Leidraad VISI-systematiek versie 1.6

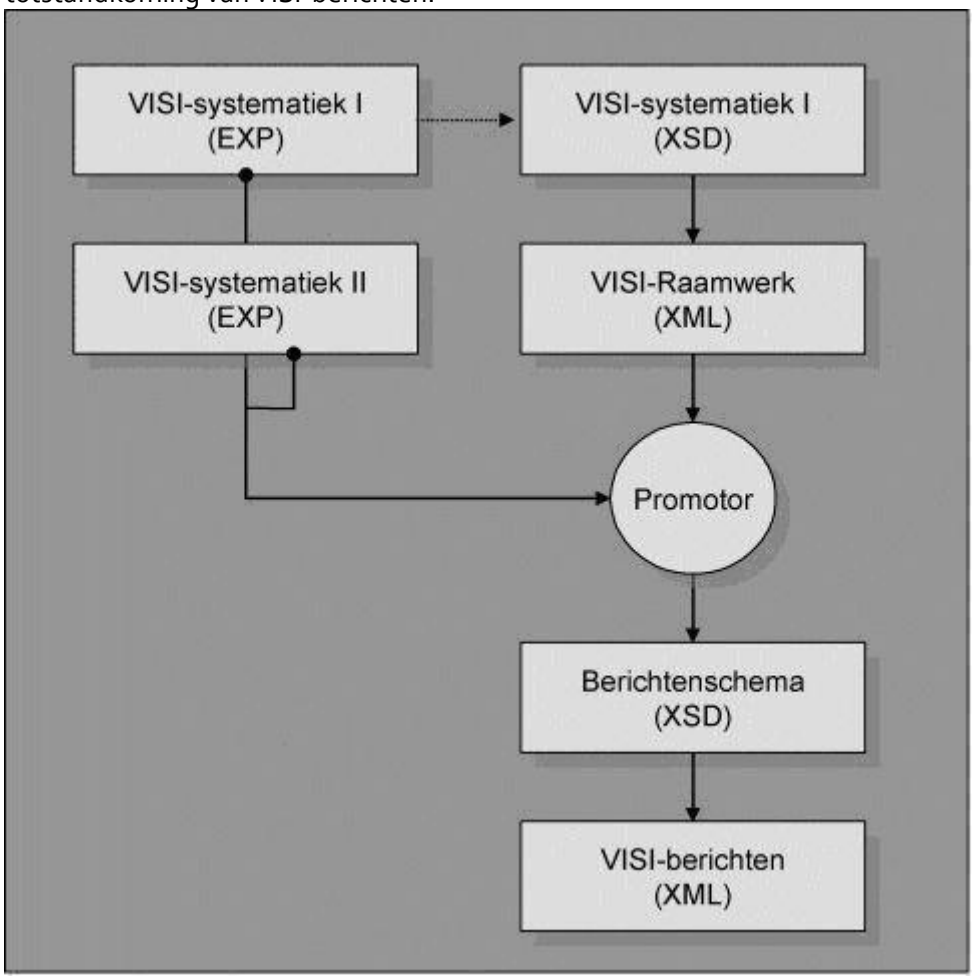
Bijlage 9
Toelichting op de werking van de Promotor

Informatief

Documentversie: 1.1
Datum: april 2019
Status: Definitief

Promotor

In een VISI-raamwerk worden de berichten beschreven die in een project gebruikt kunnen worden. De VISI-berichten zullen moeten voldoen aan een (berichten) schema dat gebaseerd is op het raamwerk. Het berichtenschema kan automatisch gegenereerd worden met een programma dat Promotor genoemd wordt. De promotor gebruikt een VISI-raamwerk en de VISI-systematiek I en II als invoer en produceert een schema (XSD) waarin alle mogelijke berichten inclusief toebehoren zijn vastgelegd. Het schema in de volgende figuur laat zien wat de plaats is van de Promotor in de totstandkoming van VISI-berichten.



Figuur 1 De Promotor dient om een Berichtenschema (XSD) te genereren.

De Promotor wordt vrij ter beschikking gesteld in de vorm van een Dynamically Linked Library (DLL) Dit is een Windows bibliotheek met functies, die door meerdere applicaties gebruikt kunnen worden. Het is hierdoor mogelijk om de Promotor te integreren met andere software.

Actuele versie

De actuele versie van de Promotor is te downloaden van de SVN-server, via de gratis tool TortoiseSVN. Toegang tot de SVN-server kan worden verkregen via CROW.

Download VISI Promote Software via de VISI-website (www.crow.nl/visi).

Eerdere versies

Eerdere versies van de promotor betreffen o.a.:
- Promote software versie 0.1 d.d. 10 oktober 2006 (zip-file 4,4Mb)
- Promote software versie 0.2 d.d. 29 november 2007 (zip-file 10,2Mb)

Deze versies zijn verouderd en sluiten niet meer volledig aan bij de meest actuele versie van de VISI-systematiek.

Disclaimer

De Promotor software en documentatie wordt vrij ter beschikking gesteld zonder enige garantie. CROW en degenen die aan de ontwikkeling van de Promotor hebben meegewerkt, hebben dat zorgvuldig gedaan, naar de laatste stand van wetenschap en techniek.
Desondanks kunnen er fouten in de software of documentatie voorkomen. Gebruikers aanvaarden het risico daarvan. CROW sluit, mede ten behoeve van degenen die aan deze publicatie hebben meegewerkt, iedere aansprakelijkheid uit voor schade die mocht voortvloeien uit het gebruik van de Promotor.

Principles for Promotor algorithm

A Promotor is an algorithm that turns an interaction framework into an interaction schema.
De Promotor is een algoritme dat een raamwerk transformeert in een interactieschema.

Since an interaction framework is recorded in XML and an interaction schema in XSD one may conclude that a promotor abstracts the data elements of the interaction framework into the entity-relation structure of an interaction schema.
Een raamwerk is vastgelegd in XML en een transactieschema in XSD

VISI-berichten die verstuurd en ontvangen worden, moeten voldoen aan een berichtenschema dat gebaseerd is op een VISI-raamwerk (xml). Het berichtenschema kan automatisch gegenereerd worden met deze service.

The basic algorithm of a promotor raises all "*Type" data elements of the interaction framework (RoleType, TransactionType, MessageType, ComplexElementType, SimpleElementType, etc.) into XSD complex type elements. Of course, certain rules must be obeyed to guarantee that the resulting XSD is a meaningful XML schema. For example, the ID attribute value of a *Type data element is interpreted as the name of its XSD complex element counterpart. As a result, the ID attribute in an interaction framework shouldn't be something like ID="Role003" but more like ID="Project_Manager".
The XSD is organized in such a way that the MessageTypes are the primary complex elements to structure the contents of a message at the level of the actual exchange in a transaction. Such a message should contain enough information to trace the actual position of this message in the total flow of messages and to determine which succeeding messages can be dispatched from this position.
To generate the correct attribute types and relation types for the resulting XSD the promoter consults a file that describes which template should be used for which complex element type (templates file).

The target namespace attribute in the XSD will be filled with the namespace attribute of the ProjectType entity of the framework XML. This helps to satisfy the requirement that each framework should be identified uniquely by its namespace.

DLL API

De volgende API calls zijn beschikbaar:

```
int  Convert__2into__3(char ** rInfo);
int  Convert__9into_10(char ** rInfo);
int  Get__3(char ** __3, char ** rInfo, char * nameSpace, char * URI);
int  Get_10(char ** _10, char ** rInfo, char * nameSpace, char * URI);
int  Promote__2__5__7into__9(char ** rInfo);
int  Set__2(char * __2, char ** rInfo);
int  Set__5(char * __5, char ** rInfo);
int  Set__7(char * __7, char ** rInfo);
char * GetErrorCode(int errorCode);
```

In de hierop volgende pagina's zal worden uitgelegd waar deze API calls betrekking op hebben.

Om een raamwerk te promoten is de volgende combinatie te gebruiken:

```
int  Set__2(char * __2, char ** rInfo);
int  Set__5(char * __5, char ** rInfo); //    it is also possible to change
                                        //    order with Set__2
int  Set__7(char * __7, char ** rInfo);
int  Promote__2__5__7into__9(char ** rInfo);
int  Convert__9into_10(char ** rInfo);
int  Get_10(char ** _10, char ** rInfo, char * nameSpace, char * URI);
```

Om een Schema te genereren waartegen een Raamwerk valide moet zijn:

```
int  Set__2(char * __2, char ** rInfo);
int  Convert__2into__3(char ** rInfo);
int  Get__3(char ** __3, char ** rInfo, char * nameSpace, char * URI);
```

int **Convert__2into__3**(*char* ** rInfo);

Converts internally an express schema loaded by Set__2 API call to XSD internal format.

VISI Structuur

_2.EXP input

Parameters

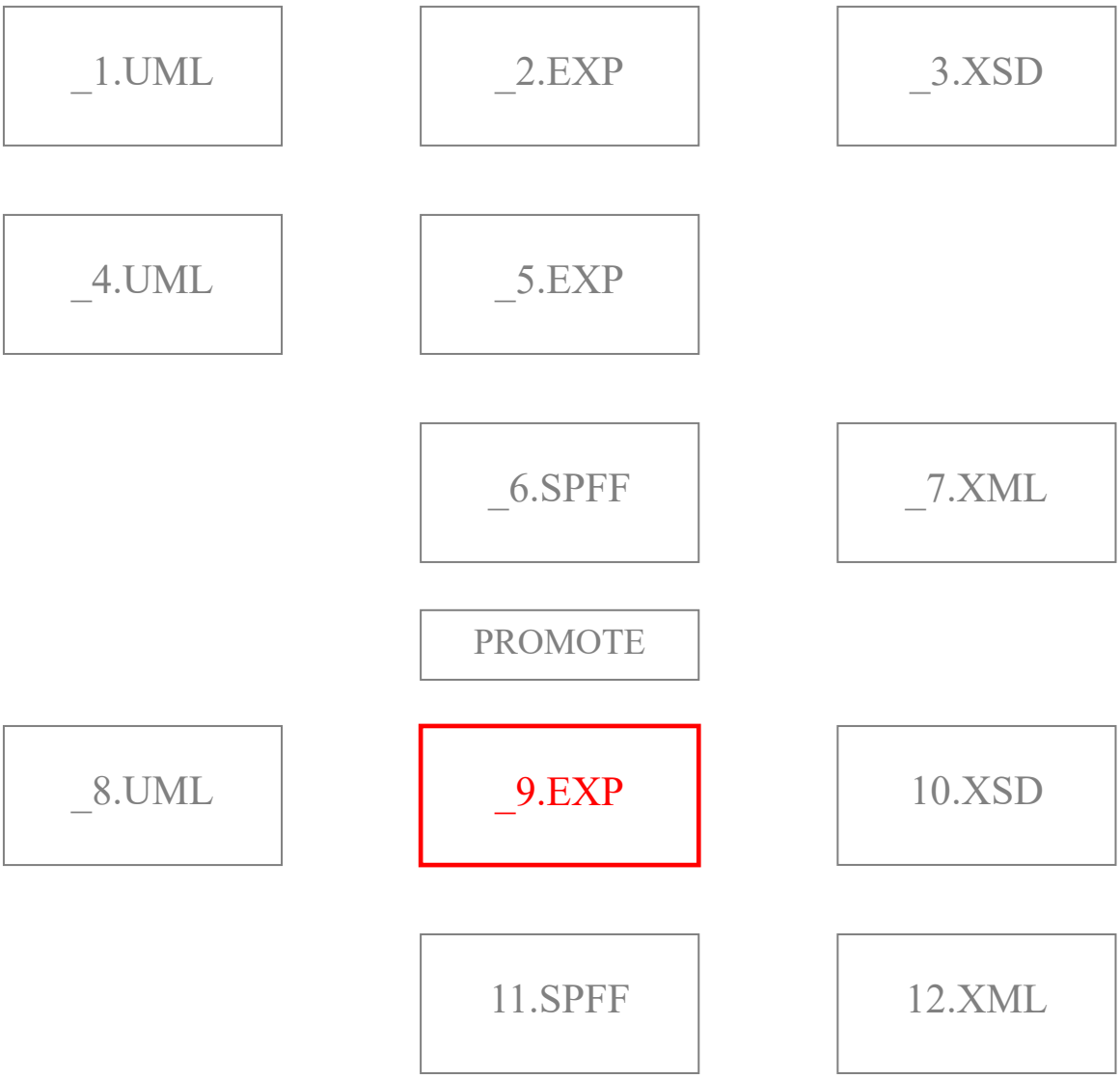
rInfo (OPTIONAL) : returns text feedback on internal process in the DLL.
return value : Error Code, if non zero there is a problem.

Source Code

Starts in dll.cpp
Sets default schema to _2.EXP

int **Convert__9into_10**(*char* ** rInfo);

Converts internally an express schema generated by Promote__2__5__7into__9 API call to XSD internal format.



VISI Structuur

_9.EXP input

Parameters

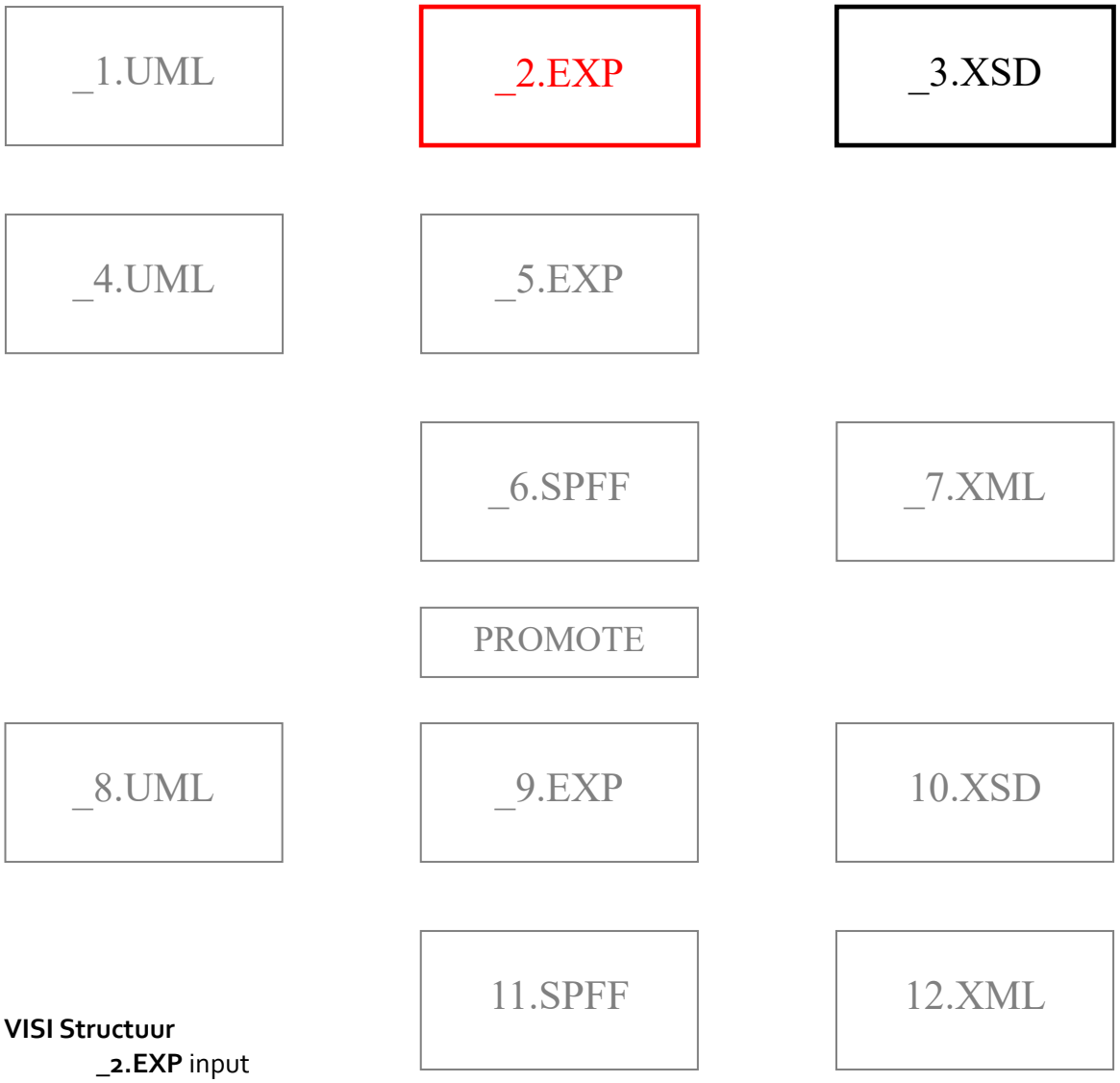
rInfo (OPTIONAL) : returns text feedback on internal process in the DLL
return value : Error Code, if non zero there is a problem

Source Code

Starts in dll.cpp
Sets default schema to _9.EXP

int **Get__3**(*char* ** __3, *char* ** rInfo, *char* * nameSpace, *char* * URI);

Generated XSD that is the schema for interaction frameworks in the DLL.



VISI Structuur

_2.EXP input
_3.XSD output

Parameters

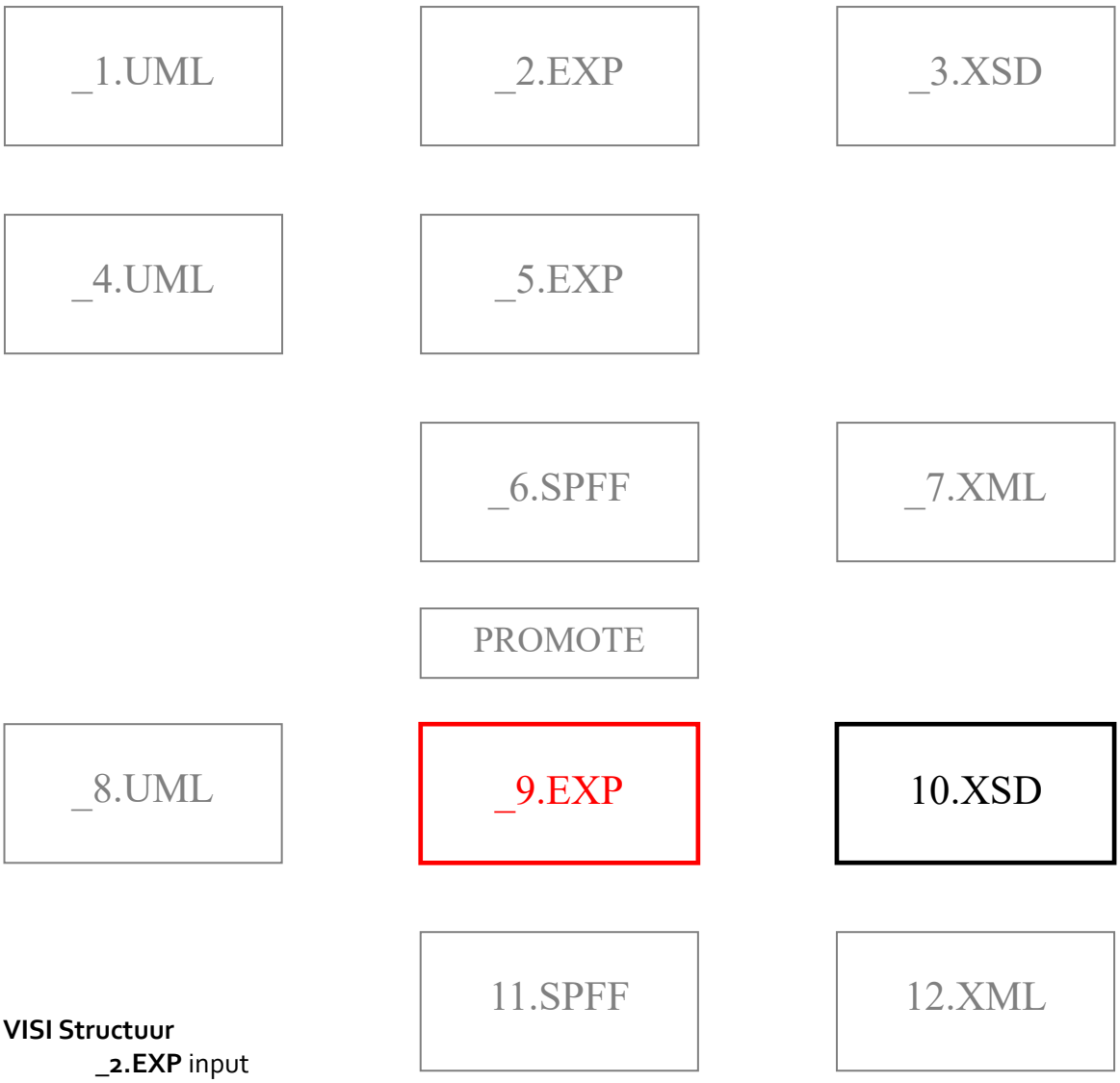
__3 : returns string containing XSD file.
rInfo (OPTIONAL) : returns text feedback on internal process in the DLL
nameSpace : namespace used in the generated XSD
URI : namespace URI used in the generated XSD
return value : Error Code if non zero there is a problem

Source Code

Starts in dll.cpp
Creation of _3.XSD is done in buildXSD.cpp

int **Get_10**(*char* ** _10, *char* ** rInfo, *char* * nameSpace, *char* * URI);

Generated XSD that is the schema for messages in the DLL.



VISI Structuur

_2.EXP input
_3.XSD output

Parameters

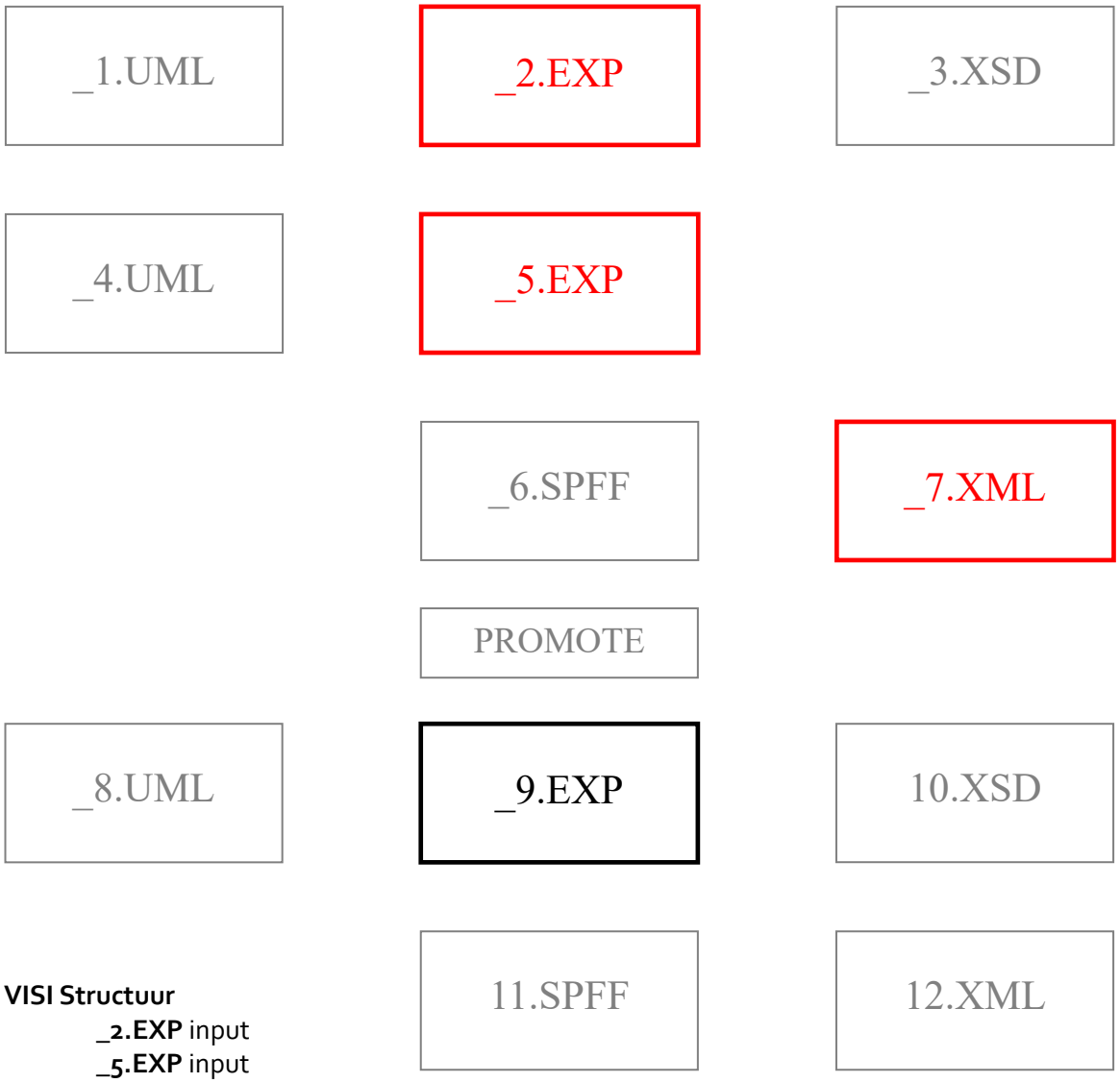
__9 : returns string containing XSD file.
rInfo (OPTIONAL) : returns text feedback on internal process in the DLL
nameSpace : namespace used in the generated XSD
URI : namespace URI used in the generated XSD
return value : Error Code, if non zero there is a problem

Source Code

Starts in dll.cpp
Creation of _9.XSD is done in buildXSD.cpp

int **Promote__2__5__7into__9**(*char* ** rInfo);

Converts internally two express schemas _2 and _5 + the interaction framework _7 into express schema _9 internally.



VISI Structuur

_2.EXP input
_5.EXP input
_7.XML input
_9.EXP output

Parameters

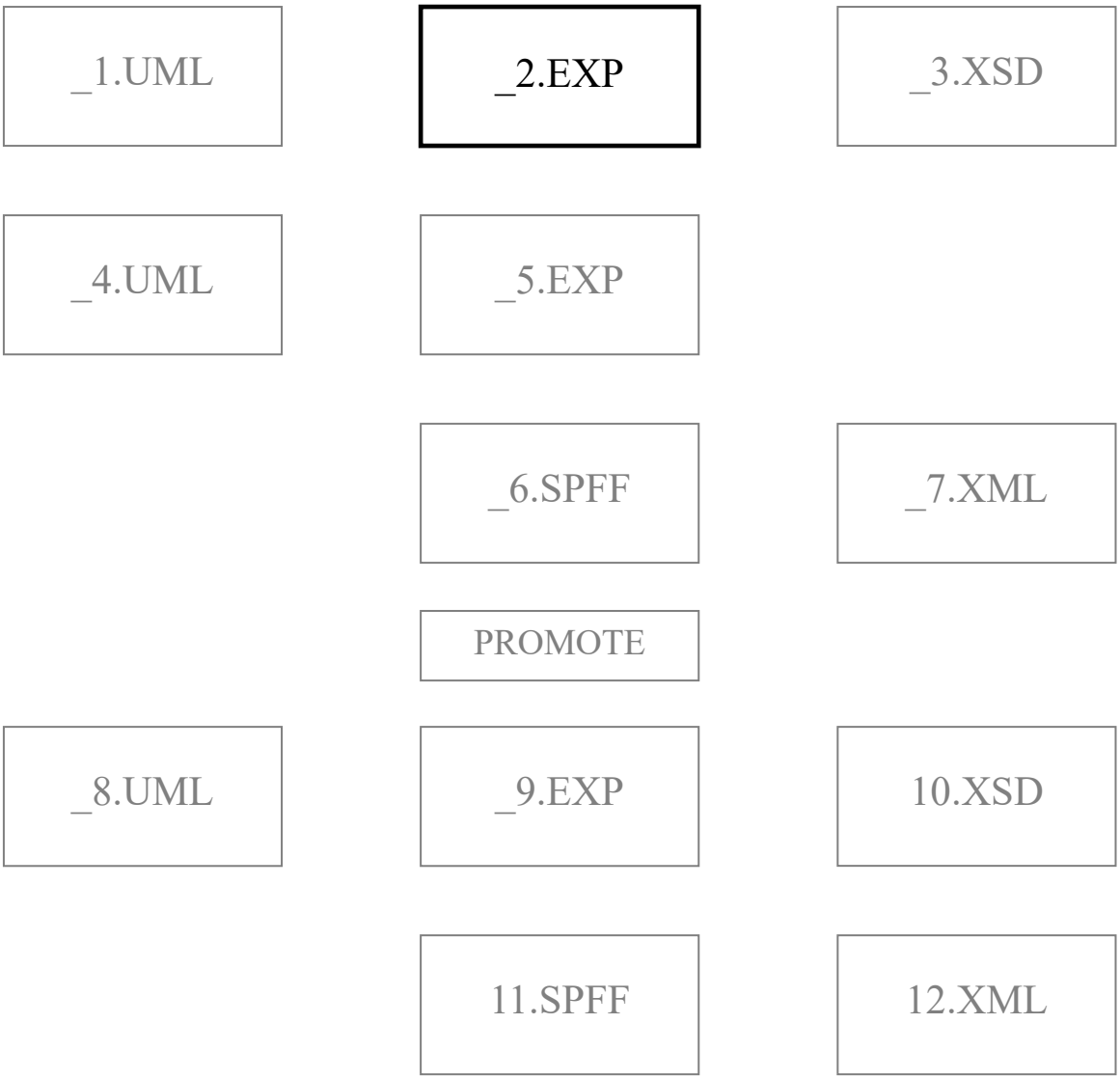
rInfo (OPTIONAL) : returns text feedback on internal process in the DLL
return value : Error Code, if non zero there is a problem

Source Code

Starts in dll.cpp
Promotion is done in promote.cpp

int **Set__2**(*char* * __2, *char* ** rInfo);

Loads an Express Schema (via URL or on local PC).



VISI Structuur

_2.EXP input

Parameters

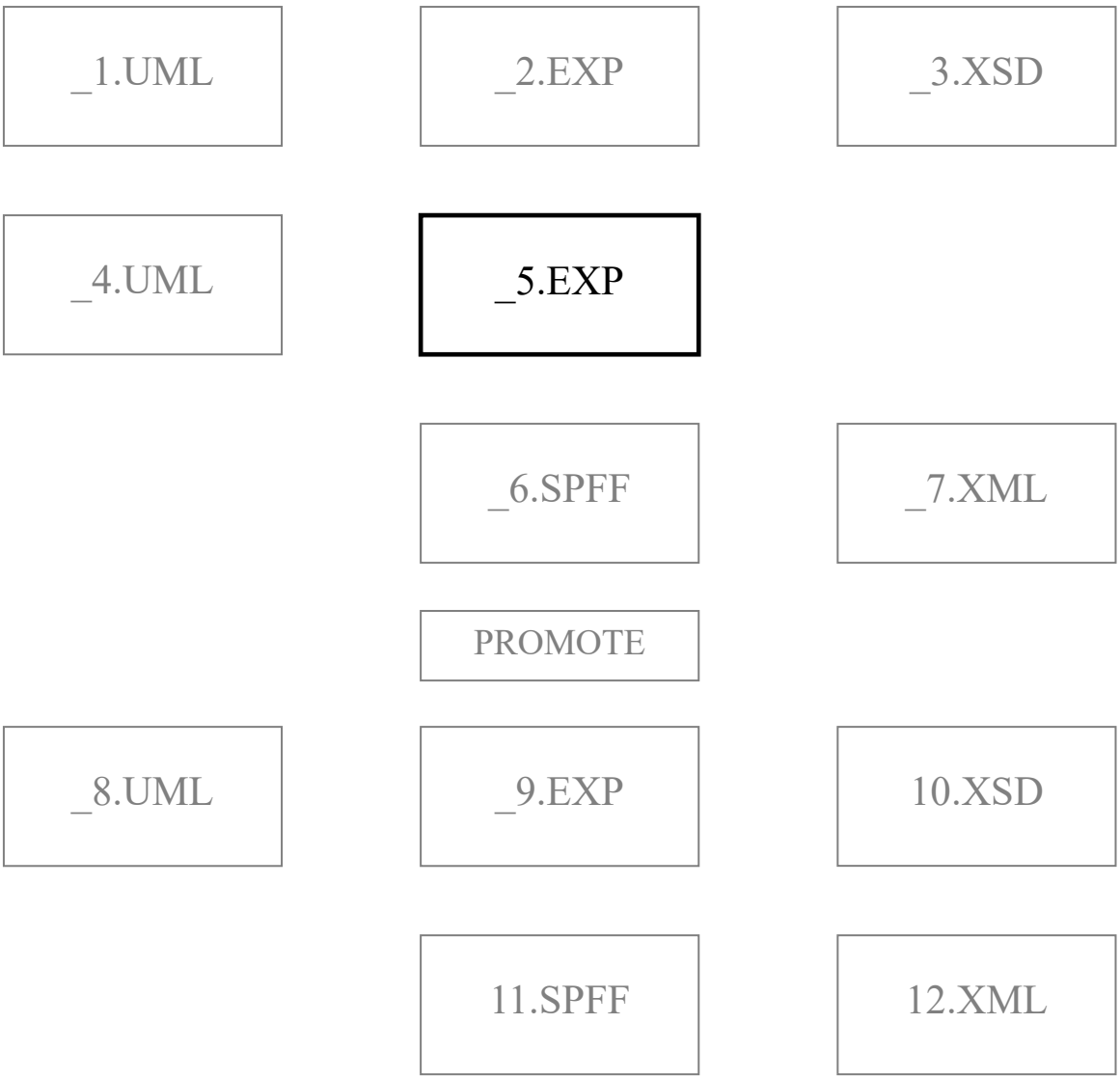
__2 : URL or file location of schema
rInfo (OPTIONAL) : returns text feedback on internal process in the DLL
return value : Error Code, if non zero there is a problem

Source Code

Starts in dll.cpp
Generic input stream is handeled by readExternalSource.cpp
_2.EXP stream is parsed by readSchema.cpp

int **Set__5**(*char* * __5, *char* ** rInfo);

Loads an Express Schema (via URL or on local PC).



VISI Structuur

_5.EXP input

Parameters

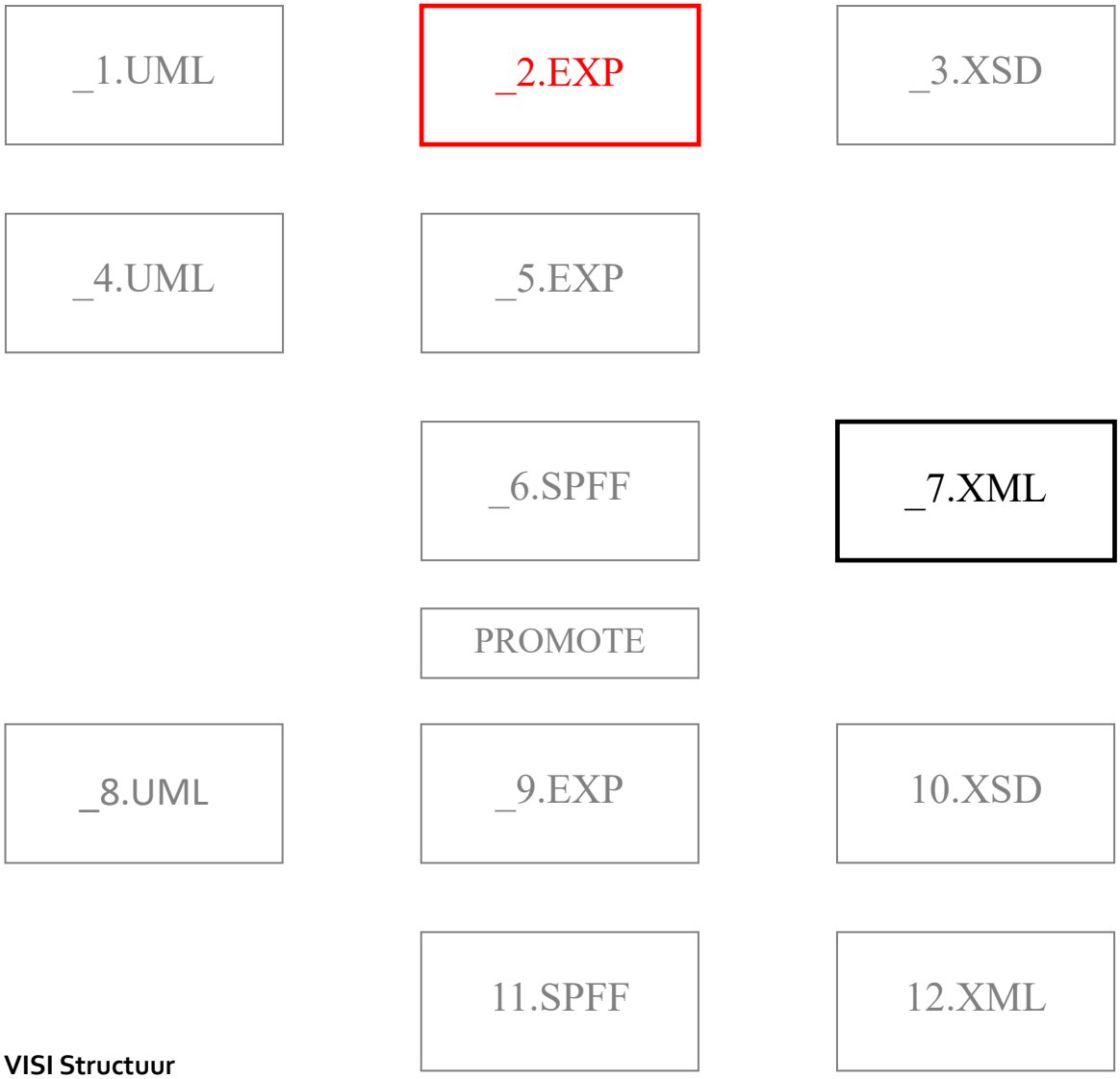
__5 : URL or file location of schema
rInfo (OPTIONAL) : returns text feedback on internal process in the DLL
return value : Error Code if non zero there is a problem

Source Code

Starts in dll.cpp
Generic input stream is handeled by readExternalSource.cpp
_5.EXP stream is parsed by readSchema.cpp

int **Set__7**(*char* * __7, *char* ** rInfo);

Loads an XML interaction framework (via URL or on local PC).



VISI Structuur

_2.EXP input
_7.XML input

Parameters

__7 : URL or file location of schema
rInfo (OPTIONAL) : returns text feedback on internal process in the DLL
return value : Error Code, if non zero there is a problem

Source Code

Starts in dll.cpp
Generic input stream is handeled by readExternalSource.cpp
_7.XML stream is parsed by readXML.cpp

char * **GetErrorCode**(*int* errorCode);

Loads text of an error code.

VISI Structuur

Parameters

errorCode : error code, returned by one of the API calls
return value : text belonging to the Error Code

Source Code

Starts in dll.cpp

< einde Bijlage 9 >