

EasyWay



Annual Forum 2010

Shortcut to the future.
Lisbon • November 16th-18th



EasyWay



Annual Forum 2010

Shortcut to the future.
Lisbon • November 16th-18th

How to create a
DATEX II service?
Josef Kaltwasser



Outline

Basic Principles of setting up DATEX II services

Selecting a data model for publication

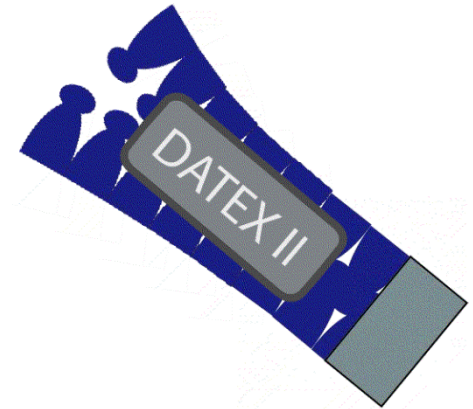
Need to extend?

How to create extensions

Exchange specifications

Frequently Asked Questions

Conclusion





Basic Principles

Old DATEX had an *Interchange Agreement* – DATEX II needs something similar: a **Service Description**

A data source must **describe** the data provided via this source

This description contains elements that describe content (e.g. Geographic coverage), but also structure: which (optional!) data model elements are actually used by the service

DATEX II does not make a distinction between **optional** and **conditional** – it is the user's service description that drops all unused (optional) elements from the model and makes all remaining optional elements conditional

Users that need data concepts not – yet – contained in the DATEX II model can **extend** it (levels B & C); this again requires description for clients

Beyond the data definition, the technical access procedures must be specified in terms of **protocol choices** (HTTP vs. SOAP, push vs. pull...) as well as **configuration data** (endpoint URL, access credentials...)



Data Element selection

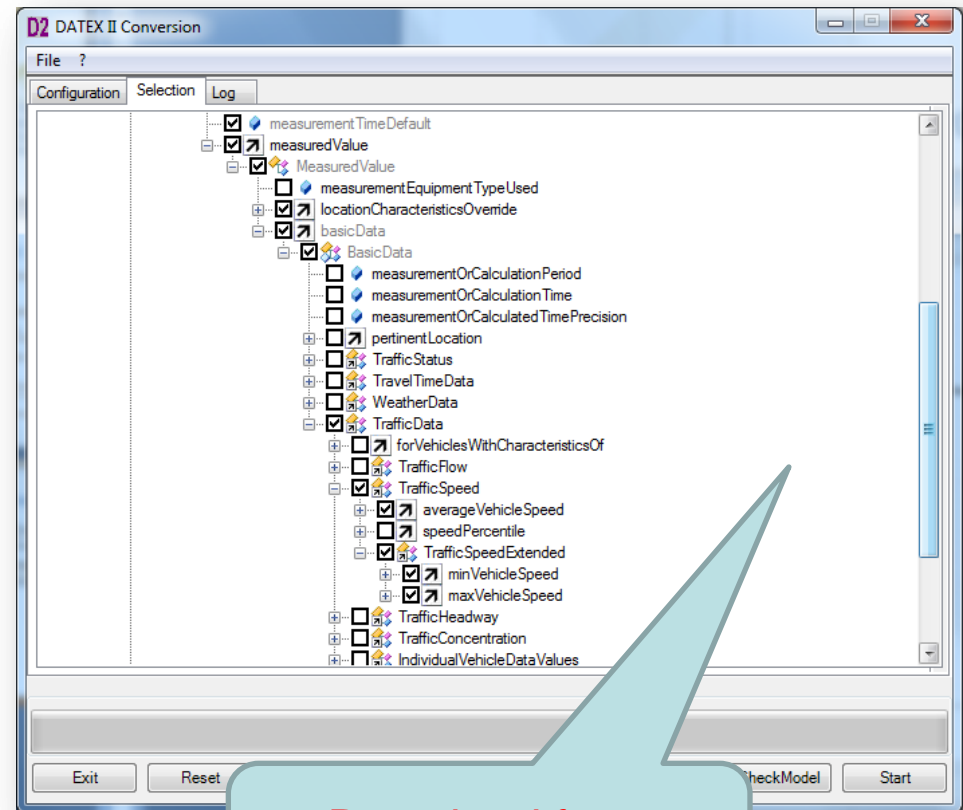
Data model selection is facilitated by the **DATEX II tool**

The tool depicts the model as a **tree structure**, where optional branches can be de-/selected

The tool unfolds composition / aggregation **relationships** as well as **inheritance**

Users **describe their data** structures of their particular interface by deselecting non-used elements

Finally, users can click to produce their **tailored schema!**



Download from
datex2.eu



Need to extend?

Many users find the existing ("Level A") data model "95+% fit for purpose", but require some minor extra **refinements / additions**

DATEX II covers these requirements by allowing for **extensions**

In case only minor additions to the model are needed ("Level B"), the resulting XML messages remain **valid instances** of the original schema! (i.e. the extended systems remain **backwards compatible**)

Only when the Level B rules can not be fulfilled, the model has to leave the DATEX II namespace and systems are no longer interoperable ("**Level C**")

Level B extensions "**do not hurt**" – i.e. they do not jeopardise interoperability – when used sensibly. In particular, the **non-extended content** shall still be comprehensible and useful

Level B extensions provide a powerful tool to **strike a balance** between standardised interfaces & interoperability vs. flexibility



The screenshot displays the DATEX II Conversion software interface. The main window, titled "ErweiterungenKommunlaesVerkehrsmanagement - EA", shows a logical diagram of a class hierarchy. The diagram includes a base class "TrafficSpeed" (part of "TrafficData") and an extension class "TrafficSpeedExtended" (marked as «class»). The extension class has two attributes: "+minVehicleSpeed 0..1" and "+maxVehicleSpeed 0..1". Below the extension class is a "DataValue::SpeedValue" class with a "+ speed: KilometresPerHour" attribute. The diagram also shows a "DataValue" class with a "speed" attribute. The "Project Browser" on the right shows the project structure, including "D2LogicalModel", "Extension", and "«class» TrafficSpeedExtended". The "Tagged Values" panel for "TrafficSpeedExtended (Class)" shows the following values:

Property	Value
changed	yes
definition	Extension class that...
extension	levelb
origin	pull...

The status bar at the bottom indicates the current class is "TrafficSpeedExtended" with dimensions: Left: 186 x Top: 118 - Width: 246 x Height: 70.



Exchange specifications

Once the DATEX II data model for a service has been created and the corresponding schema has been generated, the corresponding **interface choices** have to be made

The current DATEX II exchange model allows for two disparate exchange technologies

- **Web Services**: using the SOAP protocol to implement **RPC**-type (*Remote Procedure Call*) exchange – both, push and pull
- Pure **HTTP**: using HTTP directly to implement **RESTful** pull services

Both technologies require additional decisions to be taken regarding protocol **parameters and choices** (e.g. whether compressions should be used, the URL of the endpoints, filtering specifications inside the URL, etc.)

All these choices and parameters need to be collected and put together in an **interface description document** (including the WSDL file when using Web Services)



A brief DATEX II FAQ

Which exchange technology should I use?

Use plain HTTP for pull services if running a Web Service is difficult for you.
Use WS for push services and also for pull service if you can and your clients prefer writing a SOAP client over handling HTTP directly.

Should I use the Push or the Pull pattern?

Use push for services sensitive to latency, but keep in mind that your clients have to be able to operate a web service!
Use pull for low client implementation complexity.

Should I use compression?

DATEX II (XML) has low signal entropy – use compression if bandwidth is an issue

Should I generate a dedicated schema for my service?

YES! Fully describe your particular service – including your data model. The “full schema” is just for reference. Consider client complexity!



Conclusion

DATEX II has a sound methodology and tools to support users in creating B2B data services

Nevertheless, it cannot take away the effort required to describe an interface for potential clients

Service descriptions must describe the data model as well as the protocol choices

Describing the data model of a service means:

- deciding and selecting optional elements from the DATEX II full mode
- creating required extensions – if needed
- generating the corresponding subschema using the tool

Exchange choices need to be documented, supported features described and concrete parameter values (e.g. Internet addresses) provided