subtitle streams. The correct PID of each elementary stream has to be specified in order to extract it.

To use this function, KernelGUID "6ADD6BFE-4856-4698-9E90-7F40790B4201" is specified in `<ProjectSettings>` to identify it as a special TS demultiplexing job. The outputs are elementary streams, whose names and output directories are set in `<Destinations><Module_n>`.

Please note that demultiplexer can split any transport stream that API multiplexer can make, except ones with EBU-STL subtitle streams.

In this case the following attributes are valid:

Table 5.14: Attributes of `<Destinations><Module_n>`

Attribute	Type	Description
`OutputFile`	String	Full path of the target file (Elementary Stream)
`PID.DWD`	Integer	PID of the Elementary Stream in the Transport Stream source

The API supports demultiplexing MPEG-2 Transport Streams into the following elementary stream types:

- Video: MPEG-2 or H.264 elementary streams.
- Audio: Dolby Digital (AC-3) (System A and System B), Dolby Digital Plus (E-AC-3), MPEG-1 Layer II, MP3, MPEG-2 Layer II, PCM/302M elementary streams.
- Captions: DVB subtitle streams. These are files containing the DVB caption subpicture data compliant with ETSI EN 300 743, wrapped in a proprietary format. The only way to use the files currently is to use them in the API multiplexer when creating DVB Transport Streams from elementary streams (see section 5.4.2.1).

Listing 5.13: MPEG-2 TS Multiplexer Example

```
<?xml version="1.0" encoding="utf-8"?>
<cnpsXML CarbonAPIVer="1.2" TaskType="JobQueue" Name="Demultiplexing Job">
  <Sources>
    <Module_0 Filename="C:\Source\Multiplexer.m2t" />
  </Sources>
  <Destinations>
   <Module_0 OutputFile="C:\Target\ElementaryVideo_MP2.m2v" PID.DWD="4096" />
   <Module_1 OutputFile="C:\Target\ElementaryAudio_MP1.m1a" PID.DWD="4097" />
   <Module_2 OutputFile="C:\Target\ElementaryAudio_AC3_1.ac3" PID.DWD="4098" />
   <Module_3 OutputFile="C:\Target\ElementaryAudio_AC3_2.ac3" PID.DWD="4099" />
   <Module_4 OutputFile="C:\Target\ElementaryAudio_WAV.wav" PID.DWD="4100" />
   <Module_5 OutputFile="C:\Target\ElementaryAudio_EC3_1.ec3" PID.DWD="4101" />
   <Module_6 OutputFile="C:\Target\ElementaryAudio_EC3_2.ec3" PID.DWD="4102" />
   <Module_7 OutputFile="C:\Target\ElementarySubtitle_DVB.dvb" PID.DWD="4103" />
  </Destinations>
  <ProjectSettings KernelGUID="{6ADD6BFE-4856-4698-9E90-7F40790B4201}" />
</cnpsXML>
```

5.4.3 <Destinations><Module_n><ModuleData>

The <ModuleData> element contains all the configuration parameters for one particular exporter implementation in its attributes.

The exporter will be initialized with its default parameter set (if ModuleGUID is used) or the preset values (if PresetGUID is used). The parameters in <ModuleData> will modify the default or preset parameters you only need to specify the values you intend to change from the default.

Exporters as well as features and settings to exporters are added to Carbon on a regular basis it is impractical to exhaustively document here each exporter with all its parameters. In addition, some settings must be encoded as binary chunks (Base64 encoded) and are not within Rhozets control (such as 3rd party codec configuration data). For these reasons, it is advisable to create a project using Carbon Coder containing the exporter desired, save the project file and copy the data from the proper section of that project file into your custom job description.

Certain attributes described below, are shared across all exporters.

Table 5.15: Common attributes of <Destinations><Module_n><ModuleData>

Element	Type	Description
CML_P_BaseFileName	String	The base file name without path specification, extension, or separation characters
CML_P_Path	String	The target path for the file(s)

Listing 5.14: MPEG Exporter example

```
<Destinations>
  <Module_0 ModuleGUID="{A7264AEF-FF57-42E0-BBAD-CCF546CD515F}">
    <ModuleData CML_P_BaseFileName="MyResultfile" CML_P_Path="T:\" stream_format
        .DWD="5" hvd_format.DWD="1" vob_marker_point.DWD="1" vob_chapter_method.
        DWD="0" vob_chapter_interval.DWD="15" use_dvd_repeat_mode.DWD="0" stream
        _type.DWD="1" video_pid.DWD="4096" video_standard.DWD="0" CML_V_SizeX.
        DWD="720" CML_V_SizeY.DWD="480" frame_rate_code.DWD="4" interlace_mode.
        DWD="1" aspect_ratio_code.DWD="2" quality_code.DWD="7" bitrate_mode.DWD=
        "2" number_of_passes.DWD="1" bitrate.DWD="6900" max_bitrate.DWD="6900"
```

```
        min_bitrate.DWD="0" ip_quant.DBL="0.800000" reservoir_buffer_size.DWD="
        1024" expense_buffer_size.DWD="1024" use_fixed_mux_rate.DWD="0"
        transport_rate.DWD="0" csps_flag.DWD="1" use_system_info_each_pack.DWD="
        1" use_buffer_info_each_pack.DWD="0" profile_level.DWD="3" use_sh_each_
        gop.DWD="1" vbv_buffer_size.DWD="224" max_gop_frames.DWD="15" max_ip_
        distance.DWD="-1" picture_structure.DWD="1" use_closed_gop.DWD="0"
        chroma_format.DWD="0" dc_precision.DWD="9" use_strict_gop_bitrate.DWD="0
        " use_dvd_compatible.DWD="1" UsedStreamAudio.DWD="1" audio_stream_type.
        DWD="1" audio_pid.DWD="4097" audio_bitrate.DWD="224000" CML_A_SampleRate
        .DWD="48000" audio_channels.DWD="3" pcm_channels.DWD="0" pcm_pid.DWD="
        4098" CML_A_SampleSize.DWD="16" audio_has_error_protection.DWD="1" pcm_
        emphasis.DWD="0" use_stream_audio.DWD="1"/>
  </Module_0>
</Destinations>
```

5.4.4 <Destinations><Module_n><Filter_n>

The filter description is identical to using source filters. Please see section 5.2.4 on page 79.

5.5 <ProjectSettings>

The values in <ProjectSettings> define parameters such as stitch mode, temporary directory, etc. for this particular project.

5.5.1 Valid for

The <ProjectSettings> element is valid for TaskType being one of the following:

- JobQueue
- WatchCreate

5.5.2 Possible locations

The <ProjectSettings> element can be located at the following locations inside the XML structure:

- <cnpsXML><ProjectSettings>

5.5.3 Element description

Listing 5.15: <ProjectSettings> example

```
<cnpsXML TaskType="{TaskType}" CarbonAPIVer="1.2">
  <ProjectSettings Stitching.DWD="1"
                   TempDir="c:\temp"
                   Priority.DWD="255"
                   AssignedAgentIPs="192.168.1.177;MYSERVER"
                   AssignedTags="HardwareCodec;FlashExporter"
                   MultiTargetSplitting.DWD="0" />
</cnpsXML>
```

Table 5.16: Attributes of <ProjectSettings>

Element	Type	Description
Stitching.DWD	Integer	Specifies whether to stitch the sources or not **0** batch encode source-by-source **1** combine all sources into one logical source
TempDir	String	Define temporary storage for this project
Priority.DWD	Integer	Priority of the job to be submitted. Scale: 0 through 9 where the lower the value, the lower the priority. The default value is 5. The special value 255 executes the job immediately, disregarding the number of simultaneous jobs allowed.
AssignedAgentIPs	String, optional	If set, assign to this agent only (required, for example, if a certain agent has hardware or codecs installed. Multiple Agents are separated by ";"
AssignedTags	String, optional	If set, assign only to an agent who exposes all of the property tags listed. Multiple Tags are separated by ";". This can be used for a more dynamic control mechanism than fixed Agent names/IPs.
MultiTargetSplitting.DWD	Integer, optional	Specifies how the job(s) is created **0** Only one job is created with all the destinations **1** One job per destination is created, each with one destination. Useful for lower latency in multinode setups
RenderMacroGrid.DWD	Integer, optional	Enables/Disables segmented grid transcoding **0** A job is not segmented for transcoding **1** A job is segmented to be transcoded in macrogrid. Useful for improved speed in multiprocessor/-multinode setups.(Note that this option will be executed only if the job is suitable for it)
MacroGridSegments.DWD	Integer, mandatory if RenderMacroGrid.DWD = "1"	Maximum number of segments a job will be segmented in to be rendered in macrogrid

5.5.4 Enabling Fixed Job-to-Machine Associations

5.5.4.1 AssignedAgentIPs

AssignedAgentIPs is a list of IP addresses (or FQDNs) where this particular job can be executed. Multiple agents can be listed and they are not considered to be in any particular order. If no agent is available, the job will remain in queued mode until an agent becomes available or the job is manually removed.

If possible, the user should avoid AssignedAgentIPs and use the more flexible AssignedTags explained below. However, tagging requires manual modification (marking) of the agents when this is not possible or to be avoided, the IP assignment described here is suitable.

5.5.4.2 AssignedTags

A list of Tags, separated by ";", can be assigned to the job. For example, a job requires a physical hardware board for decoding the source and requires an installed flash exporter. In this example, we would assign these tags to a job: "DecoderBoard;FlashExporter".

Note that Tag names can freely be assigned by the user but must be consistently used in jobs and when marking agents.

To enable association of this job to the agent machine having these components, the tags must also be set on the agent machine (Agents Job Queue Manager->Tools->Advanced Kernel Options->Machine Tags). If a job has multiple tags, it can only be associated with an agent providing all of the tags.

When using Tag names, please keep in mind that a tag name should not be a sub-set of another tag name (the reason is the used tag matching mechanism which is designed to quickly scan through fields).

For example: UsesHW and UsesHWAvid should not be used together because a search for the jobs tag "UseHW" would return successfully if an agent has the "UseHWAvid" tag assigned to it. Rather use UseHW-Generic and UseHWAvid in one set.

5.6 <EmailSettings>

The values in <EmailSettings> define the default email sending settings. Within each <EmailNotify_n> element, this element can be added as a child element, replacing this global default.

5.6.1 Valid for

The <EmailSettings> element is valid for TaskType being one of the following:

- JobQueue
- WatchCreate

5.6.2 Possible locations

The <EmailSettings> element can be located at the following locations inside the XML structure:

- <cnpsXML><EmailSettings>
- ...<EmailNotify_n><EmailSettings>

5.6.3 Element description

Table 5.17: Attributes of <EmailSettings>

Element	Type	Description
SMTPServer	String	Email server
SMTPUser	String	If login is required for sending email, then this is the user name
SMTPPassword	String	If login is required for sending email, then this is the password
SMTPSender	String	The Sender name as seen in (From:)

Listing 5.16: <EmailSettings> example

```
<cnpsXML TaskType="{TaskType}" CarbonAPIVer="1.2">
  <EmailSettings SMTPServer="mail.server.com"
                 SMTPUser="user@server.com"
                 SMTPPassword="Userpass"
                 SMTPSender="carbonadmin@rhozet.com" />
</cnpsXML>
```

5.7 <FTPSettings>

The values in <FTPSettings> are defining the default FTP upload server settings. Within each <FTPUpload_n> element, they can be locally repeated to replace this global default.

5.7.1 Valid for

The <FTPSettings> element is valid for TaskType being one of the following:

- JobQueue
- WatchCreate

5.7.2 Possible locations

The <FTPSettings> element can be located at the following locations inside the XML structure:

- <cnpsXML><FTPSettings>
- ...<FTPUpload_n><FTPSettings>
- ...<Module_n><FTPSettings>

5.7.3 Element description

Table 5.18: Attributes of <FTPSettings>

Element	Type	Description
FTPServer	String	FTP server
FTPUser	String	If login is required, then this is the user name
FTPPassword	String	If login is required, then this is the password
FTPPort.DWD	Integer	Port number

Listing 5.17: <FTPSettings> example

```
<?xml version="1.0" encoding="UTF-8"?>
<cnpsXML CarbonAPIVer="1.2" TaskType="JobQueue" JobName="myRemoteTestJob">
      <FTPSettings FTPServer="XX.XX.XXX.XX"
                   FTPUser="user"
                   FTPPassword="password"
                   FTPPort.DWD="21"  />
       <Sources>
          <Module_0 RemoteFTPFilename="/var/ftp/pub/TestClip1_DV.avi">
                <PostconversionTasks>
                  <FileDelete/>
                </PostconversionTasks>
          </Module_0>
      </Sources>
      <Destinations>
          <Module_0 PresetGUID="{B3342C66-8186-4FD4-B962-01D8E72F3B59}">
              <ModuleData CML_P_BaseFileName="Remote_Output"
                          CML_P_Path="C:\CarbonOut"/>
              <PostconversionTasks>
```

```
                <FTPUpload_0 RemoteFTPFilename="/var/ftp/pub/TestClip1_DV.m2t"
                             OverWrite.DWD="1"/>
                <FileDelete/>
            </PostconversionTasks>
        </Module_0>
    </Destinations>
</cnpsXML>
```

5.8 <Notify>

The <Notify> Element encapsulates 3 classes of notifications: pre-task, post-task, and task failure.

For each of these triggers, the notifier described in its child element is executed. The <Notify> element can be added to the root of the project, to a particular source or destination and to special tasks defined in 3.8.3.

The project itself, a source, a destination and a task go through status changes: its started, its finished or alternatively it failed. Whenever one of these cases happens, the corresponding element of the <Notify> flag is triggered.

5.8.1 Valid for

The <Notify> element is valid for TaskType being one of the following:

- JobQueue
- WatchCreate

5.8.2 Possible locations

The <Notify> element can be located at the following locations inside the XML structure:

- <cnpsXML><Notify>
- ...<Sources><Module_n><Notify>
- ...<Destinations><Module_n><Notify>
- ...<FTPUpload_n><Notify>
- ...<FileCopy_n><Notify>
- ...<ExternalProcess_n><Notify>

5.8.3 Syntax

Listing 5.18: <Notify> syntax

```
<Notify>
  <PreTaskNotify>
    <EmailNotify_n />
    <WebNotify_n />
    <ProcessNotify_n />
  </PreTaskNotify>

  <PostTaskNotify>
    <EmailNotify_n />
    <WebNotify_n />
    <ProcessNotify_n />
  </PostTaskNotify>

  <TaskErrorNotify>
    <EmailNotify_n />
    <WebNotify_n />
    <ProcessNotify_n />
```

```
  </TaskErrorNotify>
</Notify>
```

5.8.4 Notify Consideration for Farm Environments

It is advisable to use source or destination attached notifiers only for retrieving source and destination file names, not for triggering any action. For a general judgment of a job succeeding or failing, only the root notifiers should be used.

5.8.5 <Notify><[Trigger]><EmailNotify_n>

5.8.5.1 Valid for

The <EmailNotify_n> element is valid for TaskType being one of the following:

- JobQueue
- WatchCreate

5.8.5.2 Possible locations

The <EmailNotify_n> element can be located at the following locations inside the XML structure:

- ...<Notify><PreTaskNotify><EmailNotify_n>
- ...<Notify><PostTaskNotify><EmailNotify_n>
- ...<Notify><TaskErrorNotify><EmailNotify_n>

5.8.5.3 Element description

Table 5.19: Attributes of <EmailNotify_n>

Element	Type	Description
Subject	String	Email Subject
Body	String	Email Body
Recipient	String	Email Recipient, multiple recipients to be separated by semicolon(s)

In this context, the following string replacement tokens can be used for the Subject and Body attribute:

In addition to the attributes, an optional element, <EmailSettings>, can be added here to override the email server defaults specified in <cnpsXML><EmailSettings>.

Listing 5.19: <EmailNotify_n> example

```
<cnpsXML TaskType="{TaskType}" CarbonAPIVer="1.2">
  <Sources>
    <Module_0>
      <Notify>
```

```
        <PostTaskNotify>
          <EmailNotify_0 Subject="%jobname% is finished"
                         Body="the file %source% has been converted"
                         Recipient="admin@corp.com;user@corp.com" />
        </PostTaskNotify>
      </Notify>
    </Module_0>
  </Sources>
</cnpsXML>

- optional with server specification

<...>
  <Notify>
    <PostTaskNotify>
      <EmailNotify_0 Subject="%jobname% is finished"
                     Body="the file %source% has been converted"
                     Recipient="admin@corp.com;user@corp.com">
        <EmailSettings
                     SMTPServer="mail.server.com"
                     SMTPUser="user@server.com"
                     SMTPPassword="Userpass"
                     SMTPSender="carbonadmin@rhozet.com" />
      </EmailNotify_0>
    </PostTaskNotify>
  </Notify>
</...>
```

Table 5.20: <EmailNotify_n> string replacement tokens

Element	Type	Description
%jobguid%	Unique GUID for this task	Valid in all elements
%jobname%	Unique name for this task	Valid in all elements
%source%	Full source path and name	Valid in <Sources> sections only
%destinationname%	Result destination file name after conversion	Valid in <Destinations><Module_n><PostTaskNotify> and <Destinations>...<PostconversionTasks>
%errormessage%	Error message	Valid in <ConversionErrorTasks> only

5.8.6 <Notify><[Trigger]><WebNotify_n>

5.8.6.1 Valid for

The <WebNotify_n> element is valid for TaskType being one of the following:

- JobQueue
- WatchCreate

5.8.6.2 Possible locations

The <WebNotify_n> element can be located at the following locations inside the XML structure:

- ...<Notify><PreTaskNotify><WebNotify_n>
- ...<Notify><PostTaskNotify><WebNotify_n>
- ...<Notify><TaskErrorNotify><WebNotify_n>

5.8.6.3 Element description

Table 5.21: Attributes of <WebNotify_n>

Element	Type	Description
WebServiceCall	String	Full address of web service to call

In this context, the following string replacement tokens can be used for the WebServiceCall attribute:

Table 5.22: <WebNotify_n> string replacement tokens

Element	Type	Description
%jobguid%	Unique GUID for this task	Valid in all elements
%jobname%	Unique name for this task	Valid in all elements
%source%	Full source path and name	Valid in <Sources> sections only
%destinationname%	Result destination file name after conversion	Valid in <Destinations><Module_n><PostTaskNotify> and <Destinations>...<PostconversionTasks>
%errormessage%	Error message	Valid in <ConversionErrorTasks> only

Listing 5.20: <WebNotify_n> example

```
<cnpsXML TaskType="{TaskType}" CarbonAPIVer="1.2">
  <Sources>
    <Module_0>
      <Notify>
        <PostTaskNotify>
```

```
          <WebNotify_0 WebServiceCall="http://myserver.com/script.aspx?Source=%
              source%" />
        </PostTaskNotify>
      </Notify>
    </Module_0>
  </Sources>
</cnpsXML>
```

5.8.7 <Notify><[Trigger]><ProcessNotify_n>

5.8.7.1 Valid for

The <ProcessNotify_n> element is valid for TaskType being one of the following:

- JobQueue
- WatchCreate

5.8.7.2 Possible locations

The <ProcessNotify_n> element can be located at the following locations inside the XML structure:

- ...<Notify><PreTaskNotify><ProcessNotify_n>
- ...<Notify><PostTaskNotify><ProcessNotify_n>
- ...<Notify><TaskErrorNotify><ProcessNotify_n>

5.8.7.3 Element description

Table 5.23: Attributes of <ProcessNotify_n>

Element	Type	Description
NotifyCommand	String	Full path of command to execute

In this context, the following string replacement tokens can be used for the Subject and Body attribute:

Listing 5.21: <ProcessNotify_n> example

```
<cnpsXML TaskType="{TaskType}" CarbonAPIVer="1.2">
  <Sources>
    <Module_0>
      <Notify>
        <PostTaskNotify>
          <ProcessNotify_0 NotifyCommand="c:\myprog.exe %source%" />
        </PostTaskNotify>
      </Notify>
    </Module_0>
  </Sources>
</cnpsXML>
```

Table 5.24: <ProcessNotify_n> string replacement tokens

Element	Type	Description
%jobguid%	Unique GUID for this task	Valid in all elements
%jobname%	Unique name for this task	Valid in all elements
%source%	Full source path and name	Valid in <Sources> sections only
%destinationname%	Result destination file name after conversion	Valid in <Destinations><Module_n><PostTaskNotify> and <Destinations>...<PostconversionTasks>
%errormessage%	Error message	Valid in <ConversionErrorTasks> only

5.9 <PostconversionTasks> and <ConversionErrorTasks>

Additional tasks can be performed before and after conversion as well as in conversion-error cases.

5.9.1 Valid for

The <[task]> elements is valid for TaskType being one of the following:

- JobQueue
- WatchCreate

5.9.2 Possible locations

The <[task]> elements can be located at the following locations inside the XML structure:

- ...<Sources><Module_n><[task]>
- ...<Destinations><Module_n><[task]>
- ...<WatchSource><[task]>

5.9.3 <[task]><FTPUpload_n>

5.9.3.1 Possible locations

The <[task]><FTPUpload_n> elements can be located at the following locations inside the XML structure:

- ...<Sources><Module_n><PostconversionTasks><FTPUpload_n>
- ...<Sources><Module_n><ConversionErrorTasks><FTPUpload_n>
- ...<Destinations><Module_n><PostconversionTasks><FTPUpload_n>
- ...<WatchSource><PreconversionTasks><FTPUpload_n>
- ...<WatchSource><PostconversionTasks><FTPUpload_n>
- ...<WatchSource><ConversionErrorTasks><FTPUpload_n>

5.9.3.2 Element description

This Task will FTP-Upload the source or target file (or specified FTPSourceFile) depending on the location of the task element to the location specified in its attribute.

In addition to the attributes, an optional element, <FTPSettings>, can be added here to override the FTP server defaults specified in <cnpsXML><FTPSettings>.

A <Notify> element can be attached to report on the copy process.

Listing 5.22: <FTPUpload_n> example

```
<cnpsXML TaskType="JobQueue" CarbonAPIVer="1.2">
  <Sources>
    <Module_0>
      <PostconversionTasks>
        <FTPUpload_0 RemoteFTPFolder="sub\folder"
                     RemoteFTPFile="submittedFile" />
      </PostconversionTasks>
    </Module_0>
  </Sources>
</cnpsXML>

- optional with server specification

<cnpsXML TaskType="JobQueue" CarbonAPIVer="1.2">
  <Sources>
    <Module_0>
      <PostconversionTasks>
        <FTPUpload_0 RemoteFTPFolder="sub\folder"
                     RemoteFTPFile="submittedFile">
          <FTPSettings
                     FTPServer="upload.server.com"
                     FTPUser="username"
                     FTPPassword="Userpass" />
        </FTPUpload_0>
      </PostconversionTasks>
    </Module_0>
  </Sources>
</cnpsXML>
```

Table 5.25: Attributes of <FTPUpload_n>

Element	Type	Description
RemoteFTPFolder	String	Remote FTP folder
RemoteFTPFile	String	Remote File name. Optional. If not specified, the local file name will be taken
FTPSourceFile	String	(optional) a local file. If specified, the "real" source is ignored. Allows for the upload of arbitrary files.
OverWrite.DWD	Integer	0/1, overwrite target if exists? Default: 0 (delivery will fail if target exists)

5.9.4 <[task]><FileCopy_n>

5.9.4.1 Possible locations

The <[task]><FileCopy_n> elements can be located at the following locations inside the XML structure:

- ...<Sources><Module_n><PostconversionTasks><FileCopy_n>
- ...<Sources><Module_n><ConversionErrorTasks><FileCopy_n>
- ...<Destinations><Module_n><PostconversionTasks><FileCopy_n>
- ...<WatchSource><PreconversionTasks><FileCopy_n>
- ...<WatchSource><PostconversionTasks><FileCopy_n>
- ...<WatchSource><ConversionErrorTasks><FileCopy_n>

5.9.4.2 Element description

Table 5.26: Attributes of <FileCopy_n>

Element	Type	Description
TargetFolder	String	Remote folder
TargetFile	String	Remote File name. Optional.I If not specified, the local file name will be taken
OverWrite.DWD	Integer	0/1, overwrite target if exists? Default: 0 (delivery will fail if target exists)

This Task will copy the source or target file depending on its location to the location specified in its attribute.

A <Notify> element can be attached to report on the copy process.

Listing 5.23: <FileCopy_n> example

```
<cnpsXML TaskType="JobQueue" CarbonAPIVer="1.2">
  <Sources>
    <Module_0>
      <PostconversionTasks>
        <FileCopy_0 TargetFolder="\\myserver\sub\folder"
                    TargetFile="submittedFile" />
      </PostconversionTasks>
    </Module_0>
  </Sources>
</cnpsXML>
```

5.9.5 <[task]><FileDelete>

This Task will delete the source file it is attached to.

5.9.5.1 Possible locations

The <[task]><FileDelete> elements can be located at the following locations inside the XML structure:

- ...<Sources><Module_n><PostconversionTasks><FileDelete>
- ...<Sources><Module_n><ConversionErrorTasks><FileDelete>
- ...<Destinations><Module_n><PostconversionTasks><FileDelete>
- ...<WatchSource><PostconversionTasks><FileDelete>
- ...<WatchSource><ConversionErrorTasks><FileDelete>

5.9.5.2 Element description

Table 5.27: Attributes of <FileDelete>

Element	Type	Description
N/A	N/A	There is no attributes for <FileDelete>

Listing 5.24: <FileDelete> example

```
<cnpsXML TaskType="JobQueue" CarbonAPIVer="1.2">
  <Sources>
    <Module_0>
      <PostconversionTasks>
        <FileDelete />
      </PostconversionTasks>
    </Module_0>
  </Sources>
</cnpsXML>
```

5.10 <ProfileParameters>

The ProfileParameters are a detailed set of meta data for each field of a destination preset, video filter preset, audio filter preset or a collections of settings.

For each field in a preset, normally represented by an attribute in the <ModuleData> element, there will be an element in the <ProfileParameters>, more precisely in a sub element.

Depending on the ProfileType, the elements will be sub-elements of:

Destination
For destination presets, the list of element descriptions is found as sub-elements of:
<ProfileParameters><Destinations><Module_0><ModuleData>

Filter_Audio
For audio filter presets, the list of element descriptions is found as sub-elements of:
<ProfileParameters><Filter_Audio><Module_0><ModuleData>

Filter_Video
For video filter presets, the list of element descriptions is found as sub-elements of:
<ProfileParameters><Filter_Video><Module_0><ModuleData>

Setting
For settings presets, the list of element descriptions is found as sub-elements of:
<ProfileParameters><Settings><Module_0><ModuleData>

Connection
For settings presets, the list of element descriptions is found as sub-elements of:
<ProfileParameters><Connections><Module_0><ModuleData>

5.10.1 Returned info

Table 5.28: Attributes of <field name> inside the <ModuleData> element

Element	Type	Description
Name	String	Name of the field
Description	String	Description of the field
Key	String	Field group title
Control	String	Control type (see table 5.29)
DefaultValue	String	Default value
MinValue	String	Minimum value
MaxValue	String	Maximum value
VarBinarySize.DWD	Integer	Max size needed to store the value
AffectsStreamType	String	The affected stream type type (see table 5.30)
IsHidden.DWD	Integer	Is this field hidden 1/0
IsEnabled.DWD	Integer	Is this field enabled 1/0
AdvancedOption.DWD	Integer	Does this field belong to the advanced tab 1/0
CanModify.DWD	Integer	Tells if this field can be modified freely

Table 5.29: Possible values for the `Control` attribute

Value	Description
CPI_CFGTYPE_INVALID	none
CPI_CFGTYPE_SLIDER	none
CPI_CFGTYPE_CHECKBOX	none
CPI_CFGTYPE_EDIT	none
CPI_CFGTYPE_EDITINT	none
CPI_CFGTYPE_RADIOBOX_FIRST	none
CPI_CFGTYPE_RADIOBOX	none
CPI_CFGTYPE_RADIOBOX_LAST	none
CPI_CFGTYPE_LISTBOX	none
CPI_CFGTYPE_COMBOBOX	none
CPI_CFGTYPE_DROPDOWNLIST	none (see section 5.10.2.1)
CPI_CFGTYPE_EDITFLOAT	none
CPI_CFGTYPE_TITLE	none
CPI_CFGTYPE_STATIC	none
CPI_CFGTYPE_LOAD_FILE	none
CPI_CFGTYPE_SAVE_FILE	none
CPI_CFGTYPE_PIDIALOG	none
CPI_CFGTYPE_FILENAME	none
CPI_CFGTYPE_SPININT	none
CPI_CFGTYPE_PATH	none

Table 5.30: Possible values for the `AffectedStreamType` attribute

Value	Description
CPI_STYPE_INVALID	none
CPI_STYPE_VIDEO	The parameter holding this attribute affects only the video part
CPI_STYPE_AUDIO	The parameter holding this attribute affects only the audio part
CPI_STYPE_ALL	The parameter holding this attribute affects both video and audio

5.10.2 Remarks

5.10.2.1 Special note for CPI_CFGTYPE_DROPDOWNLIST

When the control type is a drop down list, all the options for the dropdown list is added as extra attributes:

Listing 5.25: Attribute sample

```
ListEntryName_0="{display text}"
listEntryValue_0="{store value}"
ListEntryName_1="{display text}"
listEntryValue_1="{store value}"
...
```

The numbering starts with 0 (zero) and doesn't have any holes.

5.10.2.2 Special note for CanModify.DWD attribute

This attribute is controlled by a file located at the path specified in

Listing 5.26: Registry path for control file

```
HKLM\Software\Rhozet\{ProductName}\Common\ParamCtrlFile
```

Listing 5.27: Sample control file

```
<?xml version="1.0"?>
<cnpsXML>
<!-- ================================== -->
<!-- Global Settings                    -->
<!-- ================================== -->
  <Globals>
    <CML_P_Path AllowEdit="1"/>
  </Globals>

<!-- ================================== -->
<!-- DV AVI Exporter base Module        -->
<!-- ================================== -->
  <_70B3CA7C-5FD2-46E1-9CCA-D82E3DEEBAB1>
    <DVCPro AllowEdit="1"/>
  </_70B3CA7C-5FD2-46E1-9CCA-D82E3DEEBAB1>

</cnpsXML>
```

5.10.3 Example

Listing 5.28: Sample Command="QueryConfigInfo" call (ProfileType="Destination")

```
<?xml version="1.0" encoding="UTF-8" ?>
<cnpsXML CarbonAPIVer="1.2" TaskType="ProfileCommand">
  <ProfileCommand Command="QueryConfigInfo"
                  GUID="{9FDE86B4-A457-41A8-B6FB-572765C23296}" />
</cnpsXML>
```

Listing 5.29: Sample reply to Command="QueryConfigInfo" (ProfileType="Destination")

```
<?xml version="1.0" encoding="UTF-8"?>
<Reply Success="TRUE">
  <ProfileParameters>
    <Sources/>
    <Destinations>
      <Module_0 ModuleGUID="{9FDE86B4-A457-41A8-B6FB-572765C23296}"
                PresetGUID="{9FDE86B4-A457-41A8-B6FB-572765C23296}">
        <ModuleData CML_P_Path="" CreatorMachine="..." FullUNCPath="">
          <Title_0
                Name="Destination"
                Control="CPI_CFGTYPE_TITLE"
                VarBinarySize.DWD="0"
                AffectsStreamType="CPI_STYPE_VIDEO"
                IsHidden.DWD="0"
                IsEnabled.DWD="1"
                AdvancedOption.DWD="1" />
          <CML_P_BaseFileName
                Name="Base Name"
                Description="..."
                Key="Destination"
                Control="CPI_CFGTYPE_FILENAME"
                DefaultValue="Untitled"
                VarBinarySize.DWD="260"
                AffectsStreamType="CPI_STYPE_ALL"
                IsHidden.DWD="1"
                IsEnabled.DWD="0"
                AdvancedOption.DWD="1" />
          <CML_P_Path
                Name="Path"
                Description="..."
                Key="Destination"
                Control="CPI_CFGTYPE_PATH"
                DefaultValue="C:"
                VarBinarySize.DWD="260"
                AffectsStreamType="CPI_STYPE_ALL"
                IsHidden.DWD="1"
                IsEnabled.DWD="0"
                AdvancedOption.DWD="1" />
          ...
        </ModuleData>
        <Filter_0/>
        <Filter_1/>
      </Module_0>
    </Destinations>
    <ProjectSettings/>
  </ProfileParameters>
</Reply>
```

5.11 <AquisitionWatches>

The <AquisitionWatches> element holds information about remote locations to check periodically for changes, and if new files arrive, copy them (or shortcuts to them) to the regular watch folder for processing. Each aquisition element is described in detail by the child element <AquisitionWatch_n>

5.11.1 Valid for

The <AquisitionWatches> element is valid for TaskType being one of the following:

- WatchCreate

5.11.2 Possible locations

The <AquisitionWatches> element can be located at the following locations inside the XML structure:

- <cnpsXML><AquisitionWatches>

5.11.3 Element description

Listing 5.30: <AquisitionWatches> example

```
<cnpsXML TaskType="{TaskType}" CarbonAPIVer="1.2">
  <AquisitionWatches />
</cnpsXML>
```

5.11.4 <AquisitionWatches><AquisitionWatch_n>

<AquisitionWatch_n> describes one aquisition element. There are different types, depending on where to look for changes. The different types are listed in table 5.33

Table 5.31: Sub elements of <AquisitionWatches><AquisitionWatch_n>

Element	Type	Description
WatchSettings	Element	Settings for this Aquisition watch
ServerDownload	Element	Server specifications for WatchMethod="WatchServer"
FileDownload	Element	Folder specifications for WatchMethod="WatchFolder"

Table 5.32: Attributes of <AquisitionWatches><AquisitionWatch_n>

Element	Type	Description
WatchMethod	String	The method to use

Table 5.33: Values of WatchMethod attribute

Element	Type	Description
WatchNone	String	Doesn't watch anything (disabled)
WatchServer	String	Watches a remote server (the type of server is specified inside ModuleData)
WatchFolder *(deprecated)*	String	Watches a remote LAN location
WatchFTP *(deprecated)*	String	Watches a remote FTP server

Table 5.34: Attributes of <AquisitionWatches><AquisitionWatch_n><WatchSettings>

Element	Type	Description
WatchSubFolders.DWD	Integer	0/1, whether to also watch remote sub folders
PropagateSubFolders.DWD	Integer	0/1, whether to propagate sub folders into the main watch (only applicable if WatchSubFolders.DWD="1")
RetrieveToRenderNode.DWD	Integer	0/1, whether to copy source file directly to the agent bypassing the server
WatchDeleteProcessedSource.DWD	Integer	0/1, whether to delete the source file on completion
WatchInterval.DWD	Integer	How often to poll the remote location for changes (unit is seconds)
TriggerFileSizeKB.DWD	Integer	The minimum file size to accept (unit in KB)
WatchWildCard	String	A filter to only select files matching (default: "*")

Table 5.35: Sub elements and attributes of <AquisitionWatches><AquisitionWatch_n><ServerDownload>

Element	Type	Description
ModuleData	Element	The description of the server connection
RemoteServerFolder	String	The remote folder to monitor

Table 5.36: Attributes of <AquisitionWatches><AquisitionWatch_n><ServerDownload><ModuleData>

Element	Type	Description
ServerGUID	String	The GUID of the server connection profile

5.12 <Rules>

This element contains Rule(s) to be applied to the watch folder. A watch folder Rule is used to perform custom actions at job execution time.

5.12.1 Valid for

The <Rules> element is valid for TaskType being one of the following:

- WatchCreate

5.12.2 Possible locations

The <Rules> element can be located at the following locations inside the XML structure:

- <cnpsXML><Rules>

5.12.3 Element description

Table 5.37: Attributes of <Rules>

Attribute	Type	Description
RuleGUID	String	One of the default GUIDs specified in table 5.38
RuleDLL	String	Full path of the Rules dll (CMLRuleDefault.dll)
RuleArguments	String	Argument (if any) that needs to be passed to the Rules dll

Listing 5.31: <Rules> example

```
<cnpsXML CarbonAPIVer="1.2" TaskType="WatchCreate">
  <Rules>
     <Rule_0 RuleGUID="{93511973-A1D4-4C98-AB94-3CA88487C606}"
             RuleDLL="C:\Program Files\Common Files\Rhozet\CarbonServer
             Farm\Kernel\CMLRULEdefault.dll"
             RuleArguments=""/>
  </Rules>
</cnpsXML>
```

Table 5.38: Default Rules provided by Carbon (in CMLRuleDefault.dll)

Rule	GUID	Description
`Same Aspect Ratio as Source`	9B241E74-D31D-4AD5-8E2D-37493227C021	Keep the aspect ratio same as source. If the target does not support this ratio, round to the nearest supported value.
`<Windows Media encoding> Keep all source properties`	F4313E80-67C7-4EA5-BC81-83FBF0840696	Keep all the sources properties and adjust the encoded bitrate to 0.25 bits per pixel. Note: This rule supports WindowsMedia encoding only, it will fail for other targets.
`Encode for 4:3 playout`	93511973-A1D4-4C98-AB94-3CA88487C606	If the source is exactly or approx. 4:3, it will be encoded as 4:3. If the source is exactly or approx. 16:9, it will be encoded as cropped 14:9. For all other ratios, the closest of the 2 options will be chosen.
`Encode for 16:9 playout`	AA6FE972-EF90-4265-BC60-E993F1E8315A	If the source is exactly or approx. 16:9, it will be encoded as 16:9. If the source is exactly or approx. 4:3, it will be encoded as cropped 14:9. For all other ratios, the closest of the 2 options will be chosen.
`Attach LUT to DPX sequence`	CDA60015-2671-41C7-B5B2-2248D77E23B1	The LUT file specified in the parameter of this rule will be attached to all source DPX files. Note: This Rule requires LUT file path in RuleArguments
`Rename Incoming File`	89454F74-DEDC-4317-A02F-815B7C9AF409	Adds a time stamp to the source file name and submits it for rendering.

5.12.4 Watch Rules DLL Usage

For a standalone environment like Carbon Coder, the rules dll (CMLRuleDefault.dll) in the kernel folder (`C:\Program Files\Common Files\Rhozet\<Product>\Kernel`) will be automatically detected by Carbon Admin.

For Server - agent environment the DLL has to be in a shared location that is accessible for both Server and Agent.

Part II

Appendix

A

JobEvaluate PublicDescription Attributes

Applies to Carbon 2.0 and up

JobEvaluate returns a PublicDescription element with common attributes and values across all different sources.

Mandatory Attributes are described in section 3.1 on page 17; note that the attributes mentioned here are optional, they are not guaranteed to be available.

Rhozet will strive to make support for those attributes as complete as possible across different modules.

To simplify maintenance, these are actual code excerpts. The string in double quotes is the actual attribute value returned.

Listing A.1: `PublicDescription` attribute `RZMETA_VideoCodecName`

```
UNCOMPRESSED_PICTURE_CODING                = "04.01.02.01.00.00.00.00";
COMPRESSED_PICTURE_CODING                  = "04.01.02.02.00.00.00.00";
MPEG_COMPRESSION                           = "04.01.02.02.01.00.00.00";
MPEG_COMPRESSION_MP_ML                     = "04.01.02.02.01.01.00.00";
MPEG_COMPRESSION_422P_ML                   = "04.01.02.02.01.02.00.00";
MPEG_COMPRESSION_422P_ML_D10               = "04.01.02.02.01.02.01.00";
```

```
DV_COMPRESSION                              = "04.01.02.02.02.00.00.00";
DV_COMPRESSION_IEC61834                     = "04.01.02.02.02.01.00.00";
DV_COMPRESSION_IEC_DV25_525_60              = "04.01.02.02.02.01.01.00";
DV_COMPRESSION_IEC_DV25_625_50              = "04.01.02.02.02.01.02.00";
DV_COMPRESSION_IEC_DV25_525_60_DVCAM1       = "04.01.02.02.02.01.03.00";
DV_COMPRESSION_IEC_DV25_625_50_DVCAM1       = "04.01.02.02.02.01.04.00";

DV_BASED                                    = "04.01.02.02.02.02.00.00";
DV_BASED_DV25_525_60                        = "04.01.02.02.02.02.01.00";
DV_BASED_DV25_625_50                        = "04.01.02.02.02.02.02.00";
DV_BASED_DV50_525_60                        = "04.01.02.02.02.02.03.00";
DV_BASED_DV50_625_50                        = "04.01.02.02.02.02.04.00";

JPEG_2000_IEC_15444                         = "04.01.02.02.03.01.01.00";

MPEG1_COMPRESSION                           = "RZ_VIDEO_CODEC_MPEG1";
H264_COMPRESSION                            = "RZ_VIDEO_CODEC_H264";
DPX_COMPRESSION                             = "RZ_VIDEO_CODEC_DPX";
STILL_IMAGE                                 = "RZ_VIDEO_CODEC_STILL_IMAGE";
CAPTURE_VIDEO                               = "RZ_VIDEO_CODEC_CAPTURE_VIDEO";
MPEG4_COMPRESSION                           = "RZ_VIDEO_CODEC_MPEG4";
```

Listing A.2: PublicDescription attribute RZMETA_AudioCodecName

```
UNCOMPRESSED_SOUND_CODING                   = "04.02.02.01.00.00.00.00";
COMPRESSED_SOUND_CODING                     = "04.02.02.02.00.00.00.00";
COMPRESSED_AUDIO_CODING                     = "04.02.02.02.03.00.00.00";
SMPTE_338M_AUDIO_CODING                     = "04.02.02.02.03.02.00.00";
DOLBY_AC3                                   = "04.02.02.02.03.02.01.00";
MPEG1_LAYER1                                = "04.02.02.02.03.02.04.00";
MPEG1_LAYER2_OR_LAYER3                      = "04.02.02.02.03.02.05.00";
MPEG2_LAYER1                                = "04.02.02.02.03.02.06.00";
DOLBY_E                                     = "04.02.02.02.03.02.06.00";
```

Listing A.3: PublicDescription attribute RZMETA_WrapperCompliance

```
WRAPPER_SMPTE_360M                          = "WRAPPER_SMPTE_360M";
WRAPPER_SMPTE_377M                          = "WRAPPER_SMPTE_377M";
WRAPPER_SMPTE_378M                          = "WRAPPER_SMPTE_378M";
WRAPPER_SMPTE_390M                          = "WRAPPER_SMPTE_390M";

WRAPPER_LEITCH                              = "WRAPPER_LEITCH";
WRAPPER_LEITCH_VERSION_0                    = "WRAPPER_LEITCH_VERSION_0";
WRAPPER_LEITCH_VERSION_1                    = "WRAPPER_LEITCH_VERSION_1";
WRAPPER_OMNEON_QUICKTIME_REFERENCE          = "WRAPPER_QUICKTIME_REFERENCE_OMNEON"
    ;
```

Note: Gxf, Lxf, Omneon and MXF sources are good examples for the implementation use one of these in JobEvaluate to see an actual return chunk.

B

XML Conventions for older API versions

JobType/TaskType

Starting with API version 1.25, the name "Task" was used to refer to any one of a number of activities a Carbon system can perform, whereas the name "Job" is used to refer to an actual transcoding activity only. In previous versions of the API a "JobType" attribute was used to refer to any type of Carbon task, in 1.25 and later the proper nomenclature for this type of attribute is "TaskType".

Starting with API version 1.25, both naming schemes were supported, to aid in migration and implementation of legacy tools and interfaces. Starting with API version 2.0 the attribute "JobType" will be deprecated and can then no longer be used.

This document already refers to "TaskType" exclusively, even though the current software (1.26.03 and up) supports both naming conventions.

GUID/Guid

The "GUID" attribute name in Carbon was originally spelled "Guid". The Carbon engine currently accepts both the old usage "Guid" and the newer recommended usage "GUID" , to aid in migration and implementation of legacy tools and interfaces. It will also write both attributes when passing an XML reply structure back to the caller. Currently you are encouraged to ignore the "Guid" attribute and use the "GUID" attribute exclusively. Starting with API version 2.0 the attribute "Guid" will be deprecated and can then no longer be used.

This document already refers to "GUID" exclusively, even though the current software (1.26.03 and up) supports both naming conventions.

State/Status

The "Status" attribute name in Carbon originally was originally spelled "State" and carried different results (NEX_*).

The Carbon engine currently accepts both the old usage "State" and the newer recommended usage "Status", to aid in migration and implementation of legacy tools and interfaces. It will also write both attributes when passing an XML reply structure back to the caller. Currently you are encouraged to ignore the "State" attribute and use the "Status" attribute exclusively. Starting with API version 2.0 the attribute "State" will be deprecated and can then no longer be used.

This document already refers to "Status" exclusively, even though the current software (1.26.03 and up) supports both naming conventions.

Post/Pre Conversion Tasks

The "PostconversionTasks" and "PreconversionTasks" attribute name in Carbon were originally spelled "PostConversionTasks" and "PreConversionTasks".

Starting with API version 1.31, both naming schemes were supported, to aid in migration and implementation of legacy tools and interfaces. Starting with API version 2.0 the attributes "PostConversionTasks" and "PreConversionTasks" will be deprecated and can then no longer be used.

This document already refers to "PostconversionTasks" and "PreconversionTasks" exclusively, even though the current software (1.26.03 and up) supports both naming conventions.

Priority.DWD/JobPriority.DWD

Some older versions of the API has been using `JobPriority.DWD` in some places.

OverWrite.DWD/OverwriteTarget.DWD

Some older versions of the API has been using `OverwriteTarget.DWD` in some places.

Embedded Filters

Embedded Filters in this document are filters with an XML interface in addition to the configuration values exposed through standard XML configuration attributes.

C.1 XML Titler

The XML Titler video filter uses data from an XML file to add titles and images to a video stream. It can be supplied with title information in two ways either by pointing to an external XML file or by supplying the same set of data within a complete job-XML description.

The syntax is identical in both cases, if a file is supplied, it must be specified in the ModuleData element as a Filename attribute.

Listing C.1: XMLTitler example

```
<[titler filter module]>
  <ModuleData Filename="myfile.xml" />
</[titler filter module]>
```

If the sub-element <TitlerData> of <ModuleData> exists, the file name will be ignored and the data embedded in <TitlerData> will be used for the XML Titler.

Each title is embedded in a <Data> sub-element of <TitleData>.The XML files to use with the titler look like this:

Listing C.2: XMLTitler example 2

```
<?xml version="1.0"?>

<TitlerData>

   <Data StartTime='0.5' EndTime='4.0' Title='First title'
         CharSize='0.45' PosX='0.5' PosY='0.75'
         ColorR='220' ColorG='220' ColorB='220'
         Transparency='0.0' ShadowSize='0.5' />

   <Data StartTime='4.5' EndTime='13.5'
         Title='This is the second title'
         CharSize='0.25'/>

   <Data StartTime='16.0' EndTime='21.0' Title='A third title' />

</TitlerData>
```

C.1.1 New features of the XML titler in version 3.0

- Text can now be faded in and out (see 3e and 3f)
- Additional options have been added to the Shadow Ability (see 3n)
- Titles can now be aligned with the video (see 3t and 3u)
- Images can be offset from the top left position of the video (see 3d and 3e)
- Images are now scalable (see 3f and 3g)
- Transparency now applies to images and titles (see 3l and 3h)

C.1.2 XML Elements and Attributes for Text Files

1. `<?xml version="1.0"?>`
   This is the XML file header and must be present.

2. `<TitlerData>`
   This tag indicates the start of the titler data block. The block must be closed with the `</TitlerData>` entry. XML titler files must contain exactly one titler data block, in which there will be one data entry for each title to display.

3. Data entries: Each data entry starts with the tag Data and describes one title to be displayed. It may contain several parameters:

   a) **Font:** specify the name of the font, for instance "Arial" Only one font may be selected per titler project. If more than one font is present in the titler's XML data, only the first one will be used. If the specified font is not installed or cannot be loaded, the XML titler will fail (i.e. a different font will not be used as a fallback).

   b) **FontCharSet:** specify the character set to use for the font. Allowed values are:

      i. ANSI
      ii. BALTIC
      iii. CHINESEBIG5
      iv. DEFAULT
      v. EASTEUROPE
      vi. GB2312
      vii. GREEK
      viii. HANGUL
      ix. MAC
      x. OEM
      xi. RUSSIAN
      xii. SHIFTJIS
      xiii. SYMBOL
      xiv. TURKISH
      xv. VIETNAMESE
      xvi. JOHAB
      xvii. ARABIC
      xviii. HEBREW
      xix. THAI

   c) **StartTime:** the time, in seconds, at which the display of this data entry's title will start.
   StartTimecode: an alternative to StartTime. Instead of using a floating-point value like '5.5' for 5.5 seconds, you can use a time code value ('00:00:05:15').

   d) **EndTime:** the time, in seconds, at which the display of this data entry's title will end.
   EndTimecode: an alternative to EndTime. Instead of using a floating-point value like '5.5' for 5.5 seconds, you can use a time code value ('00:00:05:15').

   e) **FadeInTime:** duration, in seconds, for which the title or image will fade in.

   f) **FadeOutTime:** duration, in seconds, for which the title or image will fade out.

   g) **Title:** the title text to display.

      i. **Italics:** The italic function is used much like `<i>` and `</i>` in HTML. Within the Title string value, use "ital;" to begin the italics section and "/ital;" to end the italics.

Listing C.3: Sample

```
<xml version="1.0">
<TitlerData>
     <Data Title='ital;This text will be italic /ital;'/>
</TitlerData>
```

ii. **Line Breaks:** insert a line break, within the Title string use the two bytes "0D 0A" to mark the end of line, or if editing XML data with Notepad, simply press the Enter key at the desired location of the line break.

h) **CharSize:** Value between 0.0 and 1.0, which controls the size of the characters.

i) **PosX:** Value between 0.0 (left of the screen) and 1.0 (right of the screen) used to position the title horizontally.

j) **PosY:** Value between 0.0 (top of the screen) and 1.0 (bottom of the screen) used to position the title vertically.

k) **ColorR, ColorG, ColorB:** Values between 0.0 and 255.0, which respectively describe the amount of Red, Green and Blue to use for the title's color.

l) **Transparency:** Value between 0.0 (fully opaque) and 1.0 (fully transparent), which determines the title's transparency.

m) **ShadowSize:** value between 0.0 (no glow) and 1.0 (strongest glow),which determines the glow, or shadow, around the title used to enhance its readability.

n) **HardShadow:** binary value of 0 or 1. 0: normal shadow 1: shadow with a hard border.

o) **BkgEnable:** set value to 1 to enable use of black background behind text area, otherwise set to 0.

p) **BkgSemiTransparent:** set value to 1 to make text background semi-transparent, otherwise set to 0. Only takes effect to BkgEnable to set to 1.

q) **BkgExtraWidth:** specifies how much wider than the text the background should be. This value is relative to the width of the image; so setting it to 0.05 would extend the background by 5% of the image's width.

r) **BkgExtraHeight:** specifies how much taller than the text the background should be. This value is relative to the height of the image; so setting it to 0.05 would extend the background by 5% of the image's height.

s) **RightToLeft:** set value to 1 for the text to be displayed in right-to-left order, otherwise set to 0.

t) **HAlign:** horizontal alignment of the title

i. **0:** Center
ii. **1:** Left
iii. **2:** Right

u) **VAlign:** vertical alignment of the title

i. **0:** centered around the first line
ii. **1:** centered
iii. **2:** top
iv. **3:** bottom

C.1.3 XML Elements and Attributes for Image Titles

When using image scripting, most of the XML tags previously defined for text (such as background, color, and position) don't apply to images. Supported image formats are tif, png, tga, and psd.

1. `<?xml version="1.0"?>`
   This is the XML file header and must be present.

2. `<TitlerData>`
   This tag indicates the start of the titler data block. The block must be closed with the `</TitlerData>` entry. XML titler files must contain exactly one titler data block, in which there will be one data entry for each title to display.

3. Data Entries

   a) **StartTime:** the time, in seconds, at which the display of this data entry's title will start.
   StartTimecode: an alternative to StartTime. Instead of using a floating-point value like '5.5' for 5.5 seconds, you can use a time code value ('00:00:05:15').

   b) **EndTime:** the time, in seconds, at which the display of this data entry's title will end.
   EndTimecode: an alternative to EndTime. Instead of using a floating-point value like '5.5' for 5.5 seconds, you can use a time code value ('00:00:05:15').

   c) **Image:** the local path and filename for the image file to be used in the title.

   Listing C.4: Sample

```
<?xml version="1.0">
<TitlerData>
        <Data StartTime='0' EndTime='6.1' Image='E:\alphatest.tif'/>
        <Data StartTime='7' EndTime='26.1' Image='E:\betatest.png'/>
</TitlerData>
```

   d) **ImageOffsetX:** image horizontal offset, in pixels. Please note that "0" denotes the far left and increasing values will offset the image to the right.

   e) **ImageOffsetY:** image vertical offset, in pixels. Please note that "0" denotes the top of the image and increasing values will offset the image towards the bottom.

   f) **ImageScaleX:** image horizontal scale factor. Eg. 1.0 is actual size, 0.5 is half size, and 2.0 is double original size.

   g) **ImageScaleY:** image vertical scale factor. Eg. 1.0 is actual size, 0.5 is half size, and 2.0 is double original size.

   h) **Transparency:** Value between 0.0 (fully opaque) and 1.0 (fully transparent), which determines the image transparency.

C.1.4 Additional notes

- The first data entry should define all the different parameters.
- If a data entry does not set a parameter, the last value set for this parameter will be used for that data entry. For instance, if you want a transparency value of 0.2, you may put `Transparency="0.2"` only in the first data entry, and that value will be used for all other data entries.
- The title first line is always centered around the position described by PosX and PosY.
- The data entries must be sorted according to start time in the XML file.
- Unless specified, titles are displayed on a single line. The XML titler will not add a line break if a line is too long. It will be clipped from the left and right. Line break characters must be inserted in the Title in order for a line break to occur.
- Only one font may be used per XML titler filter instance.
- When using StartTime/EndTime for NTSC, in order to identify the start and end time of the titles, make sure to use drop-frame time code. Non-drop-frame time code approximates the real frame rate and will drift away from the actual time by a few frames every minute.

- When using StartTimecode/EndTimecode, both drop-frame and non-drop-frame time codes are supported. Drop-frame is in the format 'hh:mm:ss;ff', while non-drop-frame time code's format is 'hh:mm:ss:ff'. Notice that seconds and frames are separated by a semi-colon in drop-frame time code.
- When using StartTimecode/EndTimecode, the title display timing will not be accurate in the video filter's preview (Video Filter tab of Advanced - Filter).
- You can use Internet Explorer, or a number of other tools, to verify if your XML file has correct XML syntax. Simply open your file in Internet Explorer - if any errors related to the XML syntax exist, Internet explorer will display them.
- You can mix image data with regular data, but if a Data entry has an image, then its title will be ignored.
- The images should be the same size as the video you are rendering onto, otherwise they will be copied starting at the top-left corner of the image, with 1:1 pixel mapping

D

Change Log

Table D.1: Change log for this document

Revision	Date	Comment
02.00.00	04/20/06	Changed layout and graphics of section 1
01.23.16		Added SMTPSender Attribute to EmailSettings element
01.23.16		Added Priority to ProjectSettings
01.24.01		Added TaskType identifier to cnpsXML element to identify different tasks
01.24.01		Added CarbonAPI Version attribute to cnpsXML element
01.24.01		Added references to valid Job Types for all base elements
01.25.01		Merged Server Interface into this document, making it an overall-Carbon-Programming guide. Currently only job+watch modification commands included.
01.25.02		Added JobEvaluate
01.25.03		Transitioned and verified all Jobxxx commands to this API
01.25.04		Added Multi-Src chapter Slight changes on the WatchList command (one attribute moved)
01.25.05		Added a chapter about interfaces to Carbon API documentation. Added "SubmitSources" to "WatchCommand" for submitting source files to a watch folder/server drop box without copying them.
01.25.06		Added and consolidated remarks about interface v.2.0 and how to prepare. Changed JobType to TaskType
01.25.07		Added Carbon Introduction chapter
01.25.08		Introduced the concept of copy-source-to-local-folder before conversion. For this, RemoteFilename and RemoteFolder needed to be changed for the FTP case to RemoteFTPFilename, RemoteFTPFolder. FTPPort.DWD Attribute added
01.25.09		PostconversionTasks and PreconversionTasks spelling made consistent
01.25.10		Added Overwrite.DWD to FTPUpload and FileCopy
01.25.11		Added AssignedAgentIPs and AssignedTags for controlling job<->machine associations. See chapter 3.5.1 for more information.
01.25.12		Added AssignedAgentIPs and AssignedTags to Targets
01.25.14		Added XML Titler description Added Appendix 2 Minor Corrections
01.30.01		Exposed Profile Management. Note that while both Destinations and Filters are documented, the most recent version of the software implements destinations only.
01.40.01	08/10/07	Added "Carbon Advanced Tasks" chapter. Added JobStatusList TaskType. Added Enable and Disable commands to "WatchCommand". Updated "Profile*" TaskTypes to include "Setting".
01.40.02	08/22/07	Added ServerConnectionTest TaskType. Updated "Profile*" TaskTypes to include "Connection".
01.40.03	08/29/07	Added additional Profile Flags for the Connection profile type
01.40.04	08/29/07	Added further information about aquisition watches
01.40.05	04/11/08	Added TaskType "NodeList" and "NodeCommand" under Carbon Advanced Tasks
01.40.05	04/23/08	Added RenderMacroGrid.DWD and MacroGridSegments.DWD as attributes of `<ProjectSettings>` in Table 5.9
01.40.05	04/29/08	Updated XML Titler
01.40.05	05/02/08	Updated list of Audio/Video Filter GUIDs and removed Exporter GUIDs
01.40.05	05/13/08	Updated Table 3.16 and Example 3.4.4 for "JobStatusList". Also Table 3.21 and Example 3.5.4 for "JobCommand"
01.40.05	06/16/08	Updated Table 3.6 for TaskType "JobQueue" to include attributes. Replaced all occurrences of "CreateWatch" with "WatchCreate"
01.40.05	07/08/08	Removed list of Audio/Video Filter GUIDs from Appendix
01.40.05	08/06/08	Added subsection 5.2.3 to describe the element <InOutPoints>. The legacy way to specify in/out points is still supported though.
01.40.06	03/04/09	Added sections to expose new API functionality: Complex Source, Transition Source , Rules and TS Multiplexer.
01.40.06	05/07/09	Added TaskType "CarbonRole"
01.40.07	05/25/10	Added new section: TS Demultiplexer
3.15.0	05/25/10	Changed document version number to reflect Carbon Coder Build number.
3.16.0	08/23/10	Removed `<PreconversionTasks>` for "JobQueue" TaskType since it's not supported.
3.16.0	08/31/10	Added new supported parameters to TS multiplexer - section 5.4.2.1

Index