XML DTD Support in DataInterchange

Ragu Satyavolu
IBM

The Next Generation

DI

WDI

2002 User Conference

Agenda

- **XML Implementation in DI 3.1**
  - Overview
  - Steps for translation
  - Demo
  - Benefits
  - Limitations

- **XML Implementation in DI 4.1 / WDI 3.2**
  - Overview
  - Steps for translation
  - Demo
  - Benefits
  - Limitations

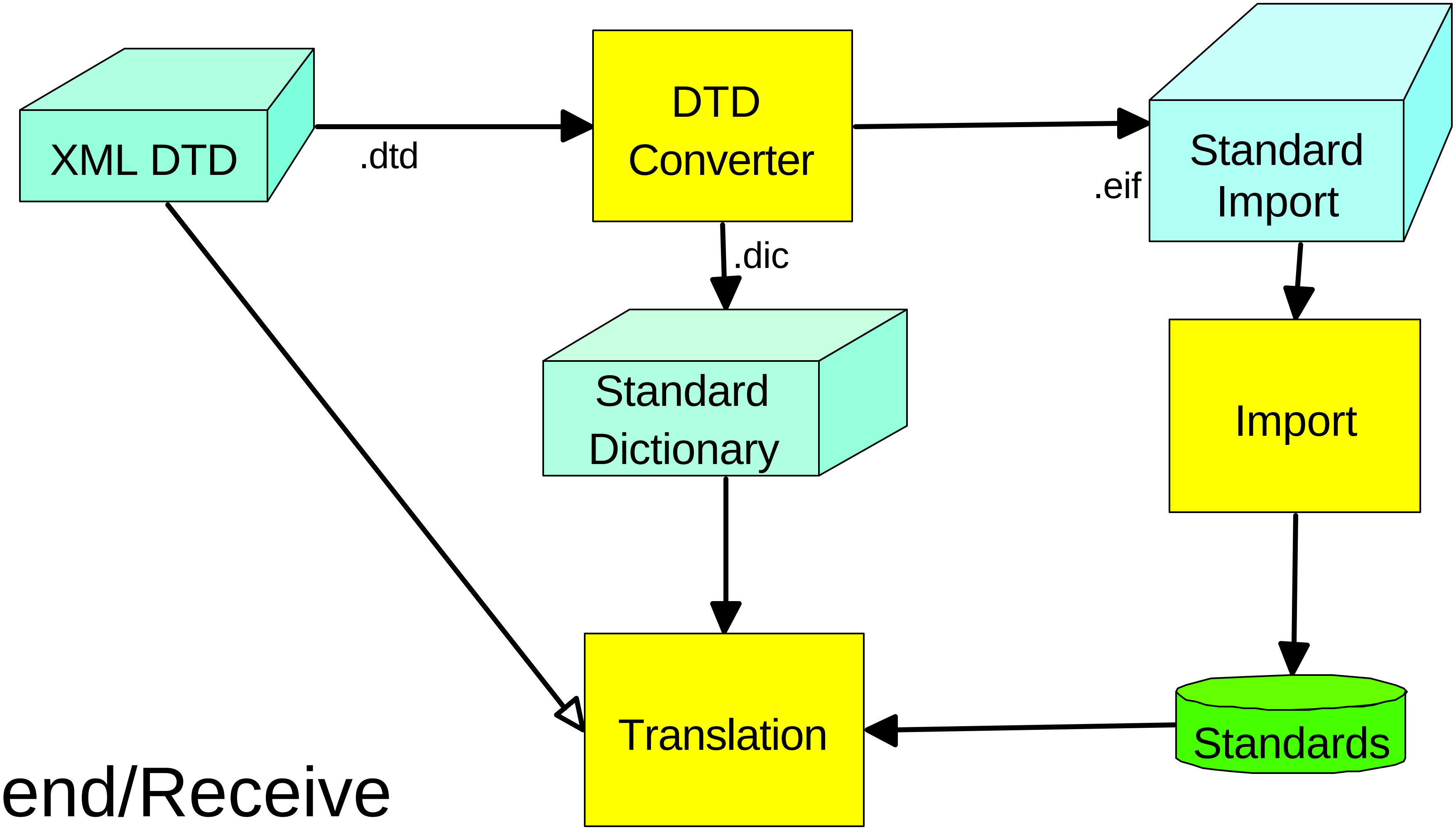
DI 3.1 XML Implementation - Overview

- This implementation allows translation for XML data in both directions ADF->XML (send) and XML->ADF (receive) translation.

- Implemented by converting XML data to a "standard" within DI. The "standard" is a representation of the XML data using an EDI-type syntax that is understood by DI.

- A DTD conversion utility (Windows application) is used to generate a Standard definition file from the user's DTD. The standard has a similar structure as the XML document.

DI 3.1 XML Implementation - Overview



Send/Receive mapping

DI 3.1 XML Implementation - Steps for Translation

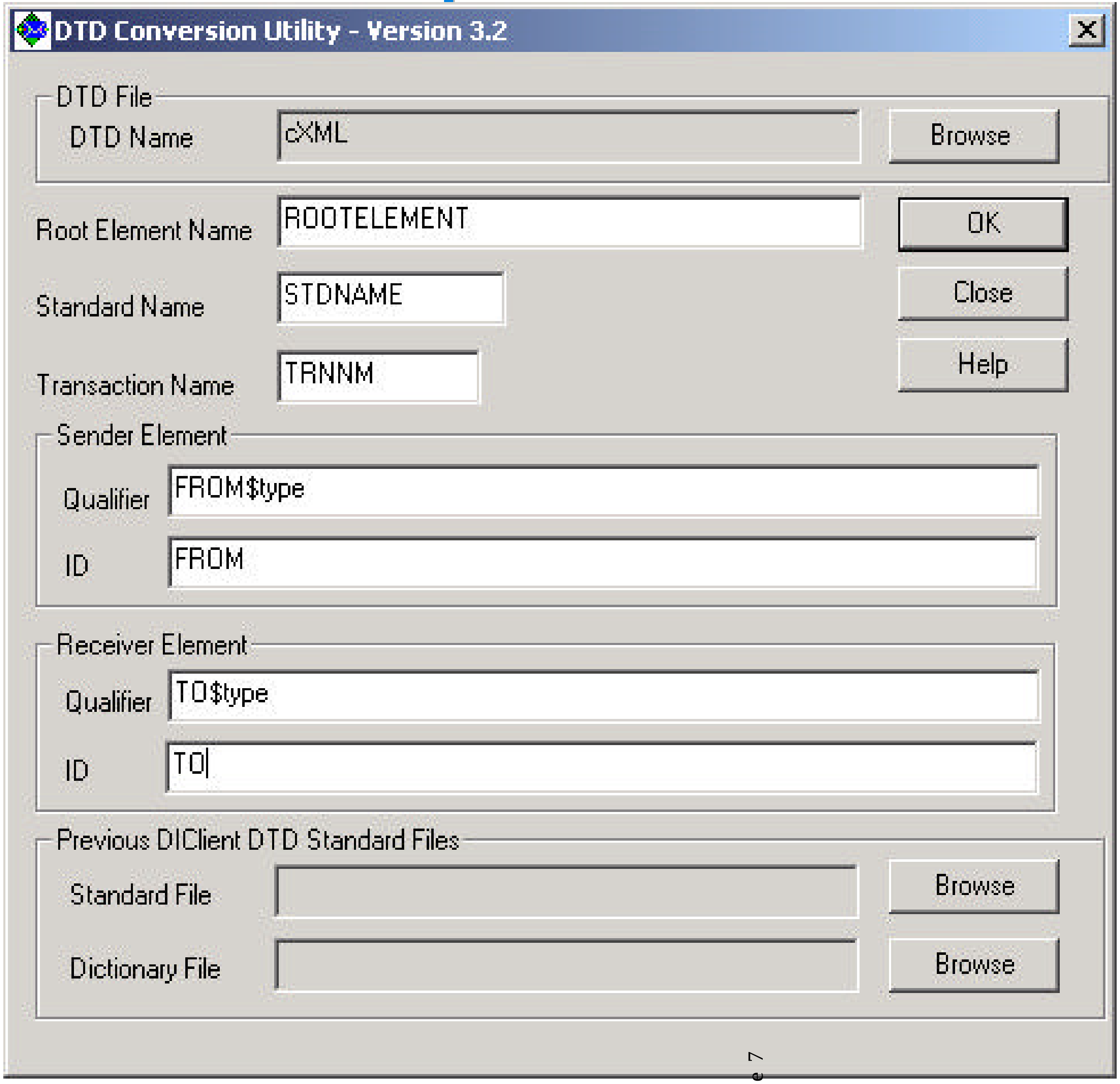
- Convert the DTD using the conversion utility.
  - This creates a standards import file for DI client, and
  - an "XML element dictionary" file to be used by the pre/post processor.
- Import the EI file (generated by the utility) into DI client.
- Save the XML element dictionary file and DTD file for later use of DI during translation.
- Map ADF to the standard (send map), or the standard to ADF (receive map) using DI client
- Generate Control string for mapping and setup trading partner information.
- Invoke DI Utility to do the translation and specify some new keywords on the PERFORM command(s).

DI 3.1 XML Implementation - Steps for Translation *(continued)*

- When doing send (ADF->XML) processing, DI will map the data to the "standard" format using the send map, then invoke a post processor to convert from the standard format to XML.

- When doing receive (XML->ADF) processing, DI will invoke a preprocessor to convert the XML data to the "standard" format, then map the standard to the user's ADF format using the receive map.

DI 3.1 XML Implementation - DTDConvertion Utility



DTD Conversion Utility - Version 3.2

DTD File

DTD Name: cXML

Root Element Name: ROOTELEMENT

Standard Name: STDNAME

Transaction Name: TRNNM

Sender Element

Qualifier: FROM$type

ID: FROM

Receiver Element

Qualifier: TO$type

ID: TO

Previous DIClient DTD Standard Files

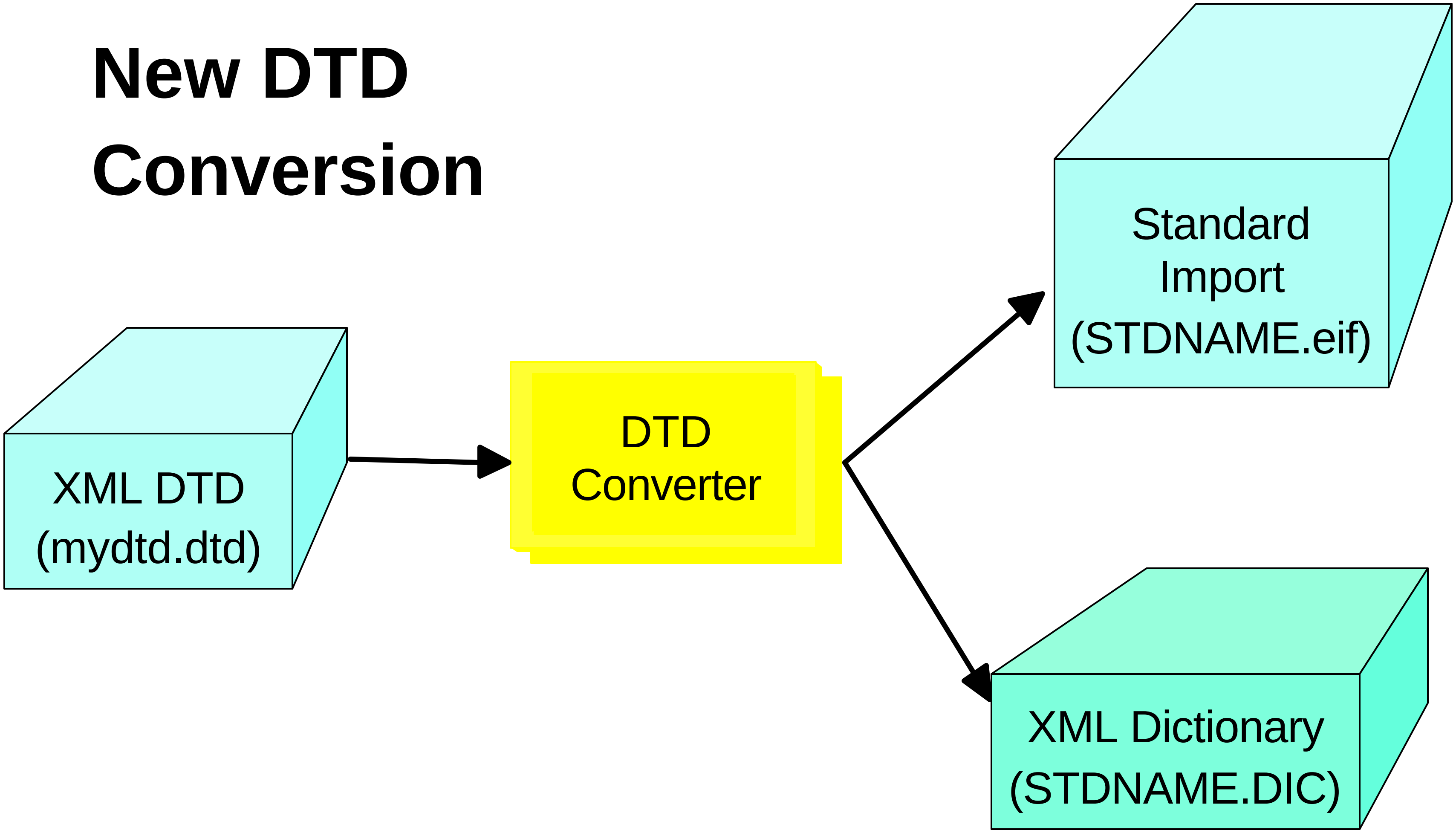
Standard File:

Dictionary File:

New DTD Conversion

DI 3.1 XML Implementation - DTDConvertion Utility

New DTD Conversion



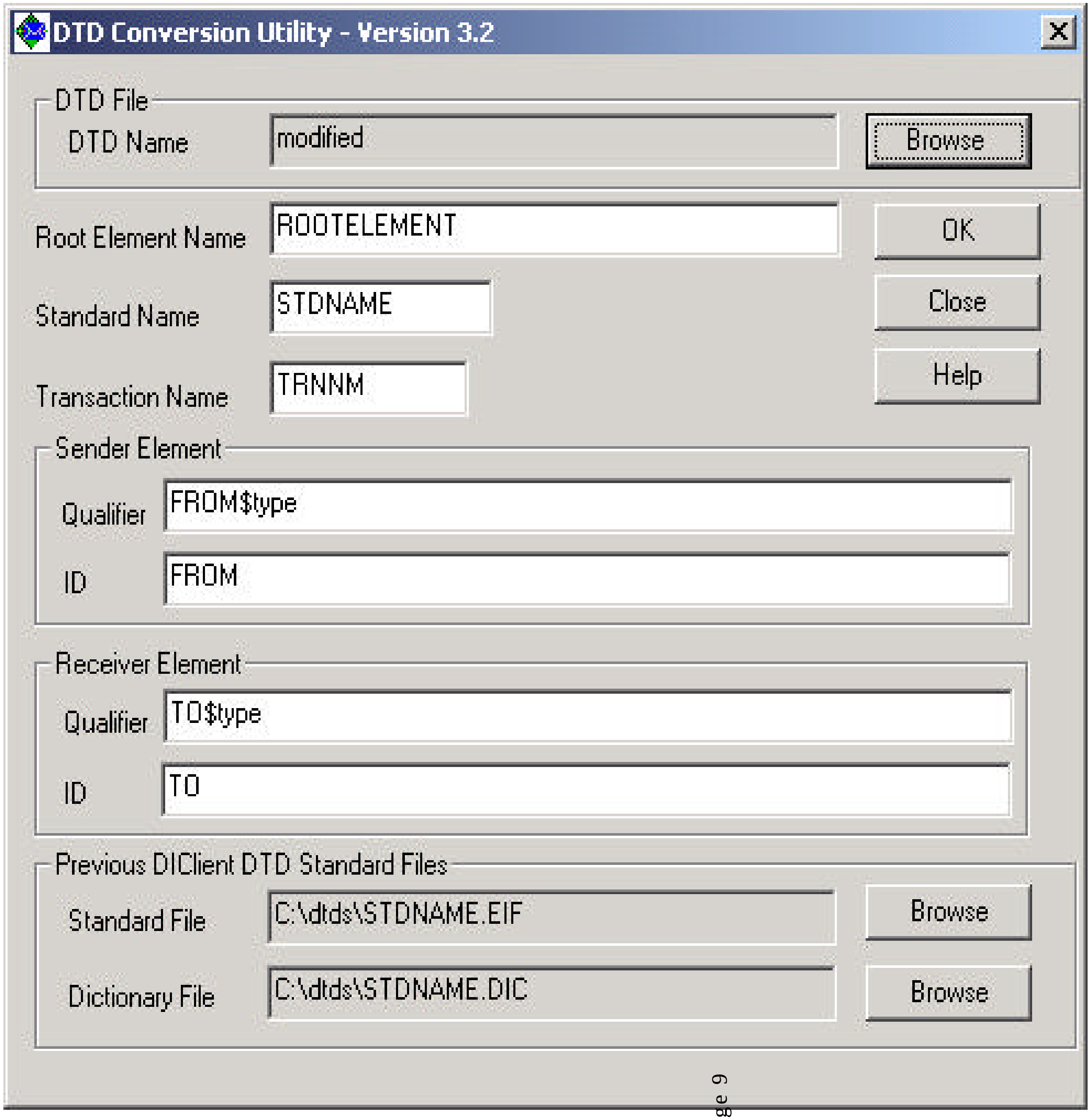
XML DTD (mydtd.dtd)

DTD Converter

Standard Import (STDNAME.eif)

XML Dictionary (STDNAME.DIC)

DI 3.1 XML Implementation - DTDConvertion Utility



DTD Conversion Utility - Version 3.2

DTD File

DTD Name: modified

Root Element Name: ROOTELEMENT

Standard Name: STDNAME

Transaction Name: TRNNM

Sender Element

Qualifier: FROM$type

ID: FROM

Receiver Element

Qualifier: TO$type

ID: TO

Previous DIClient DTD Standard Files

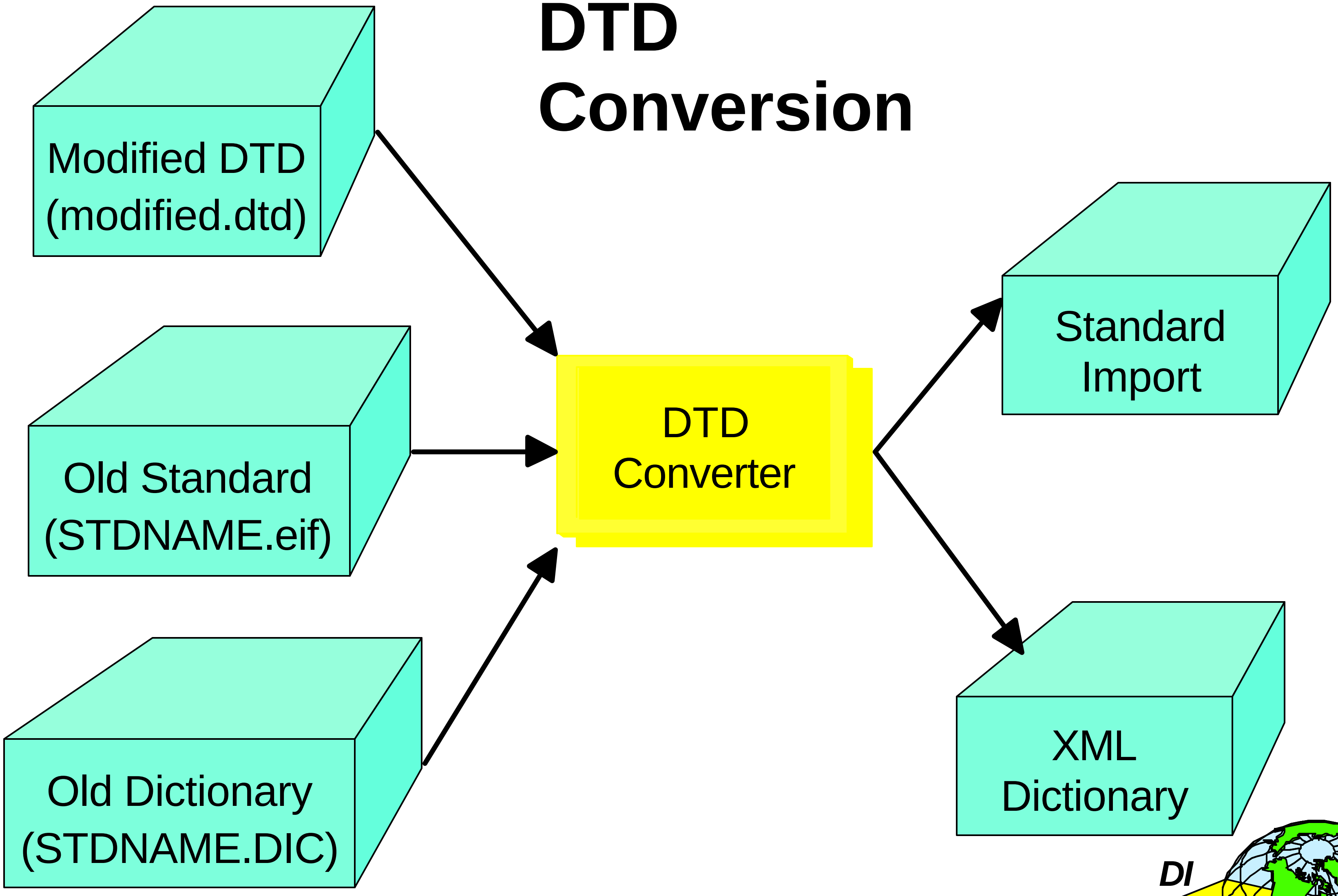
Standard File: C:\dtds\STDNAME.EIF

Dictionary File: C:\dtds\STDNAME.DIC

Modified DTD Conversion

DI 3.1 XML Implementation - DTDConvertion Utility

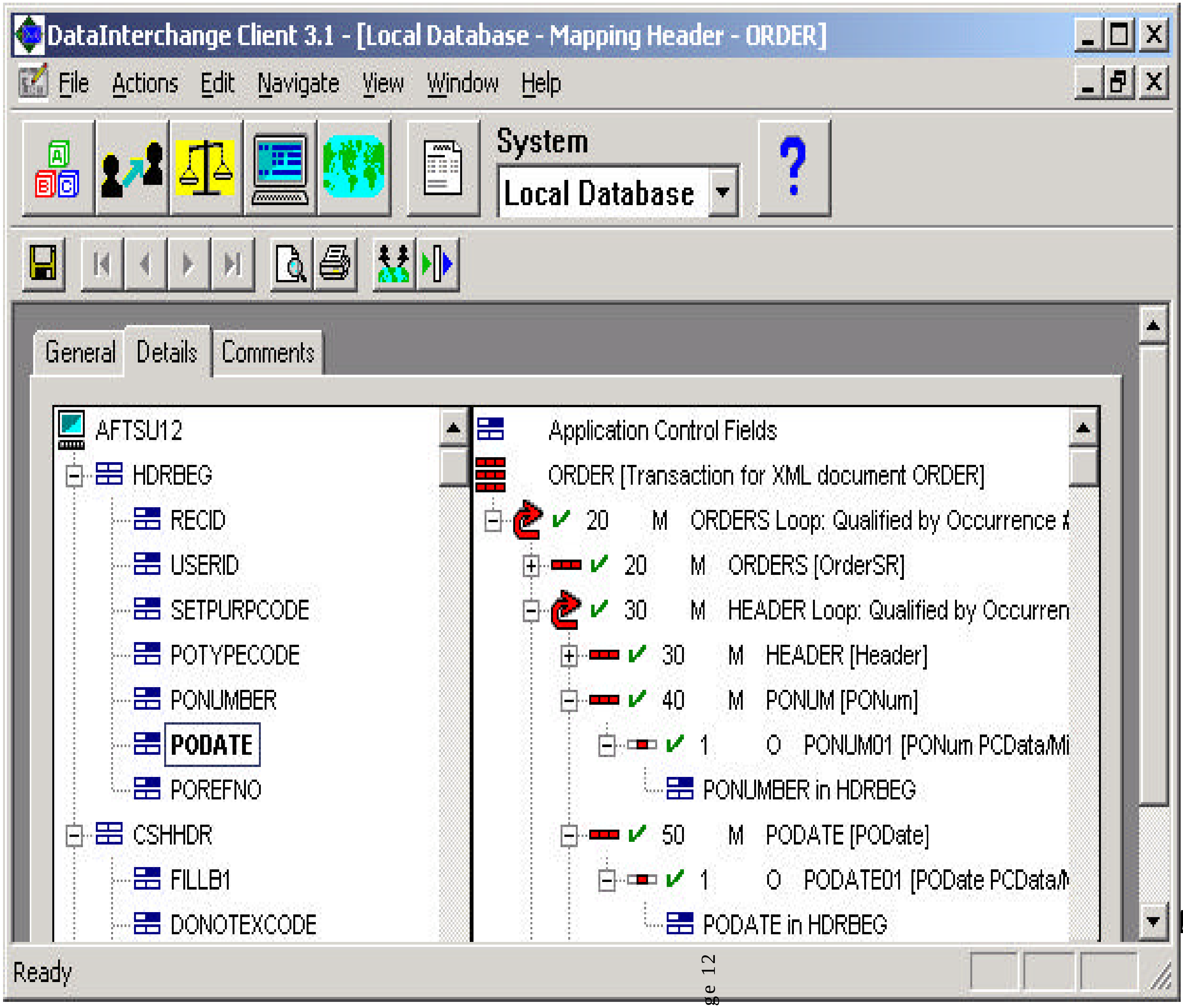
Modified DTD Conversion


Modified DTD (modified.dtd)
Old Standard (STDNAME.eif)
Old Dictionary (STDNAME.DIC)
DTD Converter
Standard Import
XML Dictionary

DI 3.1 XML Implementation - Sample DTD File

```
<!ELEMENT OrderSR (Header, DetailLoop*, Trailer)>
<!ELEMENT Header (PONum, PODate, Sender, Receiver)>
<!ATTLIST Header typecode CDATA #REQUIRED>
<!ELEMENT PONum (#PCDATA)>
<!ELEMENT PODate (#PCDATA)>
<!ELEMENT Sender (Id, Qualifier)>
<!ELEMENT Receiver (Id, Qualifier)>
<!ELEMENT Id (#PCDATA)>
<!ELEMENT Qualifier (#PCDATA)>
<!ELEMENT DetailLoop (ItemNumber, SubDetail+)>
<!ELEMENT ItemNumber (#PCDATA)>
<!ELEMENT SubDetail (Description, Quantity, UnitPrice)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT UnitPrice (#PCDATA)>
<!ELEMENT Trailer (ItemCount, TotalBucks)>
<!ELEMENT ItemCount (#PCDATA)>
<!ELEMENT TotalBucks (#PCDATA)>
```

DI 3.1 XML Implementation - Mapping editor



AFTSU12
- HDRBEG
  - RECID
  - USERID
  - SETPURPCODE
  - POTYPECODE
  - PONUMBER
  - **PODATE**
  - POREFNO
- CSHHDR
  - FILLB1
  - DONOTEXCODE

```
Application Control Fields
ORDER [Transaction for XML document ORDER]
 ✓ 20   M  ORDERS Loop: Qualified by Occurrence #
   ✓ 20   M  ORDERS [OrderSR]
   ✓ 30   M  HEADER Loop: Qualified by Occurren
     ✓ 30   M  HEADER [Header]
     ✓ 40   M  PONUM [PONum]
       ✓ 1    O  PONUM01 [PONum PCData/Mi
            PONUMBER in HDRBEG
     ✓ 50   M  PODATE [PODate]
       ✓ 1    O  PODATE01 [PODate PCData/M
            PODATE in HDRBEG
```

DI 3.1 XML Implementation - Demo

Demonstration

DI 3.1 XML Implementation - Benefits

- Provides an immediate solution to the industry demand for XML support.

DI 3.1 XML Implementation - Limitations

- Requires DTD to be converted to standard.
- Extra work to synchronize "dictionary" with standard.
- Requires double translation for XML to/from EDI or XML to XML.
- Transaction Store shows XML as EDIFACT data

DI 4.1/WDI 3.2 XML Implementation

Note

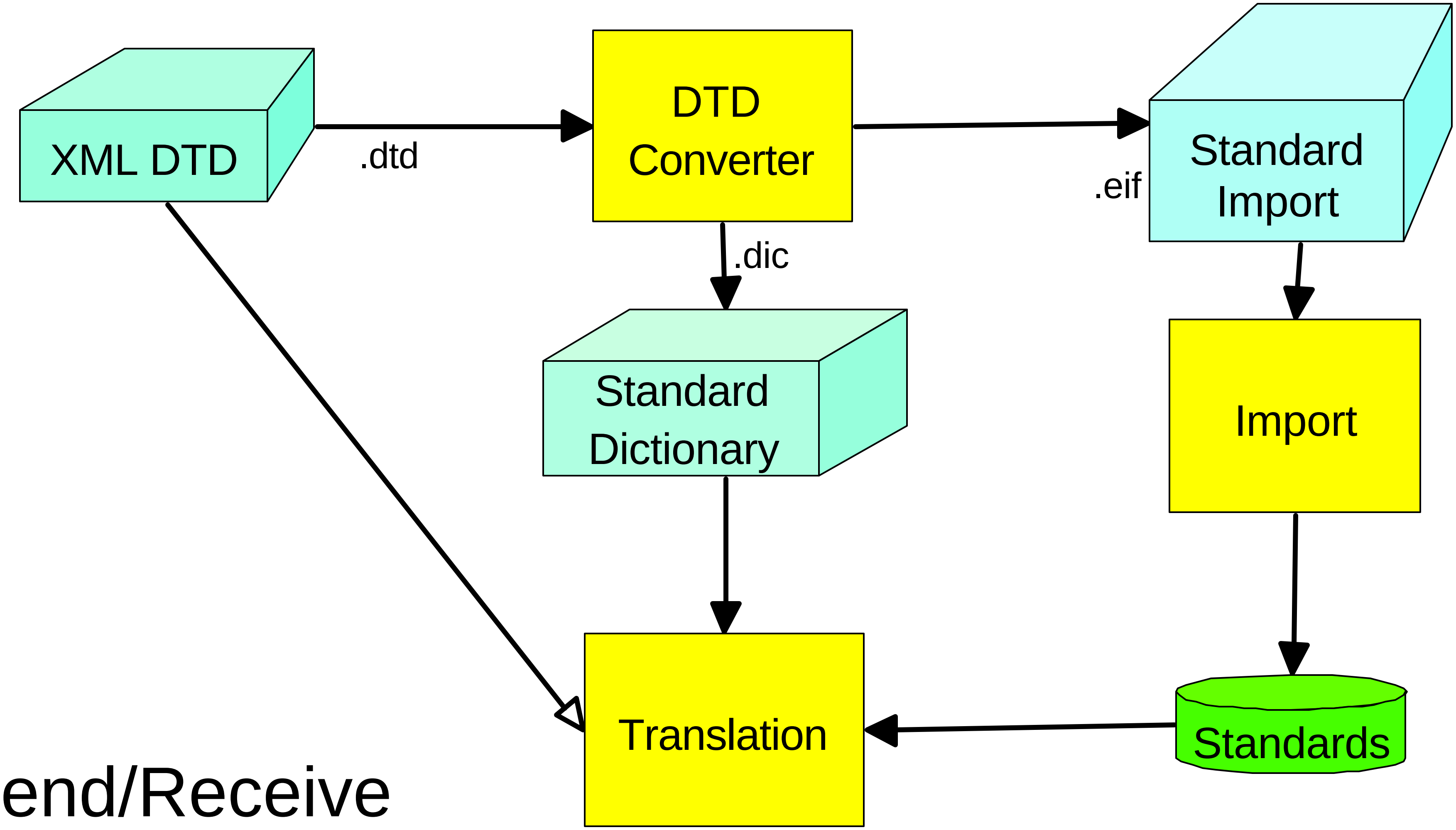
- A new way of XML translation is added in DI4.1/WDI3.2 along with the old DI3.1 implementation.

- XML processing is similar in both DI 4.1 and WDI 3.2 releases. For our demonstration purpose we will use WDI 3.2.

DI 4.1/WDI 3.2 XML Implementation - Overview

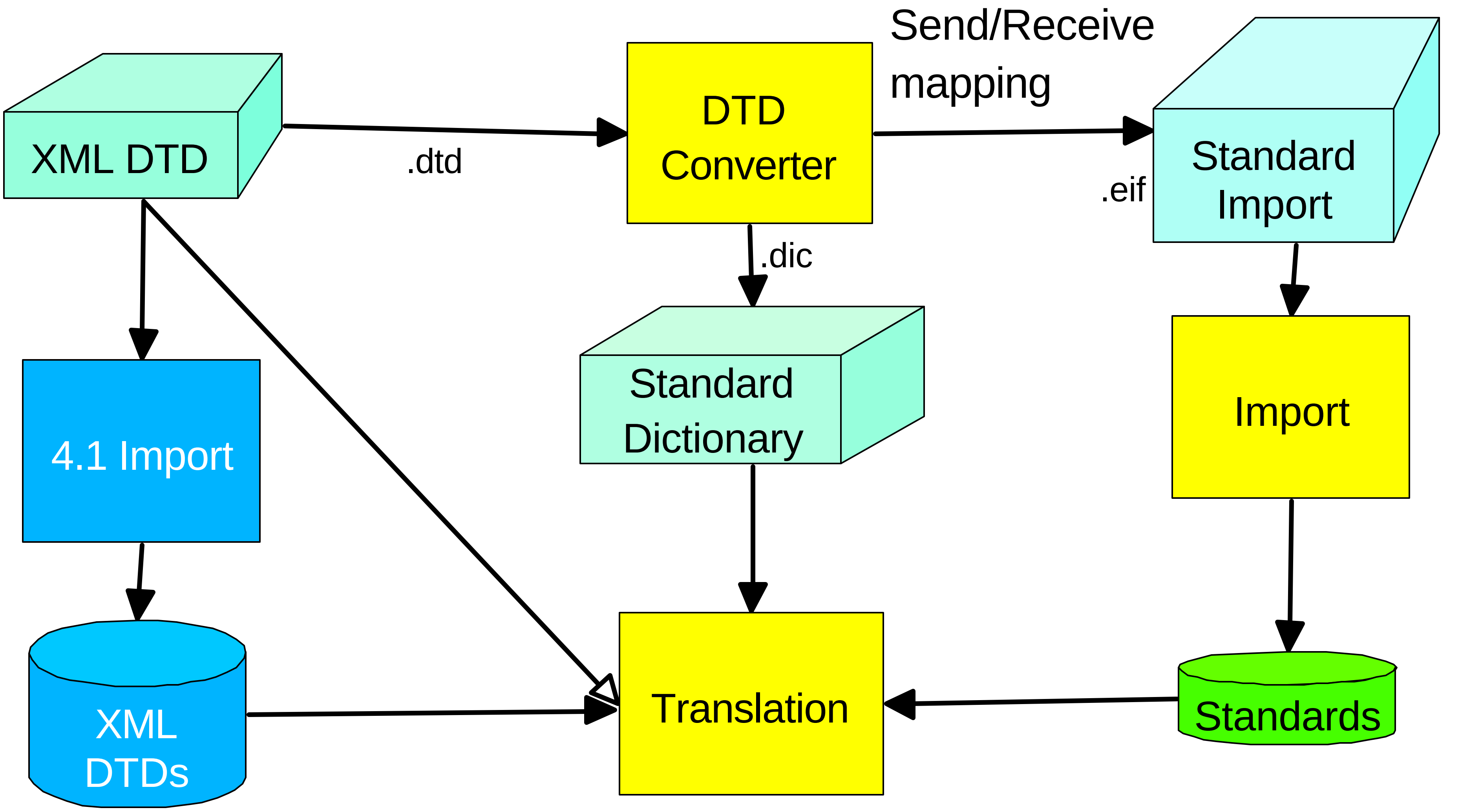
- The Industry standard or user defined XML DTDs can be imported directly for mapping and translation.
- This implementation allows translation of XML to any of the supported target formats or from any of the supported source formats to XML.
- Uses the new feature of "any-to-any" data transformation maps which allow translation between any supported source document format (ADF, EDI or XML) to any supported target format (ADF,EDI or XML).

DI 3.1 XML Implementation - Overview



Send/Receive mapping

DI 3.1 XML Implementation - Overview



Send/Receive mapping

XML DTD

.dtd

DTD Converter

.eif

Standard Import

.dic

Standard Dictionary

Import

4.1 Import

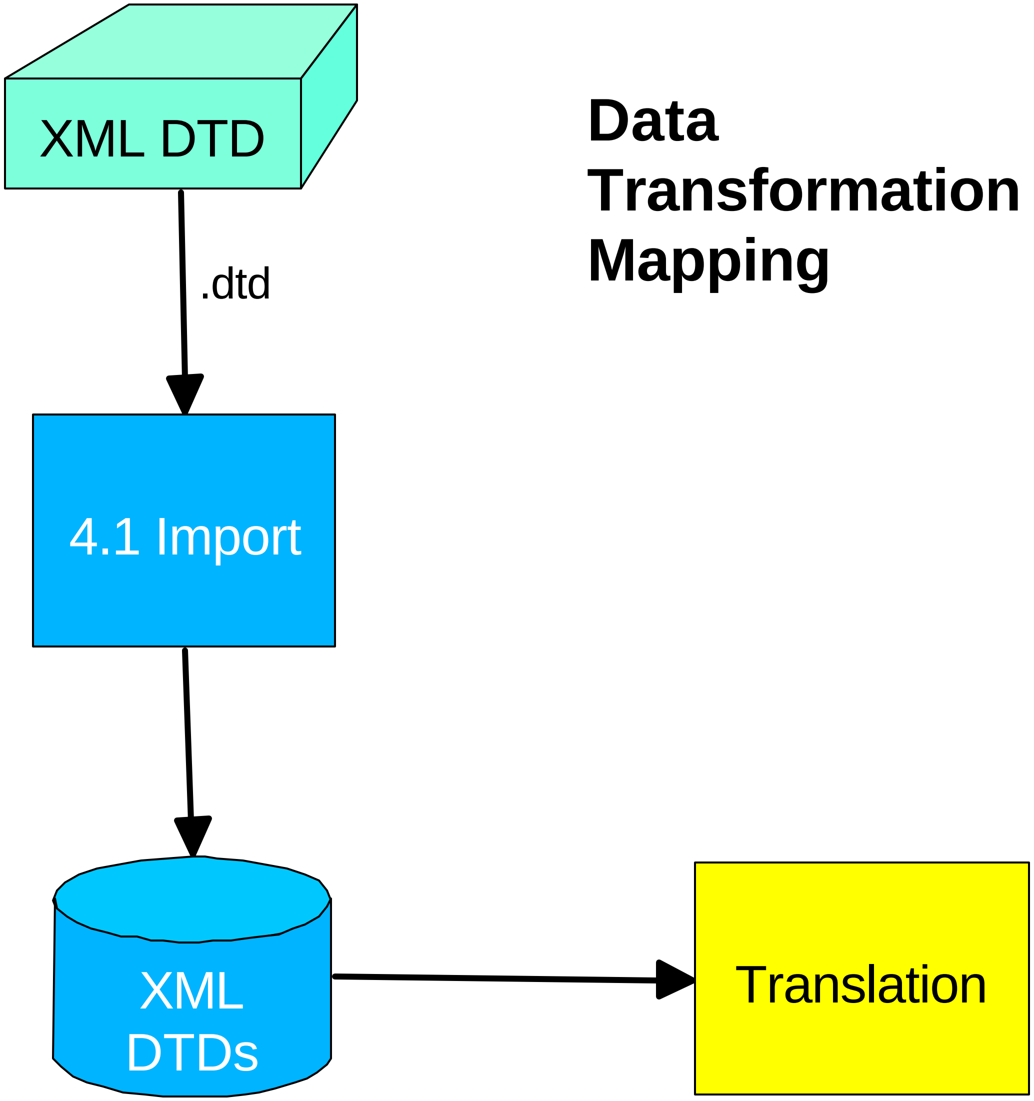
XML DTDs

Translation

Standards

Data transformation mapping

DI 4.1/WDI 3.2 XML Implementation - Overview

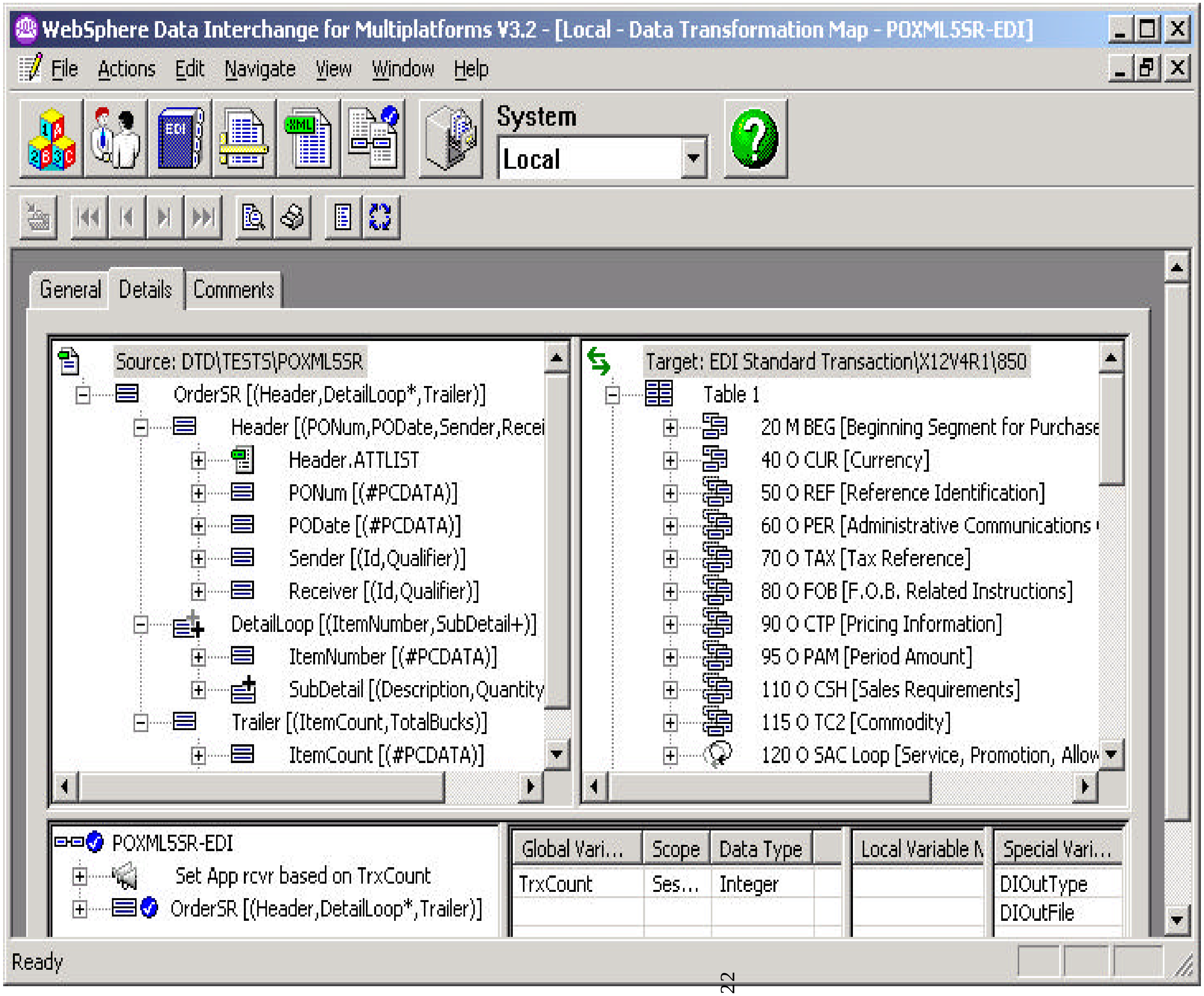


Data Transformation Mapping

DI 4.1/WDI 3.2 XML Implementation - Steps for Translation

- Import the DTD file into DI Client. The DTD name must be unique within a dictionary which is used to group together related DTDs.
- Create a data transformation map to map DTD to target document format or source document format to DTD.
- Compile this map and generate a control string.
- Invoke DI utility to run the translation.
- DI will convert the XML to target document or source document to XML depending on the map.

DI 4.1/WDI 3.2 XML Implementation - DTD Mapping Editor


WebSphere Data Interchange for Multiplatforms V3.2 - [Local - Data Transformation Map - POXML5SR-EDI]
File Actions Edit Navigate View Window Help
System
General Details Comments
Ready

Local

Source: DTD\TESTS\POXML5SR

```
OrderSR [(Header,DetailLoop*,Trailer)]
    Header [(PONum,PODate,Sender,Recei
        Header.ATTLIST
        PONum [(#PCDATA)]
        PODate [(#PCDATA)]
        Sender [(Id,Qualifier)]
        Receiver [(Id,Qualifier)]
    DetailLoop [(ItemNumber,SubDetail+)]
        ItemNumber [(#PCDATA)]
        SubDetail [(Description,Quantity
    Trailer [(ItemCount,TotalBucks)]
        ItemCount [(#PCDATA)]
```

Target: EDI Standard Transaction\X12V4R1\850

```
Table 1
    20 M BEG [Beginning Segment for Purchase
    40 O CUR [Currency]
    50 O REF [Reference Identification]
    60 O PER [Administrative Communications (
    70 O TAX [Tax Reference]
    80 O FOB [F.O.B. Related Instructions]
    90 O CTP [Pricing Information]
    95 O PAM [Period Amount]
    110 O CSH [Sales Requirements]
    115 O TC2 [Commodity]
    120 O SAC Loop [Service, Promotion, Allow
```

```
POXML5SR-EDI
    Set App rcvr based on TrxCount
    OrderSR [(Header,DetailLoop*,Trailer)]
```

Global Vari...	Scope	Data Type
TrxCount	Ses...	Integer

Local Variable N

Special Vari...
DIOutType
DIOutFile

DI 4.1/WDI 3.2 XML Implementation - Demo

Demonstration

DI 4.1/WDI 3.2 XML Implementation - Benefits

- Direct import of DTD and hence no need to use a separate tool to convert DTD to standard.
- No need to keep track of separate dictionary file.
- Better representation of DTD in map editor.
- No need for double translation when translating between XML->EDI, EDI->XML or XML->XML.

DI 4.1/WDI 3.2 XML Implementation - Limitations

- Reminder -
  TRANSFORM has no Interface with Transaction Store

- Direct Import of DTD only used for Data Transformation Maps