RSW GEOMATICS LTD

1Spatial

Rob Walker Consultancy

Technical Guidance for the INSPIRE Schema Transformation Network Service

EC JRC Contract Notice 2009/S 107-153973

Authors: Mark Howard, Simon Payne, Richard Sunderland

Date: 12 July 2010
Version: 3.0
Status: Final

Document Information

This is the Technical Guidance for the INSPIRE Schema Transformation Network Service.

Purpose

This technical guidance document provides a concrete interface specification and supporting documentation for INSPIRE Transformation Network Services (TNS). This will enable interoperability by alleviating ambiguities that could arise from different interpretations of required operations and parameters. It forms the second deliverable within the scope of work for the EC JRC Contract Notice 2009/S 107-153973, as awarded to RSW Geomatics, 1Spatial and Rob Walker Consultancy.

Legal Notice

Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use which might be made of the following information.

The views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect the views of the European Commission.

Table of Contents

Document Information **2**

1. Executive Summary **5**

2. Introduction **6**
2.1 Purpose 6
2.2 Scope 6
2.3 Intended Audience 6
2.4 Terms and Definitions 7
2.5 Structure of the Document 7

3. Architectural Goals and Constraints **10**
3.1 EC Regulations 10
3.2 Mapping Flexibility 10
3.3 Open Interfaces 10
3.4 Statelessness 10
3.5 Separation of Control Messages from Data Transfer 11
3.6 Schema Agnostic 11
3.7 Automated Process 12

4. Use Case View **13**
4.1 Correspondence Between Use Cases and INSPIRE Regulation 13
4.2 Query Transformation 16
4.3 Actors 17
4.4 Use Case: Store Configuration 18
4.5 Use Case: Transform Data 19
4.6 Use Case: Gather Technical Information 22

5. Logical View **25**
5.1 Referenced Standards 25
5.2 Component Diagram 26
5.3 Transformation Service 28
5.4 (External Component) Schema Mapping Designer 30
5.5 (External Component) XML Repository 31
5.6 (External Component) Source Datastore 32
5.7 (External Component) Target Datastore 33
5.8 (External Component) Service Registry 34
5.9 (External Component) Client Application 34
5.10 (External Component) Supporting Systems 34
5.11 (External Component) Backoffice Tools 34
5.12 Web Service Operations 35
5.13 Use-Case Realizations 40

6. Data View **43**
6.1 Rationale for Choice of Schema Description Language 43
6.2 Schema Mapping Interchange Format 44
6.3 Schema Mapping Definition Process (Supplementary) 48

7. System Qualities **49**
7.1 Interoperability and Vendor Neutrality 49
7.2 Extensibility 49
7.3 Capacity 49
7.4 Availability 49
7.5 Performance 49
7.6 Handling of Credentials 49
7.7 Error Policy 50
7.8 Reliability, Security, Quality of Service, Rights Management, etc. (Supplementary) 50
7.9 Testing Policy (Supplementary) 50

8. Implementation View (Supplementary) **51**
8.1 Web Service Interface 51
8.2 Data Manipulation Engine 51
8.3 Spatial Process and Support 51
8.4 Configuration Parser 51
8.5 User Interface 52

9. Deployment View (Supplementary) **53**

Appendix A: Interface Specifications **54**

Appendix B: Sample SOAP Requests **59**
Sample Transform Operation Request Message 59
Sample Schema Description Document 60
Sample Mapping Document 61

Appendix C: Rationale for use of Rule Interchange Format **65**
Requirement for a Mapping Interchange Format 65
Mapping Transfer Language *versus* End-Use Language 65
Selection of Rule Interchange Format 66

Appendix D: Compatibility of Rule Interchange Format and GML **69**

Appendix E: Common Transformations Expressed Using Rule Interchange Format **72**

Appendix F: Supported Simple Features Functions and Predicates **86**

Appendix G: Definitions, Acronyms, Abbreviations and Initials **87**

Appendix H: References **90**

Technical Guidance

1. Executive Summary

The INSPIRE Directive (2007/2/EC) [1] aims to establish a European 'Spatial Data Infrastructure' (SDI) based on existing National SDI in member states. The European SDI will support policy-making for the protection of the environment.

INSPIRE aims to achieve harmonisation of spatial datasets and infrastructure through the provision of interoperable services. These services will enable users to find, browse, share and download digital spatial data held in heterogeneous data repositories using a standard set of methods and techniques.

The INSPIRE Regulation [3] as amended by [4] outlines the functional requirements of an INSPIRE Transformation Network Service. This report provides an illustration of how these functional requirements can be achieved and how a Schema Transformation Network Service can be realised.

This guidance captures key use cases and an architecture for implementing a Schema Transformation Network Service. It details a network service that loads source data from a Web Feature Service (WFS) [34] or an FTP Site, performs a transformation based on a mapping definition and outputs the INSPIRE-schema compliant data to a Transactional WFS (WFS-T) [35] or an FTP Site.

This Technical Guidance has focused on using open standards, and, where possible, standards common to the wider (non-spatial) community. For this reason, it recommends the use of standard web service technologies such as SOAP/WSDL and WS-Addressing. Accordingly, it has not adopted the Web Processing Service (WPS) model which is specific to the geospatial community.

By documenting precise interface specifications, logical software components and the system qualities that must be considered for an implementation of an INSPIRE Schema Transformation Network Service, the following conclusions have been reached:

1. The mapping definitions will be provided in W3C Rule Interchange Format (RIF) [39]. This will enable the expression of the complex transformations required to map between domestic schemas and the INSPIRE schemas. RIF is an open standard designed for interoperability and interchange, so is ideally suited to the definition of a generic Schema Transformation Network Service. It is also extensible to include spatial predicates and functions. The mapping functionality capabilities of RIF are comparable to those identified in the State of the Art Analysis [9] and tool vendor survey.

2. Source schema descriptions will be provided as ISO GML application schemas. These are defined in Section 6.2 of [26]. These can be imported directly into the RIF mapping definitions, providing a standard interoperable interchange format and supporting spatial types and concepts.

3. Configuration items (all of which are XML encoded) such as the mapping and schema definitions, will be managed though an XML repository. This improves the decoupling of services and encourages collaboration between service providers and users. In addition, it provides better management of multiple versions of mapping and schema definitions.

2. Introduction

This section gives an overview and general introduction to the Technical Guidance for Schema Transformation Network Services.

2.1 Purpose

This technical guidance document provides a concrete interface specification and supporting documentation for the Schema Transformation Network Service. This will enable interoperability by alleviating ambiguities that could arise from different interpretations of required operations and parameters.

It forms the second deliverable within the scope of work for the EC JRC Contract Notice 2009/S 107-153973, as awarded to RSW Geomatics, 1Spatial and Rob Walker Consultancy.

2.2 Scope

This document provides a proposal to Member States for the implementation of a Network Service of type Schema Transformation in accordance with the INSPIRE Regulation [3] as amended by [4]. This applies to Legally-Mandated Organisations (LMO) who wish to publish their data "as is" according to their local logical schemas, and make provision for data consumers to transform the data as part of a download request.

This document details the technical aspects of the interfaces and characteristics of the Schema Transformation Network Service. It does not detail how the Service is combined or orchestrated with other services, nor how the Service should be implemented internally. Note also that testing of the prototype is not describe in this document but instead is detailed in the Prototype Report [62].

Transformation Network Services can be categorised into different areas of functionality: for example, transforming data formats (e.g. from a proprietary format to GML), coordinate reference systems (CRS) and logical schemas.

This technical guidance applies only to the transformation of logical schemas. It assumes that other types of transformation, such as data encoding format and CRS, are handled by other services; the technical guidance available for those other categories (see e.g. [11], [12]) complements the guidance presented in this document. These other transformations are performed prior to the schema transformation.

The technical challenges of schema transformation are greater than for other forms of transformation such as coordinate reference system and natural language transformation. This is because of the enormous variety of data models, encodings, transfer and storage formats and other factors that are required in order to achieve harmonisation of data in a common, European format as mandated by INSPIRE. For this reason, the technical guidance has become of necessity longer and more complex than for other INSPIRE network services.

2.3 Intended Audience

This document is aimed at Schema Transformation Network Service implementers and service integrators working on behalf of INSPIRE LMOs (see definition of service integrator in Section 4.3).

Readers are assumed to have a general understanding of the INSPIRE directives and, for some sections, an understanding of web services and related technology.

2.4 Terms and Definitions

The following definitions are the same as those detailed in [9] and are reproduced here for convenience.

Term	Definition
Data model	A model of the (geographic) data that is stored and/or exchanged.
Conceptual model	A model that defines concepts of a universe of discourse.
Conceptual schema	A platform-independent (or platform-specific), conceptual model expressed using a formal modelling language (such as UML).
Logical or application schema	A platform-specific description of the structure and constraints applicable to the storage of data for one or more applications (expressed, for example, as an XML Schema (XSD)).
Physical data model	Synonym for logical or application schema.
Physical schema	The concrete, implementation-specific description of how the data is organised in the storage technology of choice (expressed, for example, as SQL DDL).
Data instance	A single item of data expressed in a concrete storage format (for example, an XML element or database record) which corresponds in some way to an object in the real world such that it is capable of being expressed as an object in an ontology, rather than merely as a predicate or attribute of an object.
Instance data	A collective term for data instances, sometimes known as “row-level data” (especially in a database context).
Schema	A general-purpose term, rather imprecise in nature, that may refer to a generic data model, ontology or database storage structure, depending on the context.
INSPIRE Application	Software using INSPIRE network services (without access to a portal). NB This is the same definition as that given in Section 4.1 of [6].

Table 1 Terms and definitions

2.5 Structure of the Document

This document presents the architecture of a Schema Transformation Network Service as a series of views. These capture the interface specification and supplementary information that will be required by vendors who are implementing or integrating the Schema Transformation Network Service. These views are presented with supporting UML diagrams and sample XML code. The remainder of the document is divided into the following sections:

Section/Page	***Title***	***Description***
Section 3	Architectural Goals and Constraints	A brief overview of the main goals and constraints of an INSPIRE compliant Schema Transformation Network Service.
Section 4	Use Case View	Details of the requirements of the Schema Transformation Network Service, presented as a series of use cases, traced from the operations mandated in the INSPIRE Regulation [3] (as amended by [4]). These use cases described interactions with the system, detailing the pre and post

Section/Page	*Title*	*Description*
		conditions. They focus on describing the purpose and context of an interaction, rather than detailing the request and response parameters.
Section 5	Logical View	An overview of the Schema Transformation Network Service's components, including a detailed textual description of the system's primary components and the operations it supports. Details of open standards or languages used within the interface and any extensions to these standards or languages. Use case realisations illustrate how the logical schema contributes to the successful implementation of the use cases.
Section 6	Data View	Specific details of the schema description and mapping languages, including details of how they should be used together. Includes best practices for the construction of successful and unambiguous transformations.
Section 7	System Qualities	Details of a number of system qualities which must be considered during the implementation of the INSPIRE Schema Transformation Network Service.
Section 8	Implementation View (Informative)	An illustration for an INSPIRE-compatible Schema Transformation Network Service that is implemented using existing transformation tools.
Section 9	Deployment View (Informative)	Options for physical network configurations on which the Schema Transformation Network Service will run, to aid service integrators in determining which approach will be the most suitable (see definition of service integrator in Section 4.3).
Appendix A	Interface Specifications	Full interface specifications in a suitable format for importing directly into an implementation.
Appendix B	Sample SOAP Requests	Example Schema Transformation Network Service request and response parameters, including details of schema mappings.
Appendix C	Rationale	Further details of significant decisions made in the process of constructing the technical guidance.
Appendix D	GML-RIF Compatibility	A worked example showing the compatibility between GML schema description language and RIF mapping definition format.

Section/Page	*Title*	*Description*
Appendix E	Common Transformations	Common transformations expressed using RIF.
Appendix F	Definitions	Definitions, acronyms, abbreviations and initialisms.
Appendix G	References	Referential section.

Table 2 Document Sections

The majority of this technical guidance document contains recommendations for the construction of the Schema Transformation Network Service. To aid service implementers and service users, additional information has been included where it may be useful. Such sections are marked clearly throughout this document with the suffix “(Supplementary)”.

3. Architectural Goals and Constraints

This section identifies generic, high level features, technical risks or overarching constraints of the Schema Transformation Network Service, of a kind that are expected to have a significant architectural impact.

3.1 EC Regulations

A Network Service of type Transformation must meet the provisions of the INSPIRE Regulation [3] as amended by [4].

The Schema Transformation Network Service must fulfil the needs of this Regulation relating to logical schemas. This includes the transformation of data from a source logical schema to the INSPIRE logical schema.

3.2 Mapping Flexibility

This guidance seeks to define an interface that is rich enough to allow implementations to support transformation of datasets held in a wide variety of source schemas into equivalent INSPIRE schemas. Although the interface must support this range of complexity, each implementation of the Schema Transformation Network Service will have limitations imposed by its supporting technologies and, as such, may only support a subset of the source and INSPIRE schemas.

The INSPIRE schemas are defined by the INSPIRE Regulation [3] as amended by [4], as guided by the INSPIRE data specifications [8]. They should include all themes (including those that are currently, or will in future be, under development). The source schema is determined by the data providers, typically following their standard data capture and logical and conceptual schema development processes. This may change over time.

As a result, it should be possible to configure the Schema Transformation Network Service to work with a wide variety of source and target schemas. The configuration must be flexible enough to support those types of schema transformation identified in the State of the Art Analysis [9] Appendix B “Schema Transformation Levels”.

3.3 Open Interfaces

In order to enable interoperability within INSPIRE based projects, the Schema Transformation Network Service should be implemented based on the common interface specification defined in the INSPIRE Regulation [3] as amended by [4] and share the common characteristics described therein (including metadata requirements). These must not be tied to a particular transformation system’s software (commercial or open-source). It should be possible to create multiple implementations of the Schema Transformation Network Service, each using a different underlying transformation engine.

If the transformation engine has to be replaced (for example, due to performance, cost or other features) or, alternatively, another Schema Transformation Network Service is to be consumed, then it should be possible to do so without re-writing the schema mapping definitions.

3.4 Statelessness

This guidance describes a stateless interface. In effect, this means that all the information that a service requires to perform a transformation is provided in the initial operation request; thus, there is no need for the client to perform any other interaction with the service. To this end, the service does not store the transformed data; instead, the transformed data is transferred to a location (either WFS-T or FTP site) nominated by the client in the initial operation request.

Stateless design has important implications in terms of scalability and resilience, because it makes the service more amenable to load balancing (a process whereby requests to a single virtual service are routed automatically to one of several actual services).

Another import facet of this stateless approach is that the service is never responsible for the storage of persistent data. This means that any data cached during the transformation process can be, and should have been, discarded before the operation completes. This simplifies questions of data management and should help to minimise licensing issues associated with the storage and processing of the spatial data.

3.5 Separation of Control Messages from Data Transfer

The XML encodings of web service requests and responses are typically small (a few kilobytes or less). Modern web service development platforms have been designed with this as an assumption and therefore, for speed and ease of use, they process the request in memory.

However, spatial datasets are different in that, when they are encoded in XML, they can grow to be very large (many hundreds of megabytes). If the spatial datasets were to be embedded directly in either the request message or the response message of a Schema Transformation Network Service, they would be too large to be processed directly in memory.

This guidance defines an interface that is designed to be implemented using any modern web services platform. To this end, spatial datasets are never passed directly through the service interface (either in the request or the response) but are, instead, passed by reference. This allows the most appropriate technology to be used when handling the actual transfer of the spatial data.

A further advantage is that, in order to support audit and debug, web service infrastructure may persist a record of the messages sent to and received from a service. If the spatial datasets were passed by value through the interface, this would mean that the whole dataset would have be handled – and, possibly, stored - by intermediate infrastructural components. If however, only references were passed, the infrastructure would be able to make a precise and compact record of service activity.

Spatial dataset schemas (when expressed as GML application schemas), and also mapping definitions, can be relatively large, and so passage of these parameters by reference is recommended for these artefacts as well.

Recommendation: Passage of parameters should be by reference for reasons of performance, flexibility of deployment and service manageability.

3.6 Schema Agnostic

This guidance describes an interface that is entirely schema-agnostic. That is, to say, it embeds no knowledge of the structure of any source schema or any INSPIRE schema in the request message. This allows the interface definition to remain constant even when these schemas change. A consequence of this is that all data transformed by the system is considered equal; specifically, the system will handle identifiers, data and metadata in the same manner – all INSPIRE identifiers, data and metadata must be derived from the identifiers, data and metadata available in the source datasets.

3.7 Automated Process

The Schema Transformation Network Service should be supported by an operating environment that includes features such as orchestration of services, security, rights management and quality of service provisions, that function as automated network services This enables it to provide robust, accessible online services that are safe and secure, without requiring constant manual intervention.

4. Use Case View

This section documents scenarios within which the Schema Transformation Network Service must operate. These all represent significant aspects of Schema Transformation Network Service functionality. This section is based on an analysis of the INSPIRE Regulation [3] as amended by [4] and proposals for the Schema Transformation Network Service. Each use case describes what happens during the interaction, what the client (actor) does and how the system responds. It also identifies the list of pre-conditions that must be met before a use case can start, and a list of post-conditions that will be true once the use case has finished.

4.1 Correspondence Between Use Cases and INSPIRE Regulation

This section maps the operations identified in the INSPIRE Regulation [3] as amended by [4] onto the use cases proposed by this guidance. It contains only an overview for the purposes of traceability. Further details of the expected parameter datatypes and their permitted contents are given in Section 5.12.

4.1.1 Use Case Traceability

The Schema Transformation Network Service presented in this guidance conforms to the abstract service definition established in the Draft INSPIRE Regulation. In the following table, the mapping is presented from the abstract operations introduced in the Draft INSPIRE Regulation to the use cases described in this section.

Abstract Operation	***System Use Case***
Get Transformation Service Metadata	Gather Technical Information
Transform	Store Configuration
	Transform Data
Link Transformation Service	Link Transformation Service

Table 3 Operations and Use Cases

4.1.2 Get Transformation Service Metadata

Although the Schema Transformation Network Service is responsible for returning the metadata defined by the Draft INSPIRE Regulation's Get Transformation Service Metadata operation, this operation has been included in the slightly more wide-ranging Gather Technical Information use case. It is expected that this use case will form part of a larger set of use cases that describe the process of integrating the Schema Transformation Network Service with existing spatial infrastructure. This larger set of integration related use cases is beyond the scope of this guidance.

Abstract Parameter	***Type***	***System Parameter***
Language	Request	Accept-Language HTTP Header (see [63]). When this is set, the content of the metadata will reflect the language (if supported); however, the standard WSDL material will remain unchanged. If the language is not supported, the metadata will be supplied in the default language of the service. Browsers will normally configure this attribute automatically so that users will see the metadata in their natural language by default (if supported).
Transformation Service Metadata	Response	This will be included in the body of the WSDL response. The client will be able to differentiate this from the standard WSDL material because it will be in the INSPIRE namespace.
Operations Metadata	Response	This will be included in the body of the WSDL response. The client will be able to differentiate this from the standard WSDL material because it will be in the INSPIRE namespace.
Languages	Response	This will be included in the body of the WSDL response. The client will be able to differentiate this from the standard WSDL material because it will be in the INSPIRE namespace.

Table 4 Mapping of Parameters for the Get Transformation Service Metadata Operation

See Section 4.6 for more details.

4.1.3 Transform

The logical Transform operation has been mapped to two system use cases: the Store Configuration use case and the Transform Data use case. This reflects the fact that the Store Configuration use case is only used when the mappings are being designed or maintained, whereas the Transform Data use case is used every time a transformation is required (whether or not the mapping has changed).

The mapping of abstract to actual parameters for the Transform operation is shown in this table.

Abstract Parameter	***Type***	***System Parameter***

Input Spatial Data Set	Request	`soap:Envelope/soap:Body/tns:TransformRequest/sourceDataset/WFSReference/capabilitiesURL`
		`soap:Envelope/soap:Body/tns:TransformRequest/sourceDataset/WFSReference/layer`
		`soap:Envelope/soap:Body/tns:TransformRequest/sourceDataset/WFSReference/maxFeatures`

Source Model	Request	Derived from source dataset
Target Model	Request	`soap:Envelope/soap:Body/tns:TransformRequest/targetDataset/directDownload/schema`
Model Mapping	Request	`soap:Envelope/soap:Body/tns:TransformRequest/mapping/directDownload`
Spatial Data Set	Response (logically, although in practice this is an Out parameter passed in the Request)	`soap:Envelope/soap:Body/tns:TransformRequest/targetDataset/directDownload/url`

Table 5 Mapping of Parameters for the Transform Operation

In the above table, the Input Spatial Data Set is expressed in the interface as a WFSReference, which is a complex type for which the formal description is given within the WSDL in Appendix A.

Note that the `capabilitiesURL` element gives the location for retrieving information about the layers and services provided by the WFS. The `layer` element is a comma-separated string value containing the prefixes and names of the layers included in the source dataset. Also, the `maxFeatures` with value –1 (minus one) implies no upper limit on the number of source features being transformed. A value in this field greater than or equal to zero implies an upper limit on the same.

The Source Model and Target Model parameters are essential because the mapping itself does not contain sufficient information for the transformation engine to understand the model over which the mapping is applied.

Whereas in the abstract operation, Spatial Data Set is a response parameter, it has been modified in the WSDL to be a request parameter in order to permit the use of WS-Addressing [25] headers which support the asynchronous invocation of the Transform operation. Thus, the client specifies the location to which the Service will output the transformed data.

4.1.4 Link Transformation Service

The INSPIRE Regulation [3] as amended by [4] lists the Link Transformation Service operation as a mandatory part of the Schema Transformation Network Service, giving the following explanation of its role: "Allows the declaration of availability of a Transformation Service for Transforming Spatial Data Sets through the Member State's Transformation Services while maintaining the Transformation capability at the Public Authority or the Third party location." The mapping of abstract to actual parameters for the Link Transformation Service operation is as shown in this table.

Abstract Parameter	***Type***	***System Parameter***
Information About Public Or Third-Party Transformation Service	Request	`soap:Envelope/soap:Body/ tns:LinkTransformationServiceRequest/request`

Table 6 Mapping of Parameters for the Link Transformation Service Operation

If the request parameter supplied contains a non-null URL, the service will forward all future requests to the specified delegate (except for future calls to this operation). Conversely, if the request parameter is null, the forwarding is cancelled. Note that, because this service accesses resources using pass-by-reference, to be successful the delegate service must have access to any specified resources.

However, notwithstanding the above possibilities, it is recommended that service linking be done as part of the installation and configuration phase of a service's deployment, rather than as part of its run-time operation using a web service interface. Various technologies exist for forwarding web service requests from one environment to another. Hence, if (as appears to be the case) the purpose of this operation is to support location transparency and service virtualisation, there is no need for this to be implemented by the service itself. The existing IT infrastructure of the organisations involved will influence the selection of service virtualisation technology and, as such, it would be inappropriate to recommend a particular solution.

Recommendation: Service linking should be addressed as part of service installation and configuration, rather than be performed using a web service interface.

4.2 Query Transformation

The INSPIRE Network Services Draft Implementing Rules [5] describe some usage scenarios that require a query expressed in the INSPIRE schema to be translated to run against a source dataset. However, analysis of this requirement has led to a conclusion that the Technical Guidance should not support this functionality.

Query transformation is effectively a process of reverse-engineering the mapping definition. For simple class or attribute renaming, this would be trivial. However, the Schema Transformation Network Service interface must be general, and not constrained to these simple cases. For more complicated mappings that require class splitting, aggregation or spatial joins, the process of reverse-engineering the mapping definition by automated means is - at best - complex and, in some cases, computationally intractable.

Furthermore, there are many practical challenges, both technical and performance-related, to the concept of fully-automated, on-the-fly processes that involve transformation. These issues are compounded by a lack of clear evidence of a business use case. It is not clear who is responsible for providing integrated, query-supporting services with continuous support, nor who gives access to their raw data, nor how it should integrate with other transformation processes (e.g. edge-matching).

A more feasible solution to these problems would be to construct a managed system for transforming an entire dataset (potentially orchestrated with other services such as data quality with metadata, or manual fixup) and then to ask clients to access this cache.

The State Of The Art Analysis [9] highlighted the general architectural issues in terms of the strengths and weaknesses of the proposed architectures. The focus of this Technical Guidance is on schema transformation definitions and processes rather than wider business case analysis or architectural issues. Hence, this document avoids further detailed analysis and discussion of this matter.

4.3 Actors

The actors and their responsibilities associated with the Schema Transformation Network Service are shown in the following table.

Actor	*Description*
Data Expert	A user who has a detailed knowledge of a source dataset and is capable of working out the mapping required from the source dataset to the INSPIRE schema. This user would typically be a member of an LMO or a third-party agent acting for an LMO, or a team of both.
Client	A client application or system that accesses the Schema Transformation Network Service on behalf of the end user. This could be an INSPIRE application, an orchestration framework, or a download service that includes an integrated transformation step. The client could be operated by a private individual, academic institution, government agency, commercial or non-profit organisation. Also, note that the INSPIRE geoportal is a mandatory client to the Service (see Paragraph 20 of INSPIRE Directive [1] and also [60]).
Service Integrator	A developer who is responsible for creating an INSPIRE application (see the Client actor above).

Table 7 Actors' Responsibilities

These actors express roles rather than user identities. A role may, depending on the requirements, be performed either by a system component or a human actor. It is possible that a person, organisation or system may, in practice, represent more than one actor (e.g. if the service integration and data expertise aspects are handled by the same service integrator).

4.4 Use Case: Store Configuration

This section describes the Store Configuration use case.

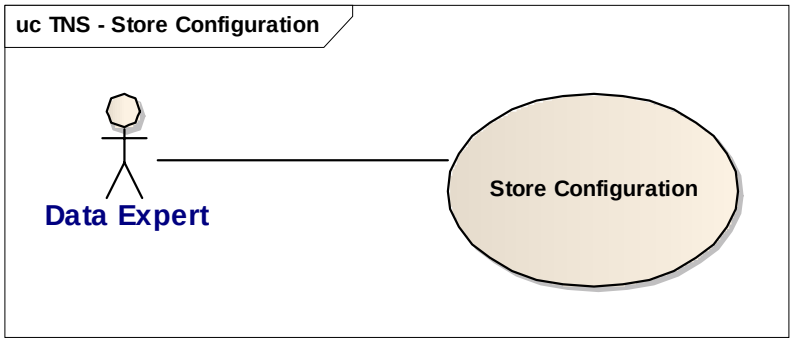

Figure 1 Store Configuration Use Case

Brief Description		
It is recommended that the source and target logical schemas and mapping definition documents be stored so that they are available for use when performing data transformation. A **Data Expert** may need to update the stored logical schemas or mapping definition in cases where: • The source logical schema has changed (either because new feature classes and/or attributes are available for transformation or as an incremental improvement and reduction in transformational data loss), • The target data model has changed (e.g. for maintenance or updates of INSPIRE data specifications), • The existing mapping definition does not identify all the data that is available for transformation in the source logical schema (or, possibly, it identifies data that is no longer available in the source logical schema).		
Basic Flow of Events		
Step	**Actor Interaction**	**System Responsibilities**
1	The **Data Expert** decides to be INSPIRE compliant in view of the requirements set out in the INSPIRE directive ([1]) and the associated Regulation [3] as amended by [4].	
2	The **Data Expert** requests that the system store a new version of a configuration item (i.e. a source or target logical schema or mapping definition), using a new and unique record identifier as the reference under which to store it.	System stores the new version of the configuration item.
Pre-Conditions		
Pre1	The **Data Expert** has a configuration item that is well-formed and valid.	
Post-Conditions		
Post1	The system has a copy of the configuration item, which may be used subsequently during data transformation.	

Table 8 Store Configuration

Figure 1 shows the data flow diagram for Store Configuration.

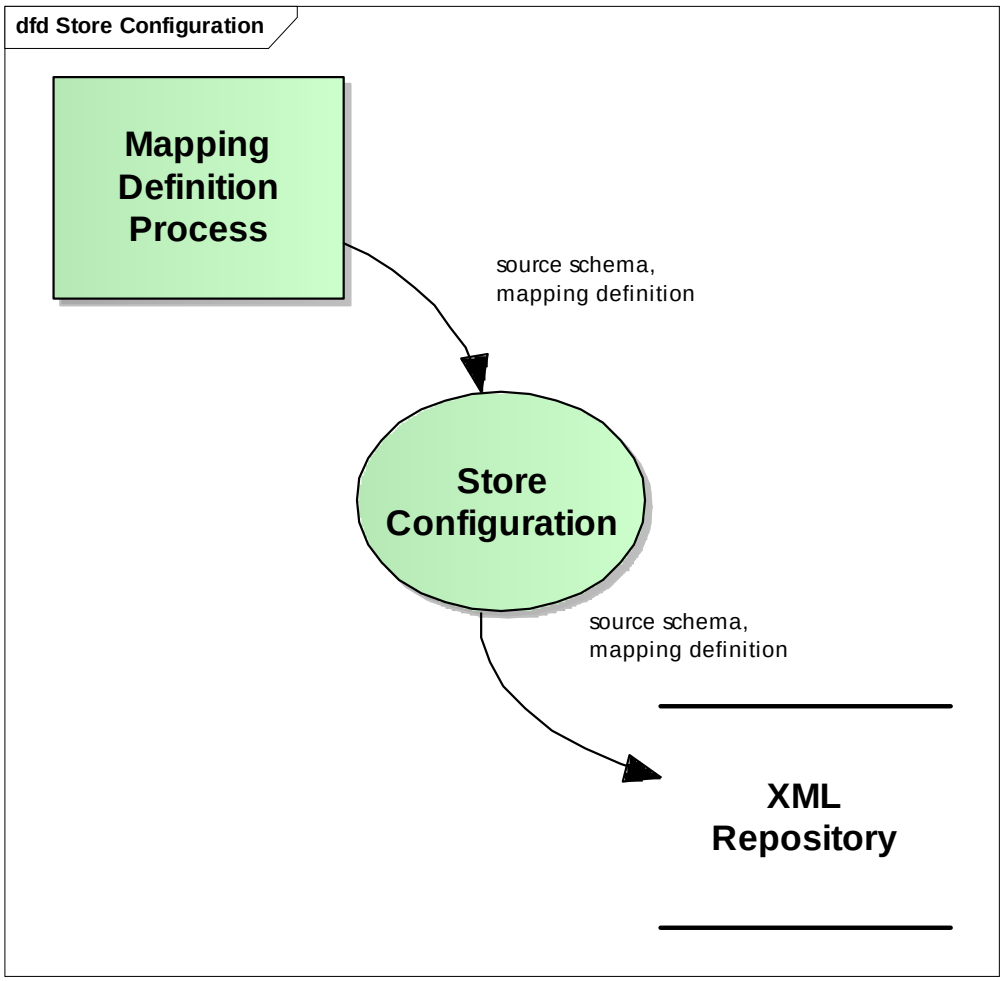

Figure 2 Data Flow Diagram for "Store Configuration" Use Case

4.5 Use Case: Transform Data

This section describes the Transform Data use case.

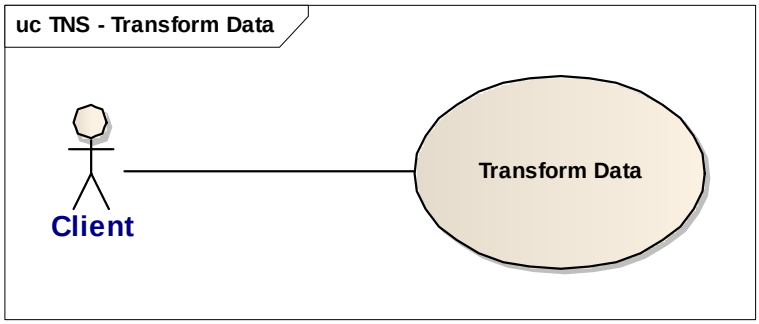

Figure 3 Transform Data Use Case

Brief Description		
This use case describes the main data transformation process. This involves orchestrating the sub-systems that need to work together to fulfil the request, such as retrieving the source dataset, logical schema mappings and schema descriptions, tasking specific software tools with the execution of a transformation and returning the result.		
Basic Flow of Events		
The use case starts when the **Client** wishes to transform a dataset into the INSPIRE logical schema as part of a chain of transformations which may include file format transformation, CRS transformation, natural language transformation, edge matching and other processes which aim to produce a harmonised dataset. The user invokes the Schema Transformation Network Service with a valid set of parameters and is informed when the process is complete.		
Step	**Actor Interaction**	**System Responsibilities**
1	**Client** invokes the transform web service operation on the Schema Transformation Network Service.	Syntactically validates the parameters (source and target logical schemas, transformation definition, source data location, target data location). A single source dataset may not contain all information required for the target schema, and therefore it may be necessary to reference the union of multiple source datasets in the mapping. Conversely, a single source dataset may contain data that is required in multiple target schemas. The source location information is required when data is being passed by reference, which is the recommended scenario; where data is passed by value, it is not necessary to specify the location. The target location information is necessary in all cases so that the client can know where to obtain the transformed data.
2		Retrieves the source data, performs the transformation and writes out the transformed data.
3	Processes the call-back message.	Informs the user that the process is complete, returning metadata about the processing and any error messages.
Pre-Conditions		
Pre1	The source data must be exposed and accessible as either GML 3.1.1 or GML 3.2.1.	
Pre2	The source logical schema must be exposed in an appropriate format. That is, to say, it is recommended to be described as a valid GML application schema using either GML 3.1.1 or GML 3.2.1.	
Pre3	The target logical schema must be defined and available in the system (this could be accessed as an online resource, e.g. the INSPIRE Registry). This must refer to the current INSPIRE application schema.	

Pre4	The types defined in the source logical schema must include the non-voidable attributes required in the target logical schema as per the regulation Implementing Directive 2007/2/EC of the European Parliament and of the Council as regards interoperability of spatial data sets and services (see [3] as amended by [4]).
Pre5	Transformation definition must be defined and available in the system. This must be applicable to source logical schema.
Pre6	Resources must be available for storing the transformed data.
Post-Conditions	
Post1	The transformed dataset is available. All the mappings supplied to the service have been performed successfully without any unexpected data losses. The completeness of the transformed dataset is constrained by the data available in the source datasets and the thoroughness of the mapping definition.
Post2	The transformed dataset is conformant to the INSPIRE application schema.

Table 9 Transform Data

There is no need for the client to perform a GET TRANSFORMATION SERVICE METADATA OPERATION, or equivalent operation. This is because the proposed interface is fully defined using SOAP/WSDL. When clients are generated, standard frameworks (like .NET [57] and JAXB2.0 [58]) generate client-binding code automatically; this is a once-off activity that is performed at compilation time, and not during day-to-day interaction between the client and service.

The Transform Data operation also covers the functionally of the logical IS TRANSFORMABLE operation. In practice, it is only by considering the source schema, the source data and the target schema that it is possible to determine whether a mapping is possible. The service will only have access to the source data during the transform operation; the IS TRANSFORMABLE and TRANSFORM operations have therefore been combined. Even though a mapping may be possible logically, it may not be supported by a particular implementation of the Schema Transformation Network Service due to limitations in its supporting technology.

To support efficient creation of the mapping definition, it is expected that any mapping testing that could be performed by consulting the source and target schemas (but not source data), would be performed by the external mapping definition client.

Figure 4 shows the data flow diagram for Transform Data.

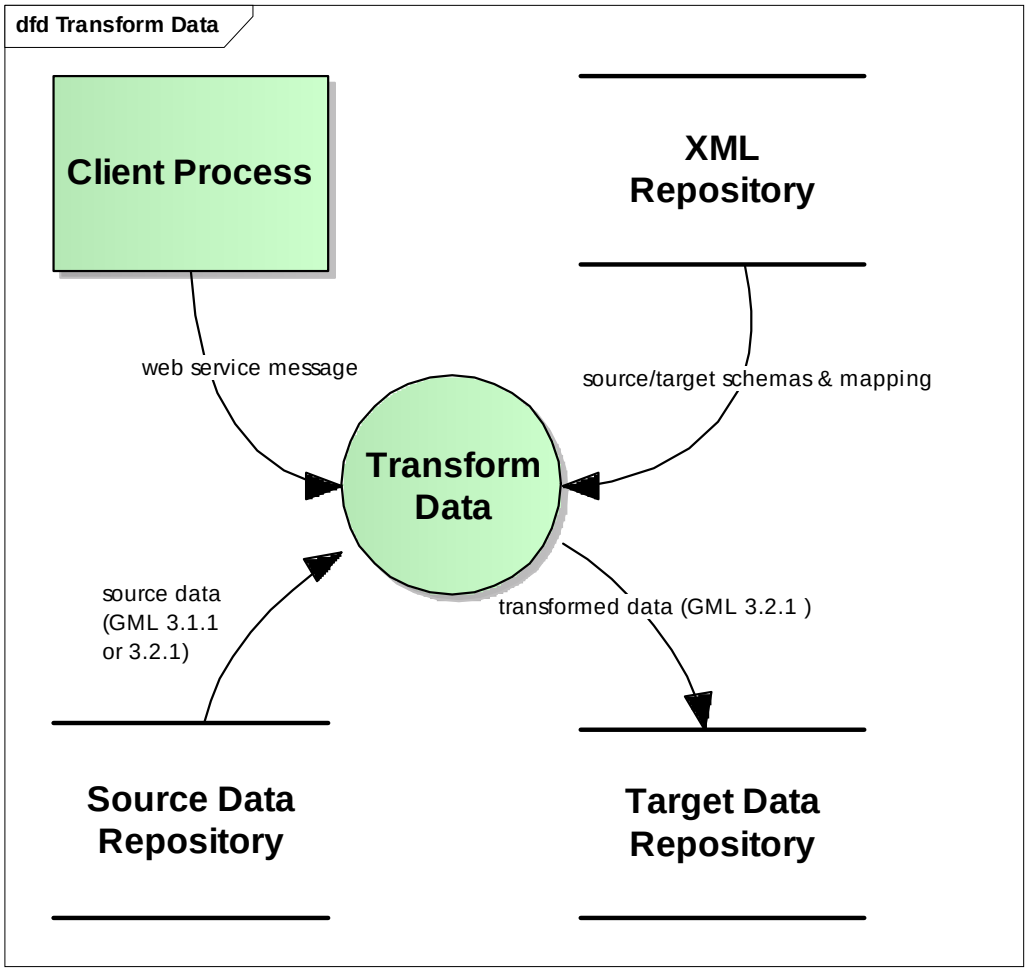

Figure 4 Data Flow Diagram for "Transform Data" Use Case

4.6 Use Case: Gather Technical Information

This section describes the Gather Technical Information use case.

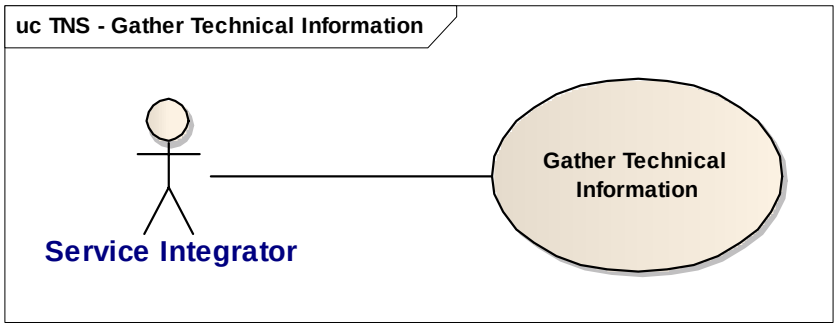

Figure 5 Gather Technical Information Use Case

Brief Description		
The **Service Integrator** is starting an integration process and needs to gather information about the Schema Transformation Network Service in order to do this. This will support decision making about which service to invoke and with what parameters. This use case maps to the abstract Get Transformation Service Metadata operation defined in the Draft INSPIRE Regulation [4].		
Basic Flow of Events		
• The use case starts when the **Service Integrator** decides that they require metadata on the capabilities of the Schema Transformation Network Service. • The **Service Integrator** queries the system to obtain details of the supported operations. • The **Service Integrator** queries the system to obtain details of the configured schemas and mapping definitions.		
Step	**Actor Interaction**	**System Responsibilities**
1	Requests details of the supported operations	Provides these details in a machine readable format
2	Requests details of the configured data models and mappings	Provides these details in a machine-readable format.
Pre-Conditions		
Pre1	Data models and mappings have been configured	
Post-Conditions		
Post1	The **Service Integrator** is aware of: • Operations supported by the Schema Transformation Network Service • Source and target schema registered with the XML Repository. • Mappings configured in the XML Repository.	

Table 10 Gather Technical Information

Note that it is the service integrator and not the client actor that is charged with performing the Gather Technical Information use case. This is because it is not appropriate for the client to attempt to assemble the required information. The client will not necessarily be in possession either of the logical schema expertise, nor the technical information required to assess the capabilities of the Schema Transformation Network Service. This shows a major difference between the Schema Transformation Network Service and other INSPIRE services (such as the View, Download and Discovery Services). Whilst those services are amenable to automated discovery of their capabilities, transformation deals with a more complex and diverse challenge.

Figure 6 shows the data flow diagram for Gather Technical Information.

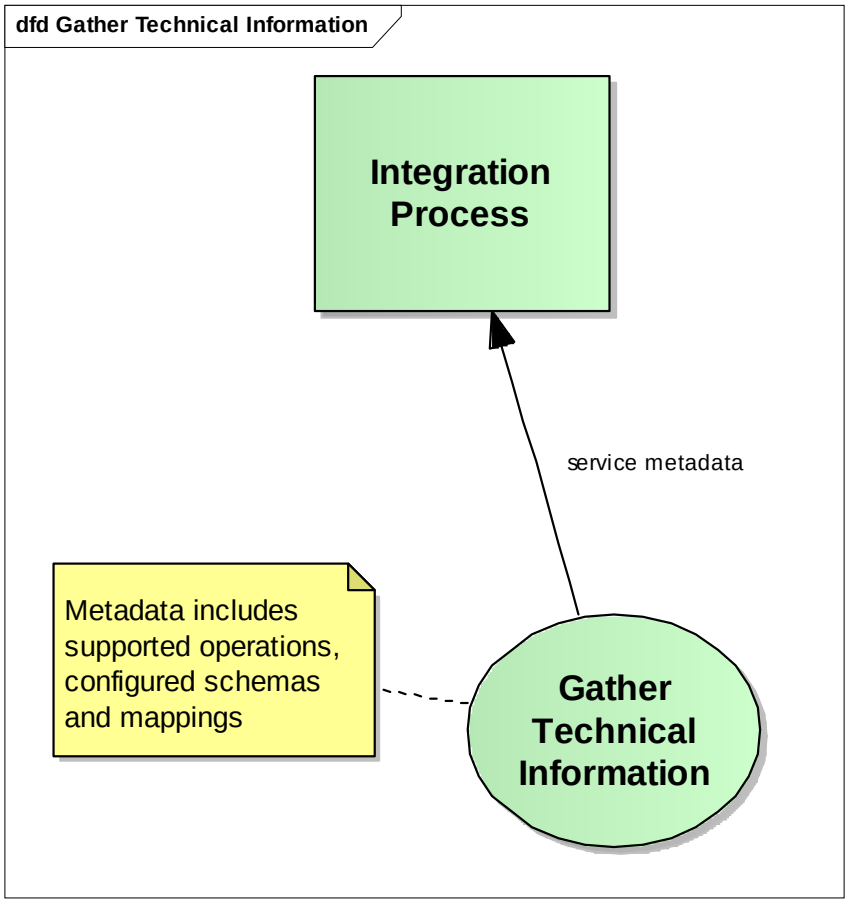

Figure 6 Data Flow Diagram for "Gather Technical Information" Use Case

5. Logical View

This section describes the significant aspects of a Schema Transformation Network Service, including the detailed interface specifications.

5.1 Referenced Standards

This subsection gives an overview of standards used by the Schema Transformation Network Service and details of specific version numbers or references to further information. A prefix is defined for each standard, which will be used in the interface specifications below.

- ***SOAP and WSDL***
  SOAP [*24*] is a protocol for exchanging structured information, based on XML. The SOAP versions and technical details should follow the INSPIRE Network Services SOAP Framework technical report [*7*]. The SOAP framework is well placed to meet the needs of relatively complicated services that have a relatively small user base. The rigorous nature of the interface definition allows significant parts of the client implementations to be automatically generated (in a range of languages). Simple services that have very large user bases can benefit from alternative approaches (like RESTful interfaces), however this does not seem a good fit for the technical complexity and demographic of the INSPIRE community.

- ***WS-Addressing***
  WS-Addressing [*25*] is the leading standard to aid the creation of asynchronous methods in SOAP web services. It is a well-documented open standard with increasingly effective tools support. Although tools support is not as mature as for foundational technologies like SOAP/WSDL there is no reasonable alternative for asynchronous web service integration.

  WS-Addressing provides the mechanism by which the requester is informed of the outcome of the Transform request. A SOAP header is prepended to the Request message which contains the callback location at which the requester will expect the Schema Transformation Network Service to register a callback and indicate the execution status of the asynchronous operation. The Service persists the parameterised data in its internal state during the Transform operation, and the manner in which it stores its state is implementation-dependent (this could be, for example, a queueing-system or relational database). However, once the Transform operation has completed, the requester has been informed and the transformed data has been transferred to the target (either WFS-T or FTP), the Service is free to discard any temporary internal storage. If the Service encounters problems during the transformation, it will inform the requester via the callback and release any temporary internal storage.

  The Prototype Report [62] Section 2.2 describes the version of WS-Addressing that was used during the prototyping exercise and its suitability for the purpose specified.

- ***Rule Interchange Format (RIF)***

  The W3C Rule Interchange Format (RIF), see [13], is highly suited to the task of representation of schema mapping definitions as collections of business rules. This document will make a recommendation that it be adopted as the interchange format for mappings (see Section 10.1.3 infra). Of the available RIF dialects, Production Rule Dialect (PRD) [18 is the most suited to the combination of rule conditions and consequent actions. See Appendix C for a more detailed justification for this choice. This is purely an interchange format – it is anticipated that most Schema Transformation Network Service implementers will make use of existing spatial tools rather than redefine their implementation based on RIF. Where this is the case, the Schema Transformation Network Service implementation must provide functionality to translate between RIF and the implementer's internal format. The Schema Transformation Network Service State of the Art Analysis [9] identified that there is no commonly-used interchange format for mapping definitions, but most surveyed tools were capable of performing a wide variety of transformations. RIF provides a good, vendor neutral format for interchange of mapping definitions. For further details, refer to:

  Section 6.2 Schema Mapping Interchange Format.

  Appendix C Rationale for use of Rule Interchange Format.

  Appendix D Worked example showing the compatibility between RIF and GML.

  Appendix E Common Transformations: use of RIF to define schema mappings.

- ***Geography Markup Language (GML)***

  Schema definitions will be provided to the Schema Transformation Network Service in the format of GML [26] application schemas.

  Further details of the schema description and choice of standards are provided in Section 6.1.

5.2 Component Diagram

Components that are outside the scope of the Schema Transformation Network Service (but required for successful operation of a the service) are indicated with the preface (External Component). For example, the XML repository component provides storage and versioning of configuration data (e.g. models and mappings); it could be implemented by more than one physical service, such as the online INSPIRE Registry [27] which could, in future, include INSPIRE schemas.

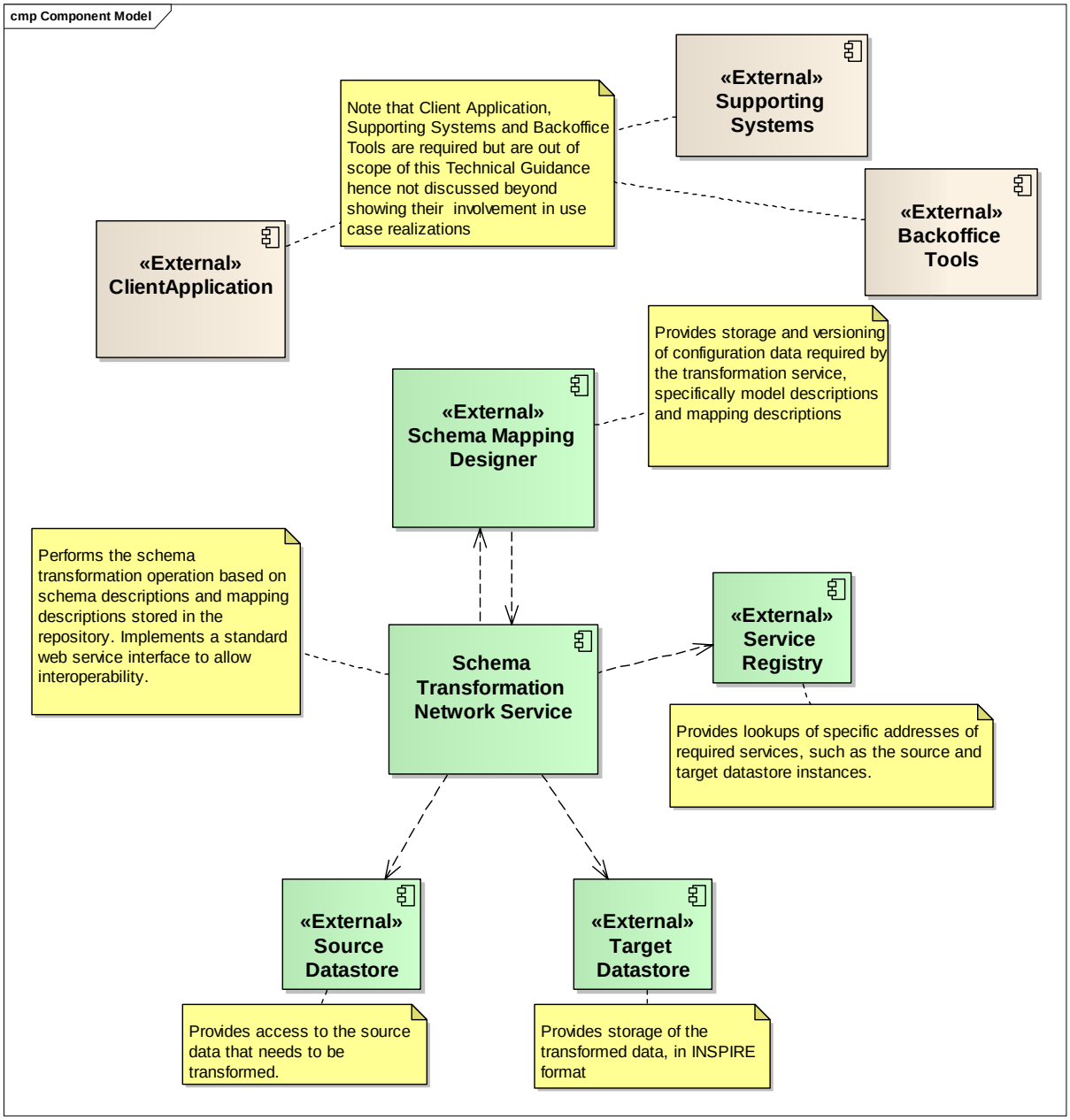

Figure 7 Component Model

In the above diagram, note that the source and target logical schemas and mapping definition do not need to be available in the same external repository: it is merely a logical convenience to refer to them as in a single repository; they could, in practice, be in different repositories. See Section 5.3.1.3 for more information.

Note that the Source Datastore must expose its data as GML 3.1.1 or 3.2.1 (see precondition 1 to the Transform Data use case).

The following sub-sections provide guidance for the implementation of each of these components, including detailed interface specifications where appropriate. This is a logical overview of the service components and therefore may not directly reflect implementation components; in particular, implementations may choose to package several components within a single application.

5.3 Transformation Service

This section describes the realization of the Transformation Service component: its responsibilities, collaborations with other components and rationale for its existence.

Component	Transformation Service
Responsibilities	• Creating data that is conformant with the schema described in the INSPIRE data specifications based on source data that is provided in a different schema. • Provide a standard interface for transformations that may be used by other INSPIRE applications, including orchestration systems. Note that the INSPIRE geoportal (see Paragraph 20 of INSPIRE Directive [1] and also [60]) is a mandatory client to the Schema Transformation Network Service.
Collaborations	• The *Source Datastore* is used to obtain source data, which must be exposed as either GML 3.1.1 or 3.2.1. • The *Target Datastore* is used to store transformed data. • The *Repository* is used to store definitions of the source and target schema and also the definition of the mapping between these logical schemas.
Rationale	• This is the main service required by a Schema Transformation Network Service.

Table 11 Transformation Service Component

The remainder of this section gives an overview of technical principles used within the service, and details of standards referenced or used by the service. A detailed description of all the operations within the service is located in Section 5.12.

Further details are available in these sections of the document:

- Section 6 Data View: How to use the service and develop its configuration.
- Section 6.1 Schema Description Language.
- Appendix D: Compatibility of RIF and GML.

5.3.1 Technical Principles

The following technical principles underpin the implementation of the Schema Transformation Network Service as described in this technical guidance.

5.3.1.1 Web Service Technology

The Schema Transformation Network Service should be implemented as a SOAP web service with WSDL definitions and be compliant with WS-I Basic Profile [28]. The use of these standard web service technologies ensures that we meet the key architectural goal of using open interfaces and reduces the difficulties involved in meeting the key architectural goal of supporting business processes. These standards are supported by most development toolkits (including Java and .net) and most modern orchestration frameworks.

SOAP/WSDL is the leading industry standard for web service communication. The only alternative in common usage is a RESTful interface. RESTful interfaces have proven successful where there are a small number of services and a very large client base (see for example Google's AJAX Search API interfaces [66]). SOAP/WSDL based interfaces have proven to be very successful when integrating services with a small to medium client base. REST does not mandate any particular format for how clients should handle messages and errors, which makes it more difficult to integrate clients than for SOAP.

Recommendation: The interface should be specified formally using SOAP/WSDL.

5.3.1.2 Parameter Passing

Both feature datasets and configuration data used by the Schema Transformation Network Service can be very large. In order to enable efficient resource usage, these will all be passed by reference to the service. This will:

- Reduce the overheads of control messages that are sent to the service (which will frequently be persisted when the service is used in an orchestrated manner).
- Enable the service to access data in the most appropriate manner for the underlying transformation engine. This may include delaying the load of the source data; querying the source data in a manner that optimises access patterns; outputting transformed data before all source data has been read.
- Enable greater control over the versioning of configuration data. This will ensure that the configuration can be well maintained and will also enable the transformation engine to perform pre-process optimisations for specific transformation definitions in advance of performing the transformation.

5.3.1.3 Feature Data Access

Source data may be accessed through a download service (see [12]) or any service with one of the interfaces listed below:

- Pre-defined dataset access, by HTTP GET or FTP from a simple URL.
- Direct data access using a WFS interface, with a query specified using filter encoding. Since this is a parameter to the Schema Transformation Network Service (see Input Spatial Data Set parameter to the Transform operation in Section 4.1.3 above), it is up to the client to define the filter required. Where there is insufficient source attribution for queries to differentiate the dataset, multiple WFS endpoints can be used to separate the data in datasets.

The Schema Transformation Network Service will use the related download service standards for these types. If a particular implementation of INSPIRE network services cannot use HTTP, FTP or WFS directly, it can still use the proposed interface by performing the following steps:

1) Using the available resources to download the dataset to local environment,
2) Convert the dataset into a GML 3.1.1 or 3.2.1 document with application schema.
3) Upload GML to a available HTTP or FTP site
4) Invoke Schema Transformation Network Service, passing the URL of the GML document.

5.3.1.4 Configuration Data Access

Configuration data will be stored in an XML repository and be referenced by direct download URL. This should ideally be stored in a managed system, such as an enterprise metadata repository (see also comments at the top of this section regarding the INSPIRE Registry [27]). There are no commonly adopted standards for such systems, although ebXML Registry and Repository [14] and UDDI [15] are often quoted. Most Service-Oriented Architecture (SOA [29]) frameworks appear to write their own systems with only cursory compliance with standards. Services currently available tend to provide direct HTTP access to resources, therefore a simple URL will be used within this interface to avoid tying the system to any specific integration platform.

For details of how the use of the XML Repository was verified during the prototype, please refer to Prototype Report [62] Section 2.2.1.

5.3.1.5 *Synchronicity*

The main 'transform' operation may take a long time to complete, particularly with large data sets or complex mappings. In order to avoid transport layer timeouts in both the server and client implementations, this operation will return asynchronously. WS-Addressing provides the means by with the requester will be informed when the transformation is complete.

All other operations are expected to complete quickly and will therefore be synchronous.

5.4 (External Component) Schema Mapping Designer

This section describes the realization of the Schema Mapping Designer component: its responsibilities, collaborations with other components and rationale for its existence.

Component	**Schema Mapping Designer**
Responsibilities	• Enabling the user to define a mapping from a source schema to the target schema. Persisting this mapping in RIF format. • Validating that the mapping covers all requested elements of the source and target schema.
Collaborations	• The *Transformation Service* will perform transformations based on mappings created by the schema mapping designer. • The *XML Repository* may provide schema definitions to be used by the schema mapping designer (although these can also be provided directly by the Source Datastore if it is implemented using a WFS).
Rationale	• In order to perform transformations, a definition of the required mapping from source to target schema must be provided by the user in RIF format. • While the RIF format is extremely flexible, in enabling all possible transformations, it is structured in a rigorous logical manner which does not map directly onto specific types of transformation. A user interface is required to ensure that the user can use their data modelling expertise to construct conformant mapping definitions.

Table 12 Schema Mapping Designer Component

This needs to provide an interface that is easily accessible by the target users, who will be performing the role of Data Expert, as described in the Use Case Model (Section 4, Use Case View), whilst also permitting access to the full power of the Schema Transformation Network Service. The Schema Mapping Designer could be implemented either as a dedicated RIF mapping definition authoring tool, or as an export functionality in an existing tool. The responsibility for implementing the RIF mapping definition rests, in principle with clients. However, the benefits to tool vendors from undertaking an extension of their software are not difficult to see: a mapping design tool that provides RIF export functionality will be an essential component in the INSPIRE toolkit used by LMOs.

For further details of common transformations that the schema mapping designer should be able to handle, see Appendix E.

5.5 (External Component) XML Repository

This section describes the realization of the XML Repository component: its responsibilities, collaborations with other components and rationale for its existence.

Component	**XML Repository**
Responsibilities	• Provides mechanism for managing schema descriptions and schema mapping descriptions. • Provides access to these by *Transformation Service* component.
Collaborations	• Used by the *Schema Mapping Designer* and *Transformation Service* components.
Rationale	The use of a standard XML Repository (implementing, typically, the OASIS ebXML standard, see [33]) specification for managing the configuration documents relating to transformation services is an alternative to always passing the configuration items directly to the services. It has several advantages: • Configuration is passed by reference, resulting in lower traffic and in many cases better performance. • Enables the *Transformation Service* component to cache the results of transformation specific optimisations, to improve performance of repeated transformations. This is possible due to the unique identifiers for each version of a configuration item. • Enabled efficient version control for registered objects. • Promotes unified understanding of registered objects, since they are accessible from a single location. • Enables and encourages collaborative development. • XML Repository technology offers enhanced storage, classification, querying and data quality tools above and beyond other document storage formats. • Implementations of the XML Repository standard are available under open source licences (for example, freebXML [61]).

Table 13 XML Repository Component

It is currently anticipated that the Schema Transformation Network Service should work with any xml repository that is capable of providing access to registered documents through HTTP GET operations.

Note: Compatibility with XML repository standards (such as ebXML [33]) was investigated during prototyping. For more details, see the Prototype Report [62] Section 2.2.1 and Section 5.3.1.4 of this document.

5.6 (External Component) Source Datastore

This section describes the realization of the Source Datastore component: its responsibilities, collaborations with other components and rationale for its existence.

Component	**<<External>> Source Datastore**
Responsibilities	• Provides access to data that is to be transformed in a standard format.
Collaborations	• Used by the *Transformation Service* component to access data.
Rationale	• Source data needs to be made accessible in a web compatible manner (this is recommended to be GML 3.1.1 or 3.2.1). • The source data is likely to be very large, so it is best handled like this rather than directly in the SOAP message. • The scope of the Schema Transformation Network Service is to transform the logical schema, rather than other aspects of transformation such as coordinate reference system or data format. By providing all input data in a standard format, the service is simplified significantly and this ensures that all source data is web accessible. • There are a number of open source and commercial WFS [34] implementations that could be used as source datastores. Many of these support automatic file format transformation, for example presenting shape files in GML format. The tool used for the Schema Transformation Network Service prototype is Geoserver [59]. • As an alternative, the source datastore could be implemented using a FTP or HTTP site. In this case no query functionality would be supported. • Where WFS is used, queries can be used to divide the data into discrete datasets for transformation. Where there is insufficient source attribution for queries to differentiate the dataset(s), multiple WFS endpoints or layers can be used to separate the data appropriately.

Table 14 Source Datastore Component

5.7 (External Component) Target Datastore

This section describes the realization of the Target Datastore component: its responsibilities, collaborations with other components and rationale for its existence.

Component	**<<External>> Target Datastore**
Responsibilities	• Providing an interface that can be used store the transformed spatial dataset.
Collaborations	• The *Transformation Service* component will write data to this component.
Rationale	• The transformed data could be quite large, so is not generally suitable to be returned directly from the service, particularly when it is used in an orchestrated environment. • WFS-T [35] provides a standard mechanism for creating GML features in a database. It is currently the only open standards' based technology that supports transactional creation of features and exchange of GML over the network. This is likely to be compatible with many download services which are likely to be the next step in the use of the data. • As an bulk-oriented alternative, in conformity with INSPIRE requirements for the mandatory availability of bulk downloads, the target datastore can be implemented using an FTP site. In this case, the transformed dataset will be written to the supplied URL as a single GML document. • The output format will always be one of the INSPIRE schemas, so the Schema Transformation Network Service only needs to support the generation of the range of geometry types used by these schemas. It is a requirement of a Schema Transformation Network Service to document any geometry encodings that it does not support. If there is a difference between those geometry encodings that are supported for reading and writing, separate lists of these may be provided.

Table 15 Target Datastore Component

5.8 (External Component) Service Registry

This section describes the realization of the Service Registry component: its responsibilities, collaborations with other components and rationale for its existence.

Component	**<<External>> Service Registry**
Responsibilities	• Provides a mapping to physical implementations of each service.
Collaborations	• The *Transformation Service* component uses this to find specific machines hosting the source and target datastores and XML Repository.
Rationale	• This is required in order to meet the non-functional requirements of the service. • The Schema Transformation Network Service needs accurate and up to date information about specific endpoints for services with which it interacts. • Note, it is possible to deploy the Service without this component, provided that all Uniform Resource Locators (URL) supplied resolve directly to the required resource for the transformation such that no additional discovery steps are required.

Table 16 Service Registry Component

The availability of the Service Registry component will depend upon wider technical decisions taken in the context of INSPIRE, as it is a resource the use of which should not be limited just to the Schema Transformation Network Service. As mentioned in the above rationale, it is essentially possible to deploy and utilise the Service without this component, but its absence means that no discovery facilities exist to resolve indirect references to resources contained within the source and target datastores and XML Repository.

5.9 (External Component) Client Application

The Schema Transformation Network Service is exposed as a web service and, as such, for clients to engage with the Service there will be a need for some form of client application. However, this is out of scope of this Technical Guidance (note Section 2.2 above) and hence will not be discussed further, except insofar as it is included in the use case realizations shown in Section 5.13. It is sufficient to say that this is standard web service technology and it is not proposed to restrict the options available to implementers, which are many and varied.

5.10 (External Component) Supporting Systems

The Schema Transformation Network Service will depend on supporting systems which may be exclusively used by the Service or shared with other services, depending on factors that are out of scope of this Technical Guidance. For example, these could be relational databases, application servers or other state persistence or runtime environments. In order to fulfil the Gather Technical Information use case (see the use case realization in Section 5.13.3), reference to these systems is required as it helps to explain the use case, which is not fulfilled by an automated process but by administrative intervention by the Service Integrator.

5.11 (External Component) Backoffice Tools

In order to query the supporting systems (see Section 5.10), the Service Integrator needs to use certain back-office tools, the details of which are out of scope of this Technical Guidance. These could be, for example, generic relational database querying tools or web service monitoring tools. Their mention helps to explain the Gather Technical Information use case (see the use case realization in Section 5.13.3).

5.12 Web Service Operations

The following sections detail the operations of the Schema Transformation Network Service web service definition. Each section gives an overview of the operation, technical details of its invocation, input and output parameters and also exception conditions.

5.12.1 WSDL Operation

The WSDL for the Schema Transformation Network Service will provide the implementation of the Get Transformation Service Metadata operation, as defined in the INSPIRE Regulation [3] as amended by [4]. This is part of the Gather Technical Information use case.

This provides all necessary information about the service and describes the service capabilities, including the supported transformation category, supported transformations, accepted input data types, supported logical schema definition and mapping languages.

This operation is required by the INSPIRE Regulation (EC) No. 976/2009 [3] as amended by [4] .

5.12.1.1Technical Detail

Protocol	HTTP GET (WSDL request)
Synchronisation	Synchronous

Table 17 WSDL Operation Protocol and Synchronisation

5.12.1.2Parameters

The full WSDL will be returned as detailed in Appendix A. This includes full WSDL documentation, covering the following aspects of the service as required by the Regulation [3] as amended by [4].

Name	*Role*
ServiceMetadata	The Transformation Service Metadata parameter shall at least contain the INSPIRE metadata elements of the Schema Transformation Network Service.
OperationsMetadata	The Operations Metadata parameter provides metadata about the operations implemented by the Schema Transformation Network Service. It shall describe each operation, including as a minimum a description of the data exchanged and the network address, and shall list the following: The transformation categories accepted by the Transform operation; The encoding for the input Spatial Data Set accepted by the Transform operation; The Logical Schema languages accepted by the Transform operation; this is expressed as a list one or more URLs, each pointing to a GML profile. The Schema Mapping languages accepted by the Transform operation (Fixed value, RIF). The encoding for the output Spatial Data Set generated by the Transform operation; this is expressed as a list of one or more of URLs, each pointing to a GML profile. It is the responsibility of the caller to ensure that the source and target logical schemas provided to the service use a subset of one of the nominated source and target GML profiles.
ResponseLanguage	the natural language used in the Get Transformation Service Metadata response
SupportedLanguages	A list of the natural languages supported by the Schema Transformation Network Service.

Table 18 WSDL Operation Parameters

5.12.2 Transform Operation

This will asynchronously perform the transformation of the source data into the INSPIRE logical schema, based on the supplied transformation specification. Internally, this may include the following steps:

- Obtain logical schema descriptions and mapping descriptions.
- Validate these configuration items.
- Translate and optimise transformation definition for internal engine.
- Obtain source data, transform data and write back data.
- Respond to the client asynchronously.

In addition to the functional aspects of this operation, it must also conform to a number of non-functional aspects, such as scalability and performance. See Section 7 System Qualitiesfor further details of these.

This operation is required by the INSPIRE Regulation (EC) No. 976/2009 [3] as amended by [4].

5.12.2.1Technical Detail

Protocol	SOAP
Synchronisation	Asynchronous

Table 19 Transform Operation Protocol and Synchronisation

5.12.2.2Parameters

All parameters are mandatory. For full details of the valid range of parameters, see the WSDL document in Appendix A**.**

Mode	***Name***	***Role***	***Regulation Equivalent***	***Valid Range***
In	dataset	Indicates one or more source datasets which are to be transformed.	*Input Spatial Data Set*. This has been implemented as a pass-by-reference, rather than a pass by value. See Section 4.5 for justification.	Either a URL, or a download service and filter specification, both of which are defined in the download service technical guidance.
	Mapping	The Schema Mapping parameter shall specify the mapping from the Source Schema to the Target Schema. This is by reference to an XML repository. The logical schema must be contained in the repository in the RIF-PRD [18] format. It will import the GML application schema of the source and target schemas.	*Model Mapping*, *Source Model* and *Target Model*. The model mapping contain references to the source and target models	xsd:anyURI This must be a reference to a logical schema definition stored in an XML repository.
	TargetDataSet	A location in which the transformed data should be stored. The caller is responsible for creating, configuring (and deleting) this dataset.	*Transformation Response.* Using pass by reference allows the client to maintain responsibility for the generated data. See Sections 4.4 and 4.5 for justification.	A reference to a transactional WFS or FTP URL.

Out	TransformedSourceObjects	The number of source objects (features) that were investigated and potentially transformed.	These parameters have no equivalent in the Regulation [3] as amended by [4] although experience with rule-based systems suggests they will be valuable.	xsd:integer
	TransformedTargetObjects	The number of target objects (features) that were created.		xsd:integer
	ObjectErrors	Details of any errors that occurred during processing specific objects, or non-conformances of the object with the data specifications. Each object error will be reported separately.		
	SystemErrors	Details of any system errors, such as failure to connect to the target datastore.		

Table 20 Transform Operation Parameters

Since the communication is asynchronous, WS-Addressing [25] parameters are used to facilitate asynchronous response. Several of these parameters may also require credentials to be passed in order to access external services. Further information about this is provided in Section 7.6.

5.12.2.3Exception conditions

Exception Message	***Condition(s)***
UnsupportedTransformation	The transformation definition is not supported by this Schema Transformation Network Service.
	The transformation requires a built-in operation that is not available in this Schema Transformation Network Service.
ConfigurationError	Unable to access the XML repository.
	Unsupported data format in the XML repository.
DataReadError	Unable to access the source repository, or the source repository does not contain the requested data.
DataWriteError	Unable to write data to the target repository.

Table 21 Transform Operation Exception Conditions

5.12.3 Link Transformation Service Operation

Enables the declaration of availability of a Schema Transformation Network Service for transforming spatial datasets through the member state's Transformation Services while maintaining the transformation capability at the public authority or the third-party location.

See Section 4.1.4 for further information about this operation.

5.12.3.1Technical Detail

Protocol	SOAP
Synchronisation	Synchronous

Table 22 Link Transformation Service Protocol and Synchronisation

5.12.3.2Parameters

Mode	***Name***	***Role***	***Valid Range***
In	request	The Link Transformation Service request parameter shall provide all information about the Public Authority's or Third Party's Transformation Service in conformity with this Regulation, enabling the use of the Public Authority's or Third Party's Transformation Service by the Member State's Transformation Service.	xsd:string

Table 23 Link Transformation Service Parameters

5.12.3.3Exception conditions

There are no special exception conditions prescribed for the Linked Transformation Service operation.

5.13 Use-Case Realizations

This section illustrates how the components work together. Text Formatting is used to highlight the **actors** and ***components*** referenced in each section.

5.13.1 Store Configuration

The following UML communication diagram summarises the key steps required to realize the Store Configuration use case.

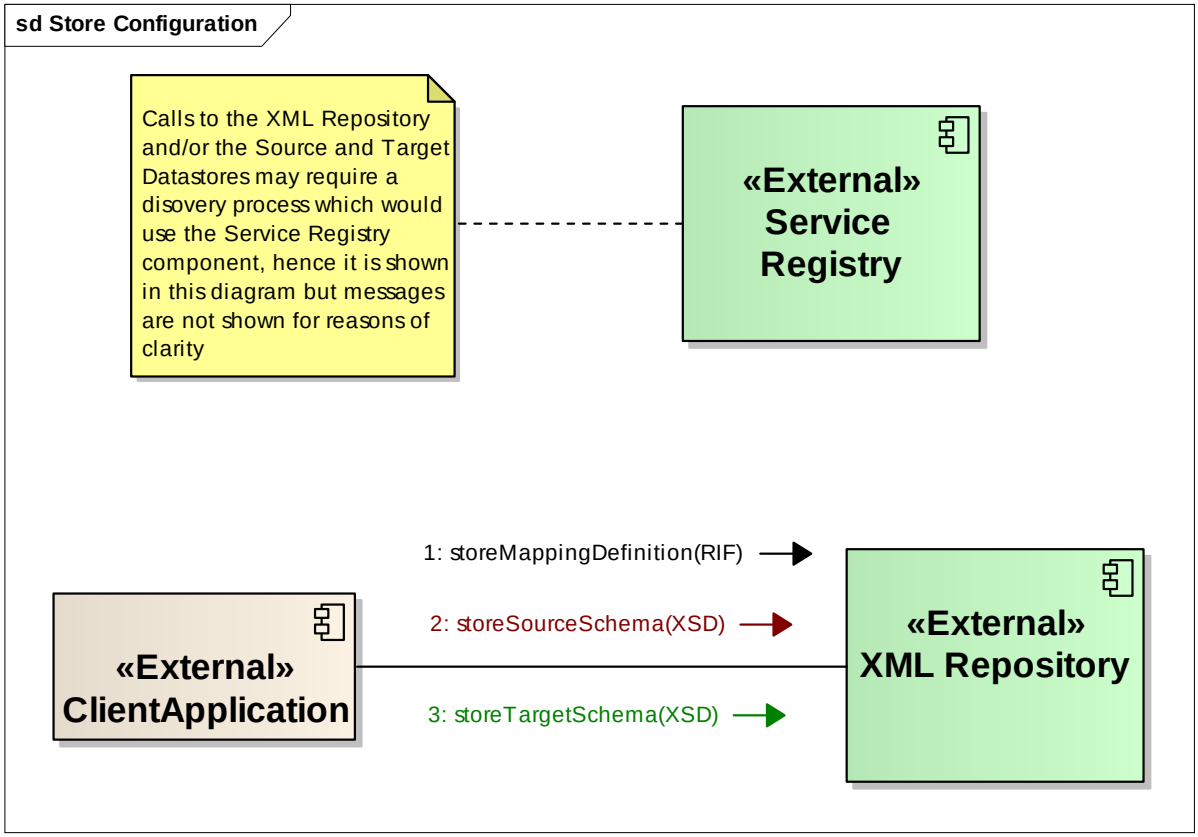

Figure 8 Store Configuration Communication Diagram

In Figure 8, the ***Client Application*** is responsible for sending RIF Mapping Definition and source and target logical schema updates to the ***XML Repository***, which may or may not require use of the ***Service Registry***, depending on the system architecture (see rationale in Section 5.8). Section 4.4 includes a data flow diagram to help explain this process.

5.13.2 Transform Data

The following UML communication diagram summarises the key steps required to realize the Transform Data use case.

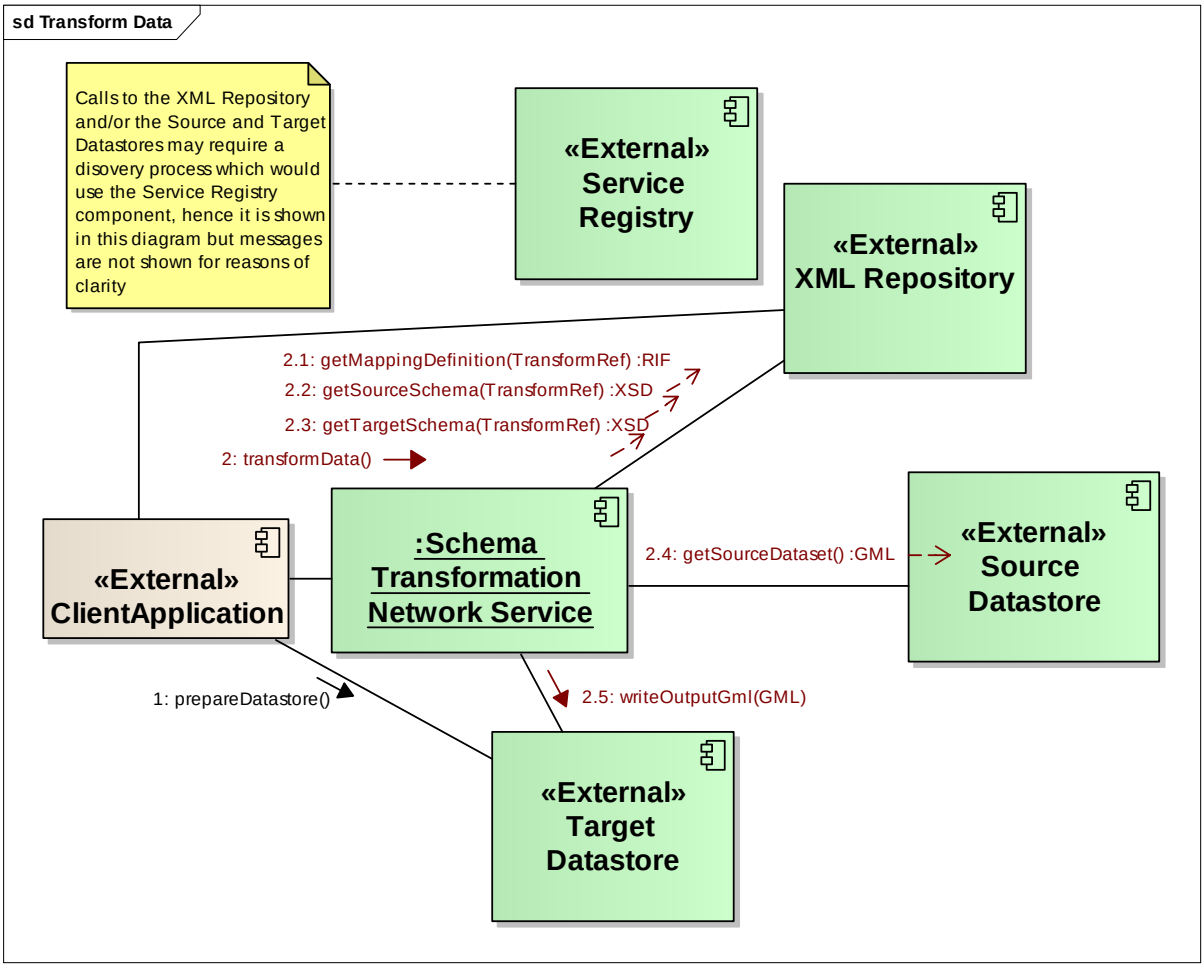

Figure 9 Transform Data Communication Diagram

In Figure 9, in order to perform a transformation, the ***Client Application*** first prepares a ***Target Datastore*** to accept the transformed data. This should be pre-configured with the INSPIRE GML application schema.

The ***Schema Transformation Network Service*** is then called to perform the transformation. It loads the source and target logical schemas and schema transformation definitions from ***the XML Repository***. Source data is loaded from the ***Source Datastore***, transformed and written out to the ***Target Datastore***.

The transformation process involves interpretation of the schema transformation definition document from the neutral interchange format into a format consumable by the chosen transformation engine.

The target data is output to the location corresponding to the abstract Spatial Data Set input parameter to the Transform operation.

Section 4.5 includes a data flow diagram to help explain this process.

5.13.3 Gather Technical Information

The following UML communication diagram summarises the key steps required to realize the Gather Technical Information use case.

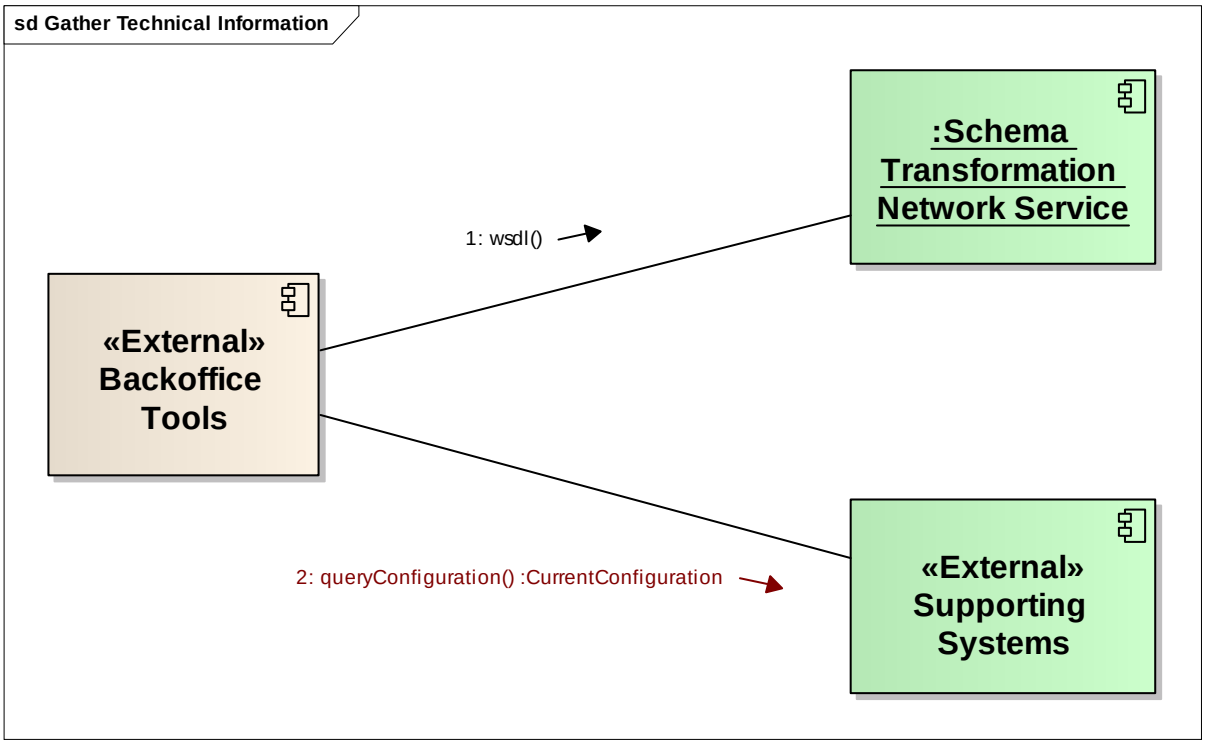

Figure 10 Gather Technical Information Communication Diagram

In Figure 10, the ***Service Integrator*** actor is responsible, with the help of the external ***Backoffice Tools*** component, for querying the state of the Schema Transformation Network Service, and its supporting systems, in order to establish its capabilities. In order to do this, the ***Service Integrator*** will make use of the Service's wsdl() operation as well as other features in the supporting systems which are out of scope of this Technical Guidance, but are referred to abstractly as the queryConfiguration() operation. Section 4.6 includes a data flow diagram to help explain this process.

6. Data View

This section contains further detail regarding the data involved in the system, specifically the schema description and schema mapping configuration items that are used by the Schema Transformation Network Service.

6.1 Rationale for Choice of Schema Description Language

The Schema Transformation Network Service requires a description of the source logical schema in order to aid the user in the construction of a schema mapping, validate the mapping and optimise the transformation process. This only needs to contain the basic structural aspects of the schema, for example:

- Class definitions
- Datatype property definitions
- Association property definitions
- Property cardinalities
- Identification of feature classes
- Class inheritance hierarchy.

For a full description of the levels of transformation which may be encountered by the Schema Transformation Network Service, see the State of the Art Analysis [9], Appendix B.

A key requirement for a schema description language identified by the State Of The Art Analysis [9] is interoperability with the mapping interchange format. Given that this document recommends use of RIF [18] for expressing mapping definitions, this essentially means that the schema description language must be either RDF Schema [36], OWL [37] or XML Schema [38]. As the schema description language is an interchange format, it is important that it is both sufficient for the service and practically usable by service clients, typically by conversion from an existing schema description. An example transformation document is given in Appendix D to this document to demonstrate the compatibility of GML [26] with RIF.

XML Schema is most closely aligned to the needs of the Schema Transformation Network Service, since it is designed for the strict definition of concrete datatypes. It has good support for common data modelling techniques such as structured types containing inner-typed properties and cardinalities, which are not naturally part of RDF Schema or OWL (although can be emulated).

Although UML can be used to express a range of modelling abstractions (including physical models) it cannot be used to express the actual data content. This means that if UML where used as the schema description language, the service would have to perform an internal translation of the UML into an appropriate concrete physical model of the datasets (like XSD). Although this process is tractable, and there are open sources tools that can perform it, this extra level of conversion places unnecessary processing overheads on the service and introduces significant extra development complexity with no substantial benefit. It is also worth noting that even though UML is well standardised, the UML interchange format XMI is used in incompatible ways by tool vendors.

Since the requirement is to support spatial systems and have a method of identifying feature types from complex property types, GML provides useful semantics and common types on top of XML Schema. Therefore, GML should be used as the schema interchange format for the Schema Transformation Network Service.

Where source datasets are initially only available in SHAPE or MAPinfo TAB files, these need to been converted into GML (with appropriate application schema). There are a range of commercial and open source tools that can perform this mapping automatically. As a vendor neutral format, GML is well positioned as an interchange format, and it is likely that tool support will improve with time. Once converted to GML, the datasets can either be provided to the system using a WFS service, or directly using the URL upload. Where data is stored in a live database, access can be provided using a WFS server (again there are several open source and commercial implementations of the WFS interface).

There are some potential issues with GML that need to be considered in the context of the Schema Transformation Network Service:

- GML does not currently support multiple inheritance. This is used in some complex data models, but the majority of source data models investigated so far do not support multiple inheritance. In cases where multiple inheritance is needed with GML serialisation, standard tools will 'copy down' attributes from one parent class to base classes. This system could easily be extended to add further application-specific information regarding the logical schema that is being implemented (in a similar way to the additional information that is supplied with xlinks, see [64]). Future versions of XML Schema will add native support for multiple inheritance.
- The structure of the current XSD encoding of GML is quite verbose for complex properties (requiring separate XML elements for property names and property values). This will impact the RIF mapping definition in terms of additional verbosity, but will not affect usability. Since most users will interact with the mapping definition through some form of user interface, this is not expected to be a problem.

6.2 Schema Mapping Interchange Format

This section describes the format chosen to encode schema mapping definitions and how it is recommended to apply it to the task of INSPIRE schema transformation.

6.2.1 Language Features

A detailed rationale for the selection of Rule Interchange Format (RIF) as a mapping definition language, and an outline of the main characteristics of the language, is given in Appendix C to this document. In addition, Appendix E to this document details numerous common transformation patterns and how they can be expressed using RIF. In the meantime, this section just gives a general overview of the expressive capabilities of the language as related to the INSPIRE schema transformation task.

6.2.2 RIF Normative and Presentation Syntaxes

The W3C RIF Specification [18] describes two syntaxes for RIF:

a) the ***normative*** syntax is XML-based and designed for machine-readability;

b) there is also a ***presentation*** syntax which is not intended to be machine-readable; it is used to present RIF concepts for discussion.

6.2.3 Spatial Extensions

RIF provides support for flexible extension to include new features. The RIF language can be extended so that it would be possible to write a language for describing spatial rules and actions based on RIF (although this is outside the scope of INSPIRE requirements and it is difficult to see where it would be appropriate, given that RIF is design for exchange of business rules rather than for design of complex algorithms) . It is also possible to plug in components written in other languages (such as XML Schema) and to import datatypes defined using XML Schema, OWL and RDF (see [21]).

The most likely approach to integrating spatial functions and predicates to meet INSPIRE requirements would be by defining a library of placeholder spatial functions and predicates, that could be translated into an end-use language implementation as required. Since RIF is not envisaged itself to be used as an end-use language, there is only a need to define the operation signatures and parameter datatypes within such a spatial library. Section 6.2.4 contains a specific recommendation for this.

6.2.3.1 RIF-Geometry dialect

The RIF Overview [38] states:

… it should be possible for motivated experts to define a new RIF dialect as a syntactic extension to an existing RIF dialect, with new elements corresponding to desired additional functionality. These new RIF dialects would be non-standard when defined, but might eventually become standards.

It is possible, for example, to define a geometric dialect of RIF, called, naturally, RIF-Geometry. The route for definition of this new dialect has been described by RIF-FLD (Framework for Logic Dialects, see [17]) and demonstrated by RIF-DTB (DataTypes and Built-ins, see [19]) which was built using RIF-FLD. Such a spatial dialect could contain the signatures and datatypes required to express concepts such as those described in the OGC Simple Features Interface Standard (SFS) [20], including:

- Zero, one, two and three-dimensional point, line and polygon geometries.
- Spatial attributes: isEmpty(), boundary(), isSimple, 3D(), etc.
- Spatial queries: area, equals, touches, overlaps, disjoint, intersects, crosses, within, contains, beyond, etc.
- Spatial analysis: union, intersection, within-distance, min-bounding-box, convex-hull, count-vertices, linear-reference, max-height, centre-point, is-closed, covers, covered-by, etc.
- Generalisation and conflation of geometries.

RIF-FLD (see [17]) defines several types of extension points: symbols, connectives, quantifiers, aggregate functions, and terms. These could be built on to implement new language elements that align precisely with geometric concepts.

6.2.3.2 Black-box approach

An alternative way to include spatial functions in RIF, is the “black box” approach. An XML schema is written externally and imported as a specific namespace into a RIF document. The concepts defined in the schema can be referred to in the rules, without there being any requirement for an implementation. This is a more lightweight route.

The strengths and weaknesses of each extensibility approach are as follows:

New dialect specialising RIF-FLD	***Black-box XML schema import***
Strengths	***Strengths***
Precise logical syntax and semantics	Quick and easy to write the schema
Easier to communicate and standardise	Enables implementation in any language
	Possible highly performant implementations
Weaknesses	***Weaknesses***
Could be intellectually challenging	Less mathematical rigour in definition

Could perform poorly in a rules engine	

Table 24 Comparison of Methods of Including Spatial Functions in RIF

A generic set of spatial functions and predicates for INSPIRE style transformations was developed during the prototyping phase to enable interoperability with common spatial operations and this forms the basis of the recommended approach for the Schema Transformation Network Service. See the following Section 6.2.4 for a key recommendation in this respect. Appendix F to this document also contains additional information about this.

6.2.4 Spatial Functions and Predicates for INSPIRE Transformations within RIF-PRD

This section describes the approach that we recommend should be used for the encoding of references to spatial operations within normative-syntax RIF-PRD documents. This applies to the expression of transformations upon data instances such that they involve operations on geometric attributes. Since there is no existing standard for such RIF expressions, it is proposed that we take the OGC Simple Feature Access Specification version 1.2 as the basis for defining a standardised approached to use of class and method name IRIs, since this is the recommended standard for geometric data types and definitions referred to in the INSPIRE data specifications (see e.g. Requirement 7 of the INSPIRE Data Specification Hydrography, [68]).

The Extended Backus-Naur Form (EBNF, see [67]) is a well-known meta-syntax for defining context-free grammars and is used in many computing contexts. For the TNS Technical Guidance, the EBNF production for `<iri>` will be:

```
<iri> ::='http://www.opengeospatial.org/standards/sfa'
          '/' <interface-name> ['#' <method-name> ]
```

This definition depends on certain non-terminals defined elsewhere:

`interface-name` : interface name.

`method-name` : method name.

The interface name will be that of the interface type for any given geometry, as this ensures that conflict is avoided when multiple classes implement the same method.

The list of supported operations for the Schema Transformation Network Service prototype is given in Appendix F to this document. This provides an illustration of the approach proposed in this section.

Recommendation: when referring to spatial functions and predicates within a RIF document, use the OGC Simple Feature Access Specification version 1.2 as the basis for identifying and naming function and predicate placeholders.

6.2.5 RIF Usage Guidelines

RIF is intended to be used by mapping designer tool authors such that the design user interface is capable of generating semantically and syntactically valid RIF documents at the conclusion of the mapping definition process. This section and Appendix E together describe ways RIF can be used to capture mapping conditions and transformation actions, and explore the expressiveness inherent in the language that makes it suitable for this purpose.

We *recommend* that RIF be used in these ways:

- To define production rules for mapping attributes from source to target feature instances.
- To create target objects based on simple classifications derived from analysis of source instance enumeration-type attributes.
- To define complex classifications of conceptual types based on analysis of source data, and iterate over those types to populate target schema.

We *do not* advise that RIF be used in the following ways:

- Implement spatial or other utility functions (beyond defining operation signatures and parameter datatypes).

Some common patterns that can be applied to facilitate RIF rule authoring are:

If…Then…Do

This is the basic pattern that governs imperative processing of a transformation.

```
If
(  And
   (  (* logical conditions *)
      Or
      (
         (* logical conditions *)
      )
   )
)
Then
(  Do
   (
      (* actions *)
   )
)
```

The `If` clause contains the set of mapping conditions that must hold true in order to proceed to apply the actions in the `Then` clause. Note that `And` and `Or` clauses are used as part of the conditions. The `Do` clause contains the actions that effect the production of the target output.

Assert / Modify / Retract

These actions correspond to the SQL statements INSERT, UPDATE and DELETE.

```
(* create new tn-ra:RailwayStationNode *)
Do
(  Assert
   (  ?iId New()
      ?iId #
      ?iId[base:localId->xsd:string(?featureId) base:namespace->"AN.LMO"^^xsd:string]
   )
```

```
)
```

In this example, the transformation is producing a new InspireId object and populating its localId and namespace attributes with data; in the first case, with a variable value (via a cast), and in the second case, a string literal constant.

Frames

Objects and attribute assignments can be expressed using **frame** constructs, e.g.

```
?line[src:featId->?featureId src:survey->?surveyDate src:geometry->?lineGeom]
```

The frame declaration starts with the variable `?line`, then left and right square brackets define the space for making statements about the contents (attribution) of the object. This example predicates that the variable `?line` has (at least) `featId`, `survey` and `geometry` attributes which are assigned the variables `?featureId`, `?surveyDate` and `?lineGeom`, respectively. Since this statement is evaluated and must be true to continue processing, we can thereafter use the variables as labels to refer to the attributes of the object when needing to pass those attributes into the target schema. Frame logic is very powerful in its ability to express object concepts and, for this reason, has been proposed as a formal foundation for object-oriented programming languages (see, for example, the paper available at [22]).

For further details, see the RIF-PRD Specification at [18].

Appendix E explores numerous categories of transformation and gives examples of RIF mapping conditions and transformation actions to assist mapping design tool authors to appreciate the functionality required of such a tool.

6.3 Schema Mapping Definition Process (Supplementary)

Please refer to the Prototype Report Section 2.3 for a description of the approach used to define schema mappings during the prototyping of the INSPIRE Schema Transformation Network Service.

7. System Qualities

This section details system qualities that must be considered during implementation or design of a Schema Transformation Network Service.

7.1 Interoperability and Vendor Neutrality

A key goal of the INSPIRE Schema Transformation Network Service interface is to provide a standardised interface to permit interoperability with clients and enable multiple vendors to implement the Schema Transformation Network Service.

To achieve this goal, any implementation specific or proprietary languages have been avoided in the definition of the data interchange formats. It is anticipated that these will need to be extended in many circumstances, for example by providing custom user defined operations. This should always be performed using the extension points documented in this report and be fully documented.

7.2 Extensibility

The expressiveness of the RIF mapping definition language can be supplemented by import of externally defined spatial functions which do not need to be written in RIF itself. See Section 6.2.3 above for details of the extension points.

7.3 Capacity

The Schema Transformation Network Service is required by the INSPIRE Regulation [3] as amended by [4] to support a minimum of 5 simultaneous requests.

In order to achieve this, the Schema Transformation Network Service must be based upon an appropriate implementation platform that permits multiple, parallel invocations and therefore interactions need to be coordinated on a secure and transactional foundation.

7.4 Availability

To meet the INSPIRE Regulation [3] as amended by [4], the probability of a network service to be available shall be 99% of the time.

The implementation of the Schema Transformation Network Service must take appropriate actions to ensure that this is possible. This must include consideration of:

- Failover for processing nodes that encounter systemic challenges (e.g. hardware outages)
- Dynamic provisioning to meet increased resource demands (if system users make simultaneous, taxing demands on system resources by seeking to transform large dataset requiring lengthy and memory-hungry transformations).

7.5 Performance

For a description of the performance analysis aspects of the the INSPIRE Schema Transformation Network Service prototype, please refer to the Prototype Report [62] Section 2.7.

7.6 Handling of Credentials

The Schema Transformation Network Service must access external download services and XML repositories. These may require some form of access control, such as passing of an authentication token or credentials.

The INSPIRE Initial Operating Capability Task Force (IOC TF) [23] is currently investigated related issues. Once a proposed solution to this is endorsed, it should be followed for this service.

7.7 Error Policy

Schema transformation definitions are complex documents. It would be relatively simple to construct an invalid transformation, for example referring to attributes or classes that do not exist, or using operations that have not been defined on the source schema.

The Schema Transformation Network Service must ensure that all inputs are validated thoroughly and all pre-requisites tested (for example, access to required data and services) before performing a large amount of transformation.

Even after validating the prerequisites, it is possible that errors will occur. This is because specific features may have combinations of attributes that are not correctly handled by the mapping definition. This may also occur because a built-in function unexpectedly returns a value that is invalid in some way. When this kind of feature related error occurs, the service should continue to process the remaining features and report these errors in the call-back message.

7.8 Reliability, Security, Quality of Service, Rights Management, etc. (Supplementary)

Production implementations of the Schema Transformation Network Service may need to support a number of additional INSPIRE requirements. These could include topics such as security, reliability, rights management, etc.

These issues are currently being investigated by the INSPIRE IOC TC [23] and therefore are not yet covered by this Technical Guidance. Many industry standards relating to these features are implemented using specific SOAP headers, so it is expected that these should fit into the Transformation Service interface defined here with relative simplicity.

Schema Transformation Network Service vendors are advised to check the latest guidance from INSPIRE regarding this work.

7.9 Testing Policy (Supplementary)

For a description of the testing policy adopted for the INSPIRE Schema Transformation Network Service prototype, please refer to the Prototype Report [62] Section 2.4.3.

8. Implementation View (Supplementary)

The Schema Transformation Network Service interface defined in this technical guidance has been designed such that it may be implemented by a variety of types of organisation whilst still being sufficiently flexible to support INSPIRE schema transformations.

This section aims to provide practical advice on how existing data manipulation engines may be used to implement the Schema Transformation Network Service. For Schema Transformation Network Service users, this will ensure that costs are reduced and reliability is increased by using tried and tested technology.

The State Of The Art Analysis [9] identified that most existing or prototyped data manipulation engines use a proprietary language to specify the operation of their tools. The tool vendor survey identified that most tools were capable of performing the most complex of the identified categories of schema transformation. Therefore, it should be theoretically possible to represent complex schema transformations in a number of different tools.

A basic implementation of a Schema Transformation Network Service will require:

1. A web service interface.
2. A data manipulation engine, to perform the transformations.
3. Spatial processing and support capabilities.
4. A configuration parser, to translate between the tool's proprietary configuration language and the generic interchange format used by the INSPIRE service.
5. A user interface to define configuration, either directly in the RIF format or with functionality to export to RIF.

Further details of each of these will be covered in the following sections.

8.1 Web Service Interface

LMOs requiring to make their data available in INSPIRE format, or third-parties engaged by them to assist with implementing INSPIRE, or other intermediate bodies such as national agencies, will need to develop a simple web service, following the interface defined in this document. This will utilise the components described in Sections 5.3 to 5.8 above to perform the transformation operations.

8.2 Data Manipulation Engine

This should be capable of loading data, performing transformations and storing data.

The following must be considered when selecting or modifying a data manipulation engine:

- Ability to perform all operations (directly or indirectly) equivalent to those required by the rule interchange language and extensions defined within this document
- Ability to meet the extensibility, capacity, availability and performance requirements based on the guidance given in Section 7.

8.3 Spatial Process and Support

The system should be capable of processing spatial data in complex and richly expressive structures, using standards-based technology, and executing sophisticated spatial analytical operations, in a scalable and performant manner.

8.4 Configuration Parser

The configuration parser is responsible for transforming a transformation definition in the RIF format into the internal configuration format required by the data manipulation engine.

The details of configuration parser depend on the selected data manipulation engine, but it may be as simple as a stylesheet application.

The configuration parser may additionally perform optimisations of the transformation process, to take advantage of any specific features of the data manipulation engine. It is important to note that the RIF configuration is merely an interchange format – it does not specify any particular implementation approach.

8.5 User Interface

This could be any tool that provides a graphical representation of source and target schemas and allows for the expression of mappings between the respective feature classes and attributes. The tool should permit export of the mapping to RIF, although RIF does not need to be its native storage format.

9. Deployment View (Supplementary)

Options for the deployment of the INSPIRE Schema Transformation Network Service depend on the business use cases to which the Service will be applied. Options are likely to include:

- Local deployment at the data provider's location
- Deployment in an external service provider
- Deployment in an INSPIRE consumer location.

This Technical Guidance is unable to provide prescriptions for how to deploy the Service that will meet every possible scenario which may be encountered in production environments. However, the Prototype Report [62] Section 2.6 describes the deployment of the prototype Service, the factors that were considered, and the result of their evaluation. It is hoped that the information in that document will assist implementers in resolving the challenges specific to their environment and business needs.

Appendix A: Interface Specifications

The full Schema Transformation Network Service web service description is included below. This document is encoded in Web Service Description Language (WSDL) (see [65]).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://inspire.jrc.ec.europa.eu/SchemaTransformation/v0-2/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   name="INSPIRE-Schema-TNS" xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:filter="http://www.opengis.net/ogc" xmlns:rif="http://www.w3.org/2007/rif#"
   xmlns:wsa="http://www.w3.org/2005/08/addressing"
   xmlns:owl2="http://www.w3.org/2002/07/owl#"
   targetNamespace="http://inspire.jrc.ec.europa.eu/SchemaTransformation/v0-2/">
   <wsdl:types>
      <xsd:schema
         targetNamespace="http://inspire.jrc.ec.europa.eu/SchemaTransformation/v0-2/">
         <xsd:import namespace="http://www.w3.org/2007/rif#"
            schemaLocation="PRDRule.xsd" />
         <xsd:import schemaLocation="ws-addr.xsd"
            namespace="http://www.w3.org/2005/08/addressing" />
         <xsd:element name="TransformRequest">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="sourceDataset" type="tns:dataset"
                     maxOccurs="unbounded">
                     <xsd:annotation>
                        <xsd:documentation>
                           This is a list of one or more source datasets
                           to be processed by the transformation service.
                        </xsd:documentation>
                     </xsd:annotation>
                  </xsd:element>
                  <xsd:element name="mapping" type="tns:mapping">
                     <xsd:annotation>
                        <xsd:documentation>This is a description of the transformation
                           that is to be performed. This should be encoded in the rule
                           interchange format. It may either be defined inline or by reference
                           to an external location (e.g. xml_repository).
                        </xsd:documentation>
                     </xsd:annotation>
                  </xsd:element>
                  <xsd:element name="targetDataset" type="tns:dataset">
                     <xsd:annotation>
                        <xsd:documentation>Details of where the transformed data should
                           be placed. This should be a reference to a pre-configured WFS
                           that supports WFS 1.1 transactional operations (Create).
                        </xsd:documentation>
                     </xsd:annotation>
                  </xsd:element>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
         <xsd:complexType name="dataset">
            <xsd:choice>
```

```
<xsd:element name="directDownload">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="url" type="xsd:anyURI">
            <xsd:annotation>
               <xsd:documentation>A location at which a GML file may be
                  downloaded directly.</xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="defaultCRS" type="xsd:anyURI">
            <xsd:annotation>
               <xsd:documentation>The default CRS of the given file.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="schema" type="xsd:anyURI"
            minOccurs="0">
            <xsd:annotation>
               <xsd:documentation>The location of the application schema.
                  This is only mandatory for the target dataset.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
<xsd:element name="WFSReference">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="capabilitiesURL" type="xsd:anyURI" />
         <xsd:element name="layer" type="xsd:string"
            maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation>A list of layers that will be loaded
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="maxFeatures" type="xsd:long"
               minOccurs="0">
            <xsd:annotation>
               <xsd:documentation>Optional maximum number of features that
                  should be loaded from the WFS. A value of -1 implies no
                  upper limit on the number of features that are loaded.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="filter" type="xsd:string"
               minOccurs="0">
            <xsd:annotation>
               <xsd:documentation>Optional filter to be passed to the WFS.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

```
            </xsd:choice>
         </xsd:complexType>
         <xsd:complexType name="mapping">
            <xsd:choice>
               <xsd:element name="directDownload" type="xsd:anyURI">
                  <xsd:annotation>
                     <xsd:documentation>A location at which the document may be
                        downloaded directly. This will typically be a location within an
                        XML repository.</xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
               <xsd:element ref="rif:Document" />
            </xsd:choice>
         </xsd:complexType>
         <xsd:element name="LinkTransformationService">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="request" type="xsd:string">
                     <xsd:annotation>
                        <xsd:documentation>If the request parameter supplied contains a non-
                           null URL, the service will forward all future requests to the
                           specified delegate (except for future calls to this operation).
                           Conversely, if the request parameter is null, the forwarding is
                           cancelled.  Note that, because this service accesses resources
                           using pass-by-reference,  to be successful the delegate service
                           must have access to any specified resources.
                        </xsd:documentation>
                     </xsd:annotation>
                  </xsd:element>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="TransformationComplete">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="transformedSourceObjects" type="xsd:integer">
                     <xsd:annotation>
                        <xsd:documentation>The number of source objects that were
                           investigated and potentially transformed</xsd:documentation>
                     </xsd:annotation>
                  </xsd:element>
                  <xsd:element name="targetObjects" type="xsd:integer">
                     <xsd:annotation>
                        <xsd:documentation>The number of target objects that were
                           created</xsd:documentation>
                     </xsd:annotation>
                  </xsd:element>
                  <xsd:element name="objectError" minOccurs="0"
                     maxOccurs="unbounded">
                     <xsd:annotation>
                        <xsd:documentation>Details of any errors that occurred during
                           processing specific objects, or non-conformances of the object
                           with the data specifications.</xsd:documentation>
                     </xsd:annotation>
                     <xsd:complexType>
                        <xsd:simpleContent>
```

```
                        <xsd:extension base="xsd:string">
                           <xsd:attribute name="objectId" type="xsd:string" />
                        </xsd:extension>
                     </xsd:simpleContent>
                  </xsd:complexType>
               </xsd:element>
               <xsd:element name="systemError" minOccurs="0"
                  maxOccurs="unbounded" type="xsd:string">
                  <xsd:annotation>
                     <xsd:documentation>Details of any system errors that occurred
                        during the processing, such as failure to connect to the target
                        datastore. These should explain the current state of the
                        system, for example if any output features were written or not.
                     </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:schema>
</wsdl:types>
<wsdl:message name="TransformRequestRequest">
   <wsdl:part element="tns:TransformRequest" name="parameters" />
</wsdl:message>
<wsdl:message name="LinkTransformationServiceRequest">
   <wsdl:part name="parameters" element="tns:LinkTransformationService"></wsdl:part>
</wsdl:message>
<wsdl:message name="TransformationCompleteRequest">
   <wsdl:part name="parameters" element="tns:TransformationComplete"></wsdl:part>
</wsdl:message>
<!-- WS-Addressing call back messages -->
<wsdl:message name="WSARelatesToHeader">
   <wsdl:part name="RelatesTo" element="wsa:RelatesTo" />
</wsdl:message>
<wsdl:message name="WSAMessageIDHeader">
   <wsdl:part name="MessageID" element="wsa:MessageID" />
</wsdl:message>
<wsdl:message name="WSAReplyToHeader">
   <wsdl:part name="ReplyTo" element="wsa:ReplyTo" />
</wsdl:message>
<wsdl:portType name="INSPIRE-Schema-TNS">
   <wsdl:documentation>This is the main port for INSPIRE Schema Transformation Network
      Services.</wsdl:documentation>
   <wsdl:operation name="TransformRequest">
      <wsdl:documentation>This will asynchronously perform the transformation of the source
         data into the inspire format, based on the supplied transformation specification.
         For further details of the requirements of this operation, please see
         the INSPIRE Schema Transformation Services Technical Guidance document.
      </wsdl:documentation>
      <wsdl:input message="tns:TransformRequestRequest" />
   </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="INSPIRE-Schema-TNS-Response">
   <wsdl:documentation>This is the port for asynchronous responses to the
      INSPIRE transformation network services.</wsdl:documentation>
   <wsdl:operation name="TransformationComplete">
```

```
         <wsdl:input message="tns:TransformationCompleteRequest"></wsdl:input>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="INSPIRE-Schema-TNSSOAP" type="tns:INSPIRE-Schema-TNS">
      <soap:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="TransformRequest">
         <soap:operation
            soapAction="http://inspire.jrc.ec.europa.eu/SchemaTransformation/v0-
1/TransformRequest" />
         <wsdl:input>
            <soap:body use="literal" />
            <soap:header message="tns:WSAReplyToHeader" part="ReplyTo"
               use="literal" encodingStyle="" />
            <soap:header message="tns:WSAMessageIDHeader" part="MessageID"
               use="literal" encodingStyle="" />
         </wsdl:input>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:binding name="INSPIRE-Schema-TNSCallbackSOAP" type="tns:INSPIRE-Schema-TNS-Response">
      <soap:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="TransformationComplete">
         <soap:operation
            soapAction="http://inspire.jrc.ec.europa.eu/SchemaTransformation/v0-
1/TransformationComplete" />
         <wsdl:input>
            <soap:body use="literal" />
            <soap:header message="tns:WSARelatesToHeader" part="RelatesTo"
               use="literal" encodingStyle="" />
         </wsdl:input>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="INSPIRE-Schema-TNS">
      <wsdl:port binding="tns:INSPIRE-Schema-TNSSOAP" name="INSPIRE-Schema-TNSSOAP">
         <soap:address
            location="http://inspire.jrc.ec.europa.eu/SchemaTransformation/v0-2/" />
      </wsdl:port>
   </wsdl:service>
   <wsdl:service name="INSPIRE-Schema-TNS-Callback">
      <wsdl:port binding="tns:INSPIRE-Schema-TNSCallbackSOAP"
         name="INSPIRE-Schema-TNSCallbackSOAP">
         <soap:address
            location="http://inspire.jrc.ec.europa.eu/SchemaTransformation/v0-2/callback" />
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Appendix B: Sample SOAP Requests

The following sample SOAP request, schema description document and schema mapping document are based on real examples from the prototyping exercise.

Sample Transform Operation Request Message

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <wsa:ReplyTo xmlns:tns="http://inspire.jrc.ec.europa.eu/SchemaTransformation/v0-2/"
xmlns:rif="http://www.w3.org/2007/rif#" xmlns:wsa="http://www.w3.org/2005/08/addressing">
            <wsa:Address>http://lslbvm41:44369/TNS/callbackport</wsa:Address>
        </wsa:ReplyTo>
        <wsa:MessageID xmlns:tns="http://inspire.jrc.ec.europa.eu/SchemaTransformation/v0-2/"
xmlns:rif="http://www.w3.org/2007/rif#"
xmlns:wsa="http://www.w3.org/2005/08/addressing">a161d586-8e0e-47b2-933c-
31f16f5acb34</wsa:MessageID>
    </soap:Header>
    <soap:Body>
        <tns:TransformRequest
xmlns:tns="http://inspire.jrc.ec.europa.eu/SchemaTransformation/v0-2/"
xmlns:rif="http://www.w3.org/2007/rif#" xmlns:wsa="http://www.w3.org/2005/08/addressing">
            <sourceDataset>
                <WFSReference>
                    <capabilitiesURL>http://geoserver:8080/tnstg-
geoserver/ows?service=WFS&amp;version=1.0.0&amp;request=GetCapabilities</capabilitiesURL>
                    <layer>dcp:Perceelvlak</layer>
                    <maxFeatures>-1</maxFeatures>
                </WFSReference>
            </sourceDataset>
            <mapping>
               <directDownload>
                  file:/home/jrc/rif/digest/dutch cadastral parcel.rif
               </directDownload>
            </mapping>
            <targetDataset>
                <directDownload>
                    <url>/tmp/gml/dutch.xml</url>
                    <schema>/XSD/CadastralParcels.xsd</schema>
                </directDownload>
            </targetDataset>
        </tns:TransformRequest>
    </soap:Body>
</soap:Envelope>
```

Sample Schema Description Document

The following is a sample source logical schema that describes Dutch Cadastral Parcel data.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:bcp="http://jrc.onespatial.com/cp/belgium"
xmlns:dcp="http://jrc.onespatial.com/cp/dutch" xmlns:gml="http://www.opengis.net/gml"
xmlns:nhy="http://jrc.onespatial.com/hy/norwegian"
xmlns:nitn="http://jrc.onespatial.com/tn/lpsni"
xmlns:outline="http://jrc.onespatial.com/outline"
xmlns:shy="http://jrc.onespatial.com/hy/swedish" xmlns:sitn="http://jrc.onespatial.com/tn/osi"
elementFormDefault="qualified" targetNamespace="http://jrc.onespatial.com/cp/dutch">
  <xsd:import namespace="http://www.opengis.net/gml"
schemaLocation="http://geoserver:8080/tnstg-geoserver/schemas/gml/3.1.1/base/gml.xsd"/>
  <xsd:complexType name="PerceelvlakType">
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element maxOccurs="1" minOccurs="0" name="the geom" nillable="true"
type="gml:MultiSurfacePropertyType"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="MI PRINX" nillable="true"
type="xsd:double"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="PCVL VESTI" nillable="true"
type="xsd:string"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="PCVL PRCL " nillable="true"
type="xsd:string"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="PCVL DATUM" nillable="true"
type="xsd:dateTime"/>
          <xsd:element maxOccurs="1" minOccurs="0" name="PCVL DAT00" nillable="true"
type="xsd:dateTime"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="Perceelvlak" substitutionGroup="gml: Feature"
type="dcp:PerceelvlakType"/>
</xsd:schema>
```

Sample Mapping Document

The following is a sample RIF mapping definition document that specifies how to transform a source dataset (containing Dutch Cadastral Parcel data) into the INSPIRE Cadastral Parcels format.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Document
   xmlns="http://www.w3.org/2007/rif#"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:gml="http://www.opengis.net/gml/"
   xmlns:omwg="http://www.omwg.org/TR/d7/ontology/alignment"
   xmlns:goml="http://www.esdi-humboldt.eu/goml"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <payload>
    <Group>
      <sentence>
        <Implies>
          <if>
            <Exists>
              <declare>
                <Var>dutch instance</Var>
              </declare>
              <declare>
                <Var>dutch pcvl prcl</Var>
              </declare>
               <declare>
                <Var>dutch pcvl datum</Var>
              </declare>
              <declare>
                <Var>dutch pcvl dat00</Var>
              </declare>
               <declare>
                <Var>dutch geometry</Var>
              </declare>
              <formula>
                <And>
                  <formula>
                    <Member>
                      <instance>
                        <Var>dutch instance</Var>
                      </instance>
                      <class>
                        <Const
type="rif:iri">http://jrc.onespatial.com/cp/dutch:Perceelvlak</Const>
                      </class>
                    </Member>
                  </formula>
                  <formula>
                    <Frame>
                      <object>
                        <Var>dutch_instance</Var>
                      </object>
                      <slot ordered="yes">
                        <Const
type="rif:iri">http://jrc.onespatial.com/cp/dutch:PCVL_PRCL_</Const>
```

```
                    <Var>dutch pcvl prcl</Var>
                  </slot>
                  <slot ordered="yes">
                    <Const type="rif:iri">
                       http://jrc.onespatial.com/cp/dutch:PCVL DATUM
                    </Const>
                    <Var>dutch pcvl datum</Var>
                  </slot>
                  <slot ordered="yes">
                    <Const type="rif:iri">
                       http://jrc.onespatial.com/cp/dutch:PCVL DAT00
                    </Const>
                    <Var>dutch pcvl dat00</Var>
                  </slot>
                  <slot ordered="yes">
                    <Const type="rif:iri">
                       http://jrc.onespatial.com/cp/dutch:the geom
                    </Const>
                    <Var>dutch geometry</Var>
                  </slot>
                </Frame>
              </formula>
            </And>
          </formula>
        </Exists>
      </if>
      <then>
        <Do>
          <actionVar>
            <Var>inspire_instance</Var>
            <New/>
          </actionVar>
           <actionVar>
            <Var>inspire_id</Var>
            <Frame>
              <object>
                <Var>inspire_instance</Var>
              </object>
              <slot ordered="yes">
                <Const type="rif:iri">
                    urn:x-inspire:specification:gmlas:CadastralParcels:3.0:inspireId
                 </Const>
                <Var>inspire_id</Var>
              </slot>
            </Frame>
          </actionVar>
          <actions ordered="yes">
            <Assert>
              <target>
                <Member>
                  <instance>
                    <Var>inspire instance</Var>
                  </instance>
                  <class>
                    <Const type="rif:iri">
                    urn:x-inspire:specification:gmlas:CadastralParcels:3.0:CadastralParcel
```

```
                    </Const>
                  </class>
                </Member>
              </target>
            </Assert>
            <Assert>
              <target>
                <Frame>
                  <object>
                    <Var>inspire id</Var>
                  </object>
                  <slot ordered="yes">
                    <Const type="rif:iri">
          urn:x-inspire:specification:gmlas:BaseTypes:3.2:localId</Const>
                    <Var>dutch pcvl prcl</Var>
                  </slot>
                  <slot ordered="yes">
                    <Const type="rif:iri">
          urn:x-inspire:specification:gmlas:BaseTypes:3.2:namespace</Const>
                    <Const type="http://www.w3.org/2001/XMLSchema#string">
                      NL.KAD.CP
                    </Const>
                  </slot>
                </Frame>
              </target>
            </Assert>
            <Assert>
              <target>
                <Frame>
                  <object>
                    <Var>inspire_instance</Var>
                  </object>
                  <slot ordered="yes">
                    <Const type="rif:iri">
          urn:x-inspire:specification:gmlas:CadastralParcels:3.0:geometry</Const>
                    <Var>dutch_geometry</Var>
                  </slot>
                   <slot ordered="yes">
                    <Const type="rif:iri">CadastralParcels:label</Const>
                    <Var>dutch_pcvl_prcl</Var>
                  </slot>
                  <slot ordered="yes">
                    <Const type="rif:iri">
  urn:x-inspire:specification:gmlas:CadastralParcels:3.0:nationalCadastralReference
                    </Const>
                    <Var>dutch_pcvl_prcl</Var>
                  </slot>
              <slot ordered="yes">
                    <Const type="rif:iri">
          urn:x-inspire:specification:gmlas:CadastralParcels:3.0:validFrom
                    </Const>
                    <Var>dutch pcvl datum</Var>
                  </slot>
              <slot ordered="yes">
                    <Const type="rif:iri">
          urn:x-inspire:specification:gmlas:CadastralParcels:3.0:validTo
```

```
                    </Const>
                     <Var>dutch pcvl dat00</Var>
                   </slot>
                <slot ordered="yes">
                     <Const type="rif:iri">
       urn:x-inspire:specification:gmlas:CadastralParcels:3.0:beginLifespanVersion
                    </Const>
                     <Var>dutch pcvl datum</Var>
                   </slot>
                 </Frame>
               </target>
             </Assert>
           </actions>
         </Do>
       </then>
     </Implies>
   </sentence>
  </Group>
 </payload>
</Document>
```

Appendix C: Rationale for use of Rule Interchange Format

This section describes, in detail, the reasons why RIF was selected for the mapping definition format, and why other candidate formats were rejected for this purpose.

Requirement for a Mapping Interchange Format

The domain of solutions that handle schema transformation contains a large number of mapping languages. Some of these are specific to systems engineering models (such as MOF [40] -based languages like ATL [41] and QVT [42]) and thus designed primarily for assisting engineers in understanding and communicating modelling concepts. Others are designed primarily for the specialised requirements of XML processing (such as XSLT [43]). However, no accepted standard exists for mapping interchange in the same way as UML [31] is a standard for defining system engineering models, or XML Schema [37] is for describing XML documents.

Schema mapping takes instance data from a given location in a source schema and transposes it into a corresponding location in a target schema. A number of transformation "levels" have been defined as part of the State of the Art Analysis [9] (see Appendix B to the document). Correspondences between source and target locations are defined using rules. Therefore, the essential requirement of a schema mapping language is that it be capable of defining rules and rulesets. It is, in addition, necessary for such a language to be able to define actions that produce target data on the basis of predicates and functions applied to the source data. It is also necessary for the language to be able to identify and make assertions about instance data (i.e. objects), and not just on the level of classes and their attributes.

Mapping Transfer Language *versus* End-Use Language

Even with these qualifications on the role of a schema mapping language, there are still many candidate languages that could be used. In addition to languages, many tools that are currently distributed (either commercially or under open source licences) use their own, internally-defined mapping formats. Therefore, a realistic solution to the challenge of schema transformation in a heterogeneous environment like INSPIRE must involve enlistment of a mapping transfer language. Interchange of mappings enables the concepts essential to the schema mapping to be distilled into a precise, machine-readable format that can be validated, exchanged, stored and passed freely about the INSPIRE community.

This approach facilitates inclusion of diverse solutions into a common system. The benefits of such an approach are analogous to those inherent in the design of the internet. With the internet, the key feature of the architecture is that no single network host is essential to the viability of the whole. To apply this analogy, we could say that each "host" is a language format, and that a good architecture would be one in which no single language has to be *both* used for storage and exchange of mappings *and* applied at the point of end-use. Adoption of a single mapping transfer language, whilst enabling the end-use language to be implemented in any compliant way, decouples the mapping definitions from any single implementation format, promotes interoperability and discourages vendor lock-in.

The only requirement that must apply to an end-use language is that it be possible unambiguously to translate to it from the mapping transfer language at the point of end-use, i.e. when running a specific transformation. This translation must be semantics-preserving so that mapping definitions are able to be used across multiple, heterogeneous systems.

Selection of Rule Interchange Format

The Schema Transformation Network Service State of the Art Analysis identified three mapping languages which could serve as the mapping transfer format for the Schema Transformation Network Service: RIF, QVT and XSLT.

Before discussing the relative merits of the rule interchange formats, first let us establish some terminology that should help clarify the context.

The objective of the mapping language is to define a transformation between a source dataset (***sd***) in source logical schema (***LS***) into a target dataset (***td***) in a target logical schema (***LT***). In operation, the service must process a concrete physical encoding of the source and target datasets (***sp*** and ***tp*** respectively) and the source and target schema (***PS*** and ***PT*** respectively).

Query View Transformation (QVT)

The QVT/MOF/OMG family of languages are well suited to the conversion of logical schema expressed in one physical encoding into an alternative physical encoding. In terms outlined this would be like translating ***PS*** into ***PS****, where both ***PS*** and ***PS**** are equivalent expression of the same logical schema (***LS***) in different encoding formats (say UML and XSD for example). This does not help with either defining the relationship between the logical schema (***LS*** and ***LT***), the physical schemas (***PS*** and ***PT***) or the data itself. QVT has therefore been rejected as a candidate model definition language.

Extensible Stylesheet Language Transformations (XSLT)

XSLT is part of the XML tool stack. This stack contains two closely coupled technologies for expressing physical encodings of physical model structure (XSD) and dataset content (XML). In the terms outlined an XSLT document describes directly how a target dataset (***tp***) can be derived from a source dataset (***sp***). XSLT itself does make reference to the data models of either source or the target dataset (that is to say ***LS***, ***LT***, ***PS*** or ***PT***). The XML stack has very good tools support and a very high degree of developer adoption. It should be noted that the physical model encoding language currently does not support multiple inheritance. This has been address in a recent version of the standard (XSD 1.1), however this standard is only just beginning to be adopted by industry and full adoption may be a long way away. Developer expertise with XSLT is more limited; its strictly declarative nature (no bad thing in a transformation language) can present a steep learning curve to developers from an object-oriented or procedural background. There are two limitations of XSLT which make unsuitable for use as the mapping definition language for the Schema Transformation Network Service. Firstly, XSLT is focused at transforming XML at a syntactic level, rather than a feature/object level. This is not a problem for it typical use in the presentation layer of applications, however for spatial transformation it forces the developer to focus on the structure of the XML encoding, rather than of the structure of the logical schema. This in turn leads to verbose expressions that are hard to read and to debug. A second limitation of XSLT is more to do with the available implementation technologies, rather than the standard itself. XSLT engines load the entire source document into memory, before processing. Spatial datasets, when expressed in GML, may grow to many hundreds of megabytes. Documents of this size do require specialist handling (streaming/disk-backed-storage) and are not well suited to direct in memory transformation. For these two reasons XSLT has been rejected as a candidate model definition language.

Rules Interface Format (RIF)

RIF, combined with GML application schema for source and target schemas, presents a strong candidate. Like XSLT, it directly defines how a target dataset (***tp***) can be derived from a source dataset (***sp***). Unlike XSLT it is less tightly bound to XML syntax, making it easier to translate into other transformation languages.

It supports frame logic: i.e. this is the formal underpinning of object-oriented programming (see, for example, [22], which discusses the close similarity between the concepts of *frame* and *object*). Therefore, RIF can deal with object instances. It also has a strong intellectual and practical background, see below for more information.

RIF has been designed primarily to be a rule/action interchange language (rather than an implementation language). It is mathematically rigorous, and has been disseminated by a well-respected international standards body (W3C [44]). RIF has been designed by a group of experts representing the distilled experience of the Business Rules Management Systems industry. It is appropriate to the challenge of schema transformation, and indeed it is one of the use cases for which the format was designed, see [45].

Documentation on RIF is available at the W3C RIF Working Group website (see [46]). This website contains the full specification of RIF dialects, test cases, papers and information on current implementations of RIF.

RIF has several dialects, which the following table summarizes:

Dialect	Description
Core	Subset of basic rules common to all dialects.
Basic Logic Dialect (BLD)	First-order logic-based language focussing on the expression of relations between source and target models. It was written as an application of RIF-FLD to demonstrate the extensibility of RIF.
Production Rules Dialect (PRD)	Provides support for rules and actions. It has not been written as an application of RIF-FLD as the framework is only currently able to support first-order logic-based languages and not rules/actions type languages, but is instead specified separately.
Framework for Logic Dialect (FLD)	A framework that enables the definition of new logical languages without having necessarily to state explicitly every aspect of the new language, since the principle is that it re-uses default settings stated in the framework, while enabling the extending implementation to change only those aspects it is designed to target.

Table 25 RIF Dialects

The only RIF dialect capable of expressing the creation of new target instances (a requirement for a schema mapping language) is RIF-PRD.

Recommendation: The Schema Transformation Network Service should use RIF-PRD as the mapping definition language.

As well as supporting a range of basic functions, RIF has support for the inclusion of user defined functions. To support the spatial nature of mappings, the Schema Transformation Network Service should support all the operations identified in the OGC Simple Feature Specification (see [52]). Appendix F to this document describes how the operations defined in the OGC Simple Feature Specification can be referenced from RIF mappings.

Recommendation: The Schema Transformation Network Service should support all the basic RIF functions.

Recommendation: The Schema Transformation Network Service should support all the operations identified in the OGC Simple Feature Specification.

Appendix D: Compatibility of Rule Interchange Format and GML

It is possible to import GML application schema directly into RIF documents and make reference to the datatypes defined in them, in order to construct data transformations. Here is an example of this.

Given a GML document in which multiple features are defined such as this railway line:

```
<gml:featureMember>
    <src:TransportFeature gml:id="ida6c50110-dcac-4560-90b6-22dfdabaf948">
      <src:fema>CL RAIL</src:fema>
      <src:survey>20040906</src:survey>
      <gml:curveProperty>
        <gml:LineString srsName="EPSG:29902" srsDimension="3">
          <gml:posList>305152.443986936 316670.370978073 0 305169.344009866 316442.815006442 0
305196.160952081 316259.726042089 0 305276.066960803 316072.356043934
249999999968949000000000000000000000000</gml:posList>
        </gml:LineString>
      </gml:curveProperty>
    </src:TransportFeature>
  </gml:featureMember>
```

A RIF document to express the transformation could be written like this (this example is abbreviated to focus on the essentials of the mapping):

```
Document
(  Prefix(xsd <http://www.w3.org/2001/XMLSchema#>)
   Prefix(gml-311 <http://schemas.opengis.net/gml/3.1.1/gml.xsd#>)
   Prefix(gml-321 <http://schemas.opengis.net/gml/3.2.1/gml.xsd#>)
   Prefix(net <urn:x-inspire:specification:gmlas:Network:3.2#>)
   Prefix(tn <urn:x-inspire:specification:gmlas:CommonTransportElements:3.0#>)
   Prefix(tn-ra <urn:x-inspire:specification:gmlas:RailwayTransportNetwork:3.0#>)
   Prefix(src <http://www.somedataprovider.org/gml/transport#>)

   Group
   (  Forall ?feat ?trans ?type ?date ?geom ?id ?lstr
      (  If
         (  And
            (  ?feat # gml-311:featureMember
               ?feat[src:TransportFeature->?trans]
               ?trans[src:fema->?type
                      src:survey->?date
                      gml-311:curveProperty->?geom
                      gml-311:id->?id]
               ?geom[gml-311:LineString->?lstr]
            )
         )
         Then Do
         (  Assert
            (   ?iId New()
                ?iId # net:inspireId
                ?iId[base:localId->xsd:string(?id)
                     base:namespace->"AN.LMO"^^xsd:string]
                ?rllink New()
```

```
                ?rllink # tn-ra:RailwayLink
                ?rllink[net:beginLifespanVersion->xsd:dateTime(?date)
                        net:inspireId->?iId
                        net:endLifespanVersion[@nilReason]->"missing"^^xsd:string
                        net:inNetwork->"LMO"^^xsd:string
                        net:fictitious->"false"^^xsd:boolean
                        net:centerlineGeometry->?geom
                        tn:validFrom->?xsd:dateTime(?date)
                        tn:validTo[@nilReason]->"missing"^^xsd:string]
                 ?clg New()
                 ?clg # net:centerlineGeometry
                 ?curv New()
                 ?curv # gml-321:LineString
                 ?rllink[net:centerlineGeometry->?curv)
                 ?rllink[gml-321:id->?id]
                 ?curv[gml-321:LineString->?lstr]
                ?featMemb New()
                 ?featMemb # gml-321:featureMember
                 ?featMemb[net:NetworkElement->?rllink]
            )
         )
      )
   )
)
```

When combined with other rules, as part of a transformation, this helps to build the following INSPIRE GML:

```
<TransportFeatures aggregationType="set">
   <gml:boundedBy>
      <gml:Envelope srsName="urn:x-ogc:def:crs:EPSG:29902">
         <gml:lowerCorner>305152.443986936 316072.356043934</gml:lowerCorner>
         <gml:upperCorner>305276.066960803 316670.370978073</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
   <featureMember>
      <tn-ra:RailwayLink gml:id="ida6c50110-dcac-4560-90b6-22dfdabaf948">
         <net:beginLifespanVersion>2004-09-06T00:00:00.000</net:beginLifespanVersion>
         <net:inspireId>
            <base:localId>ida6c50110-dcac-4560-90b6-22dfdabaf948</base:localId>
            <base:namespace>AN.LMO</base:namespace>
         </net:inspireId>
         <net:endLifespanVersion nilReason="missing"/>
         <net:inNetwork>LMO</net:inNetwork>
         <net:fictitious>false</net:fictitious>
         <net:centerlineGeometry>
            <gml:curveProperty>
               <gml:LineString srsName="EPSG:29902" srsDimension="3">
                  <gml:posList>305152.443986936 316670.370978073 0
                               305169.344009866 316442.815006442 0
                               305196.160952081 316259.726042089 0
                               305276.066960803 316072.356043934
                               249999999968949000000000000000000000000
                  </gml:posList>
               </gml:LineString>
            </gml:curveProperty>
```

```
         </net:centerlineGeometry>
         <tn:validFrom>2004-09-06T00:00:00.000</tn:validFrom>
         <tn:validTo nilReason="missing"/>
      </tn-ra:RailwayLink>
   </featureMember>
   <featureMember xlink:href="#f3"/>
</TransportFeatures>
```

Appendix E: Common Transformations Expressed Using Rule Interchange Format

Transformations, as encoded using rule languages, are composed of **patterns**. These patterns provide a way to modularise the construction of a rule document. Here follow details of patterns identified based on third-party research and our experience of INSPIRE testing. For each pattern, an example is given using RIF to perform a transformation from a nominal source schema (which is not explicitly identified but is based on an actual example LMO dataset) to conform to the INSPIRE logical schema.

Please note, this appendix does not constitute a primer in the use of RIF, but a demonstration of its ability to express the mappings required for INSPIRE schema transformation. However, an attempt has been made to show only the essential features of the mappings in order to make them useful for instruction in the basic constructs of RIF.

Patterns Drawn From Third-Party Research

Work has already been done to analyse the kinds of transformations that are commonly required to achieve schema transformations (see [47]). This study identified various categories of schema transformation activity. The key concept is that the process of mapping from a source to a target schema requires the identification of **correspondences** between source and target data in the form of classes and attributes (which themselves could be classes). The broad categories of correspondences are in the following table. Within these categories, further sub-divisions can often be identified, depending on the context.

Case	*Type of correspondence*	*Description*
1	Single correspondences [1:1]	A datum that is required by the target schema is available in the source schema in the same format
2	Single correspondences [n:1]	Several data in source schema map to one datum in target schema
3	Single correspondences [1:n]	One datum in source schema maps to several data in target schema
4	Single correspondences [m:n]	Several data in source schema map to several data in target schema.
5	Missing correspondences [source]	A datum that is available in the source schema is not available in the target schema
6	Missing correspondences [target]	A datum that is required in the target schema is not available in the source schema
7	Multiple correspondences [choice]	Special situations with choice (i.e. co-occurrence) constraints
8	Multiple correspondences [structural]	Differences between source and target schema structures
9	Duplicate correspondences	Two or more items in source schema that map to the same item in target schema (or possibly to as many duplicates, depending on the mapping policy)

Table 26 Categories of Correspondences

The approach used in the following subsections is to present the relevant parts of the source and target schema, both from the viewpoint of the logical schema and of the actual data. The examples are based on actual INSPIRE requirements, although presented in simplified format to make the rulesets easier to understand. The INSPIRE network, hydrography and cadastral parcel application schema are referred to in many of these examples to fulfil the role of the target schema (see [8]). Some of the examples are based on hypothetical scenarios.

Case 1: Single Correspondences (One-To-One)

Here is an example where an attribute in a source schema class maps directly to an attribute in a target schema class. In this case, the `line` instance's `geometry` attribute maps to the `RailwayLink`'s `centerlineGeometry` attribute. A new instance of the target class is created in order to receive the data.

The following RIF ruleset expresses the required transformation.

```
Forall ?line
(  If ( ?line # src:line )  )
   Then Do
   (  Assert
      (
         (?lineGeom ?line[gml:geometry->?lineGeom])
          ?rllink New()
          ?rllink # tn-ra:RailwayLink
          ?rllink[net:centerlineGeometry->?lineGeom]
      )
   )
)
```

Case 2: Single Correspondences (Many-To-One)

Here is an example where multiple instances in the source schema map to a single instance in the target schema. In this case, a series of lines stored as distinct objects in the source schema map to a `RailwayLinkSequence` in the target schema (as well as mapping to other elements in the schema, N.B. the extra attribute mapping is not shown here).

The following RIF ruleset expresses the required transformation.

```
(* rule1 – first pass to create RailwayLinks and DirectedLinks and associate them *)
Group
(  Forall ?line
   (  If( ?line # src:line )
      Then Do
      (  Assert
         (  ?rllink New()
            ?rllink # tn-ra:RailwayLink
            (?lineGeom ?line[gml:geometry->?lineGeom])
            ?rllink[net:centerlineGeometry->?lineGeom]
            ?dirlink New()
            ?dirlink # net:DirectedLink
            (* create 1-to-1 relationship between rllink and dirlink *)
            ?dirlink[net:link->?rllink]
         )
```

```
      )
   )
)

(* rule2 - create a new RailwayLinkSequence object if none exists *)
Group
(  Forall ?rlseq
   (  If
      (  And
        (  ?rlseq # tn-ra:RailwayLinkSequence
           Not
           (  Exists
              (  ?rlseq # tn-ra:RailwayLinkSequence  )
           )
        )
      )
      (  Then
         (  Do
            (  Assert
               (  ?rlseq New()
                  ?rlseq # tn-ra:RailwayLinkSequence
               )
            )
         )
      )
   )
)
(* rule3  - assign DirectedLinks to the new RailwayLinkSequence list *)
Group
(  Forall ?dirlink
   (  If
      (  And
         (  ?dirlink # net:DirectedLink
            Not
            (  Exists
               (  ?rlseq
                  (  And
                     (  ?rlseq # tn-ra:RailwayLinkSequence
                        (?dirlink External(pred:list-contains(?rlseq[link] ?dirlink)))
                     )
                  )
               )
            )
         )
      )
      Then
      (  Do
         (  Modify
            (  (?list ?rlseq[net:link->?list])
               External(func:append(?list ?dirlink))
            )
         )
      )
   )
)
```

Case 3: Single Correspondences (One-To-Many)

Here is an example where a single source instance maps to multiple target instances. In this case, a single source object `line` is transformed into multiple target objects: the `inspireId` identifier plus the `RailwayLink` and `RailwayType` feature types (this is only one kind of multi-cardinality transformation: it is equally possible to transform instances of the same type, whereby for example one has an object with a list attribute in the source schema and individual top-level instances in the target schema).

In addition to the excerpt given at the start of this section, this example also refers to the following part of the INSPIRE transport networks logical schema.

```
Forall ?line, ?featureId ?surveyDate ?lineGeom
(  If
   (   And
   (  ?line # src:line
      ?line[src:featId->?featureId src:survey->?surveyDate
                gml-311:geometry->?lineGeom]
   )
   )

Assert
(  And
   (  ?iId New()
      ?iId # net:inspireId
      ?iId[localId->xsd:string(?featureId) namespace->"NI.LPSNI"^^xsd:string]

      ?rllink New()
      ?rllink # tn-ra:RailwayLink
      ?rllink
      [
         net:inspireId->?iId
         net:beginLifespanVersion->xsd:dateTime(?surveyDate)
         net:centrelineGeometry->?lineGeom
      ]

      ?rltype New()
      ?rltype # tn-ra:RailwayType
      ?rltype
      [
         net:inspireId->?iId
         net:networkRef->?External(function:make-list(?rllink))
         net:beginLifespanVersion->uml:DateTime(?surveyDate)
         tn-ra:type->"train"^^xsd:string
      ]
   )
)
```

Case 4: Single Correspondences (Many-To-Many)

Here is an example where multiple source instances map to multiple target instances.

Consider a source schema which has its transport network logical schema expressed using sets of road lines named `Road`, each of which contains a multi-cardinality `leadsTo` attribute which is a list of object references of all other road lines that touch the given line, and thus expresses an association between road lines. In the INSPIRE network application schema, however, links are associated with nodes, rather than directly with each other. Therefore, in order to map this data into the INSPIRE logical schema it is necessary to infer the presence of network nodes from the fact of lines touching each other. This is different from Case 8 below (where leaf and inner nodes are cross-mapped) because, in this case, the nodes required in the target schema do not exist in the source schema, although they are logically inferrable.

This example shows the required transformation expressed using RIF. The example is simplified to show only the essential elements of the transformation. It shows a spatial library being included in the document to enable the source geometry to be analysed. Note that this example has the potential to create redundant, duplicate nodes, and needs more work to address those limitations.

```
Document
(  Prefix(gml-311 <http://schemas.opengis.net/gml/3.1.1/gml.xsd#>)
   Prefix(net <urn:x-inspire:specification:gmlas:Network:3.2#>)
   Prefix(tn-ro <urn:x-inspire:specification:gmlas:RoadTransportNetwork:3.0#>)
   Prefix(src <http://www.somedataprovider.org/gml/transport#>)
   Prefix(spatial http://1spatial.com/inspire/spatialfunctions#)

   (* rule1 - create RoadLinks and associated RoadNodes *)
   Group
   (
      Forall ?line ?geom
      (  If
         (  And
            (
               ?line # src:RoadLine
               ?line[gml-311:geometry->?geom]
            )
         )
         Then Do
         (  Assert
            (
               ?rdlink New()
               ?rdlink # tn-ro:RoadLink
               ?rdlink[net:centrelineGeometry->?geom]
               ?rdnode New()
               ?rdnode # tn-ro:RoadNode

               (* assign the last set of coordinates in the line to be
                  the coordinates of the node *)
               (?coordlist External(spatial:getCoords(?geom)))
               ?rdnode[net:geometry->External(func:get(?coordlist
                                          External(func:count(?coordlist))))]
               ?rdnode[net:spokeEnd->External(function:append(?rdlink)]
            )
         )
      )
   )
)
```

Case 5: Missing Correspondences (Source Lacks Data)

When data required by the target schema is lacking in the source schema, there are many strategies for handling the lack.

b) Assign a default value to a target instance.

e.g.

```
?iId New()
?iId # net:inspireId
?iId[net:namespace->"AN.LMO"^^xsd:string]
```

In this case, the `namespace` attribute of the `inspireId` is assigned a constant string value identifying the source data provider.

c) Assign a random value to a target instance (provided this is optional data: in the case of mandatory data, it is unwise to assume a random value would meet business requirements). This strategy is primarily of interest to transformation mapping designers as a way of facilitating trial-and-error development of mappings (e.g., see below section 8.3, under the Mapping Designer user interface discussion, the proposed random value assignment feature of the Mapping Definition's target tab). However, it will not be expanded upon in this section because it is clearly outside the scope of the INSPIRE Schema Transformation Network Service solution.

d) Leave a target instance null (transformation ok, only applies where target value is nullable). In this snippet, the `validFrom` and `validTo` attributes of the `RailwayType` NetworkProperty have not been assigned. Since they are specified as voidable in the logical schema this is acceptable. If suitable, a ReasonForVoidValue may be provided to explain why the value has not been populated.

```
    ?rltype New()
    ?rltype # tn-ra:RailwayType
    ?rltype
    [
       net:inspireId->?iId
       net:networkRef->?External(function:make-list(?rllink))
       net:beginLifespanVersion->uml:DateTime(?surveyDate)
       tn-ra:type->"train"^^xsd:string
    ]
```

e) Transformation fails, where it is not possible to follow one of the above strategies.

Case 6: Missing Correspondences (Target Lacks Data)

Here is an example where data present in the source schema has no corresponding location in the target schema i.e. information would be lost if the mapping were reversed.

Take for example the `RoadSurfaceCategory` feature type in INSPIRE transport networks logical schema. INSPIRE schema says: `paved`, `unpaved`. A national dataset with a feature classifier `Road` could conceivably have an enumerated attribute called `surface` with permitted values being: `asphalt`, `concrete`, `brick`, `cobblestone`, `permeable`, `wood`, `gravel`, `cinder`, `earth`. The first two values map more or less well to the target value `paved`. The last three values map to the target value unpaved. But can we say that a wooden road surface is paved? Hence, the target schema, in this case, lacks the expressiveness required. It is noted that this is seldom the case with the INSPIRE schema, and the example is an artificial one; furthermore, it is not challenging to express a transformation in RIF, the problem is more that one would have to make a business decision as to whether to void the target attribute or to apply the more likely of the two permitted target values. Therefore, this case will not be discussed any further.

Case 7: Multiple Correspondences (Choice)

The INSPIRE GML application schema use substitution groups heavily. Many of the substitution groups values are domain-restricted to various codelists (e.g. `FormOfWay` in the Transport Networks data theme in expressed as a substitutable element for the `tn:TransportProperty` head element). For example:

In the INSPIRE schema we have

```
<element name="FormOfWay" substitutionGroup="tn:TransportProperty" type="tn-ro:FormOfWayType">
    <annotation>
      <documentation>-- Definition --&#13;
A classification based on the physical properties of the Road Link. [TWG TN, based on
EuroRoadS]</documentation>
    </annotation>
  </element>
```

A RIF ruleset to manage this is shown under “domain mapping” below. Alternatively, A multi-choice type of construct could be expressed in RIF by testing for the presence of certain values in source attributes. An example of this is

```
Group
(   Forall ?line ?featureId ?surveyDatye ?theme ?lineGeom
   (  If
      (  And
         (  ?line # src:line
            ?line[src:featId->?featureId src:survey->?surveyDate src:fema->?theme
                     gml:geometry->?lineGeom]
            ?line[src:tema->?category]
            Or
            (  External(pred:boolean-equal(?category "MOTORWAY"^^xsd:string))
               External(pred:boolean-equal(?category "A_CLASS"^^xsd:string))
               External(pred:boolean-equal(?category "DUAL_CARR"^^xsd:string))
               External(pred:boolean-equal(?category "B_CLASS"^^xsd:string))
               External(pred:boolean-equal(?category "<4M_TARRED"^^xsd:string))
               External(pred:boolean-equal(?category "<4M_T_OVER"^^xsd:string))
               External(pred:boolean-equal(?category "CL_MINOR"^^xsd:string))
               External(pred:boolean-equal(?category "CL_M_OVER"^^xsd:string))
            )
         )
      )
      Then
      (  Do
            (* create new tn-ro:RoadLink and associated properties (not shown here) *)
```

In this rule condition block, the `fema` attribute is tested against a set of values and, if it meets at least one of the conditions in the `Or` block, a new instance of the `tn:roadLink` class is created. The selection of which type of target object to create is controlled by assertions on source attributes.

Case 8: Multiple Correspondences (Structural)

Here is an example where there are structural differences between source and target models such that leaf (attribute) and inner (class) nodes change their relationships and it is necessary to navigate these differences to transform meaning from one to the other.

```
If
(  Or
   (  External(pred:boolean-equal(?category "RL TUNNEL"^^xsd:string))
      External(pred:boolean-equal(?category "CL RAIL"^^xsd:string))
      External(pred:boolean-equal(?category "UNSHOWN RL"^^xsd:string))
   )
)
Then
(   Do
   (  ?rllink New()
      ?rllink # tn-ra:RailwayLink
   )
)
```

In this case, a check is made on the value of the variable `?category` which has previous been assigned the category of the transport object defined in the source schema. Based on this value being within one of the constant value in the `Or` clause, a new object instance is created in the target schema. Thus, a source leaf node maps to a target inner node.

Case 9: Duplicate Correspondences

This refers to the same data appearing twice in the source schema (i.e. identical elements part of the same parent element or collection). The fact of their duplication does not automatically mean there is redundancy, as their cardinality can in itself be useful information. Therefore, it is necessary to enable the user to specify whether to filter them out, or to copy them across intact.

It is a tenet of RIF-Core [49] that duplicate elements in a source schema, when matched by a predicate in a rule condition, do not cause the outcome of the evaluation of the condition to produce a different result from the result in the absence of such duplicates. However, the challenge in the case of production rules and creation of target schema objects, is that the cardinality of such objects may not be respected in cases where it should be. In a `Forall` condition combined with a `Do… Assert` construct, every instance in the fact base that satisfies the rule condition will be transformed into a new instance. So the default behaviour of RIF is to respect the cardinality of objects during transformation. In order to filter out duplicates, it would be necessary to change the approach slightly, to re-test the condition of the fact base after each rule condition is evaluated. This could be done as follows:

```
Group
(  Forall ?X ?C
   (  If
      (  And   ( ?X[ex:count -> ?C] pred:numeric-greater-than(?C 0) )
         //other rule conditions
      )
      Then
      (  Do
         (  Modify (?X[ex:count-> func:numeric-subtract(?C 1)]) )
            Assert ( ... )
         )
      )
      ex:foo[ex:count -> 1]
   )
)
```

This example is based on the RIF-PRD test case located at [48]. In this case, alongside the rule conditions which would be evaluated by default, a counter value is also evaluated. It is initialised with the value 1 and decremented by one each time the rule conditions are tested. This means the `Assert` action will be performed exactly once. The contents of the `Assert` block are left empty.

Pattern Drawn From INSPIRE Testing Activities

The following patterns are drawn from our experience of INSPIRE testing.

Case No.	***Type of correspondence***	***Description***
10	Straight mapping including sub-attribute level	Mapping source feature attributes to target instances, including where attributes are nested within source objects
11	Default value setting	Supplying default values for certain attributes
12	Functional value setting	Calling external functions to transform attribute values
13	Domain mapping	Aligning of source and target codelists and enumerations
14	Classifying	Identifying correct target instance taxonomy based on source attributes
15	Class Inheritance	Defining rules against parent classes that can apply to all child classes
16	Assignment of unique identifier values	Assigning guaranteed unique identifier values to target instances on creation
17	Complex operations	Sophisticated target schema enrichment operations
18	Intermediate objects	Use of intermediate objects to decompose mapping into multiple steps.

Table 27 Further Patterns Drawn From INSPIRE Testing

Case 10 Straight Mapping Including Sub-Attribute Level

This is a scenario where you have a source feature classifier with certain attributes. It maps fairly straightforwardly to a target feature class, with perhaps some direct attribute mappings and possibly sub-attribute mappings. It is equivalent to the scenario shown above under *Case 1: Single Correspondences (One-To-One)* and the example given under that heading serves to illustrate this category as well.

Case 11: Default Value Setting

See Missing correspondences (Source Lacks Data), point (a).

Case 12: Functional Value Setting

Consider this snippet:

```
?obj[gml:id->func:concat("LPSNI"^^xsd:string "_"^^xsd:string xsd:string(?featureId))]
```

Here, an externally-defined function (in fact, part of the RIF-DTB dialect [19]) named `concat` is called and returns the concatenation of its three string arguments, which is assigned to the `gmlid` attribute of the containing object.

Another example shown here, based on the Cadastral Parcels data theme, has the RIF-DTB function `substring()` which returns a truncated string being the first six characters of the `?parcelId` variable's contents, and the function `getPoint()` which is defined in a third-party spatial library and derives the value required to be assigned to the cp:referencePoint attribute as a side node (we assume the function guarantees that the returned point lies within the source geometry).

```
?parcel # src:PropertyRegistrationParcel
?geometry # gml:geometry
?cparcel # cp:CadastralParcel
?iId # net:inspireId
?iId[localId->xsd:string(?parcelId) namespace->"TDK.CP"^^xsd:string]
?parcelId ?parcel[cp:inspireId->?iId]
?cparcel[cp:label->External(func:substring(?parcelId, 0, 6))
         cp:referencePoint->External(func:getPoint(?geometry)))
```

Case 13: Domain Mapping

A typical INSPIRE transformation problem is the mapping of domains of values (e.g. codeLists and Enumerations) from one schema to another. The definition of the mapping requires domain expertise. However, the separate question of how to express the mapping using a rules language can be tackled in RIF using the following constructs.

Firstly, a set of lists are created to hold the source and target enumerations. Note that the index position of each element provides the required information to map from source to target values, e.g. `DUAL_CARR` in the source schema maps to `motorway` in the target schema.

```
   "http://1spatial.com/inspire/fema_list"^^rif:iri
   [func:make-list("MOTORWAY"^^xsd:string
          "A_CLASS"^^xsd:string
```

```
          "DUAL CARR"^^xsd:string
          "B CLASS"^^xsd:string
          "CL MINOR"^^xsd:string
          "CL M OVER"^^xsd:string
          "<4M TARRED"^^xsd:string
          "<4M T OVER"^^xsd:string )]


   "http://1spatial.com/inspire/fow list"^^rif:iri
   [func:make-list("motorway"^^xsd:string
         "singleCarriageway"^^xsd:string
         "motorway"^^xsd:string
         "singleCarriageway"^^xsd:string
         "singleCarriageway"^^xsd:string
         "singleCarriageway"^^xsd:string
         "singleCarriageway"^^xsd:string )]
```

Secondly, the lists are used to perform the mapping. The `index-of()` and `get()` functions defined in RIF-DTB enable retrieval of the index of the source value and, from that, the target value.

```
(?theme ?line[src:fema->?theme])
?fway New()
?fway # tn-ro:FormOfWay
?fway[tn-ro:formOfWay->External(func:get(fow_list External(func:index-of(fema_list
?theme))))]
```

Case 14: Classifying

This is the scenario where we map a source object to a target object, and make the selection of the target object's type (i.e. class) dependent on a logical formula which is a series of tests applied to one or more source attributes. In this example drawn from the Transport Networks data theme, the source feature class is CO_BUILDING, a generic building class, with a feature type attribute BUILDINGUS, which specifies the type of building. Suppose that we want to make the instantiation of a `RailwayStationArea` in the target schema happen only in cases where the `BUILDINGUS` value is `7.0` and the `DATE` is greater than `2009.1.1`. We could express that meaning using the following RIF constructs.

```
Forall ?building such that ?building # src:CO BUILDING
( If
  (  ?building[src:BUILDINGUS->"7.0"^^xsd:double src:DATE->"2009.1.1"^^xsd:string])
  (  Then
     (  Do
        (  Assert
           (  ?rsa New()
              ?rsa # tn-ra:RailwayStationArea
              (?geom ?building[src:GEOMETRY->?geom])
              ?rsa[net:geometry->?geom]
           )
        )
     )
  )
```

```
)
```

This exemplifies application of a target object classification based on source attributes.

Case 15: Class Inheritance

In many cases, source schema features can be identified as occupying a place within an ontological inheritance hierarchy, as a child class of another, perhaps abstract, class. For example, in the INSPIRE application schema, the Hydrography application schema contains the `hydro-base` application schema which defines the `HydroObject` class. This class is a parent for every other hydrographic feature class and provides a central place for defining certain identifier type information. The creation of subclass instances of the `HydroObject` during a transformation could involve use of rules that take advantage of the fact that everything being created in the target schema (as far as hydrography is concerned) "is a" `HydroObject` element. A rule such as the following can be defined once for the entire transformation, rather than repeatedly for each concrete class instance. This approach simplifies the rules base. N.B. these rules operate on target instances created in a previous pass or action. In cases where there is already an inheritance hierarchy in the source schema, rules of this sort can also operate during a first pass. Note that this approach could also apply to setting of the `gmlid` on target features being subclasses of `AbstractFeature`.

```
 (* Rule 1 - process a concrete class. *)
Forall ?riversegment
(  If
   (  ?riversegment # src:RiverSegment  )
   Do
   (  Assert
      (  ?wcls New()
         ?wcls # hy-n:WatercourseLinkSequence
         (* etc for other straight attribute mappings *)
      )
   )
)

(* Rule 2  - process its abstract class. *)
(* assuming rule condition variables ?id and ?dp have already been declared and assigned *)
Forall ?ho
(  If
   (  ?ho # hy:HydroObject )
   Do
   (  Modify
      (  ?gname New()
         ?gname # gn:GeographicalName
         ?hi New()
         ?hi # hy:hydroIdentifier
         ?hi[hy:localId->?id hy:namespace->?dp]
         ?ho[gn:GeographicalName->External(function:append(?gname))
             hy:hydroId->External(function:append(?hi))]
      )
   )
```

```
)
```

Case 16: Assignment Of Unique Identifier Values

The Schema Transformation Network Service is responsible for deriving INSPIRE features from source features. Each INSPIRE feature needs a unique and persistent identifier. To be unique and persistent it is recommended that all INSPIRE identifiers be derived, using a deterministic process, from information contain in the attributes of the source features. In this way, if the transformation process is repeated, an identical set of inspire features will generated. The derivation process should be insensitive to the order in which features are transformed. Specifically, it is not recommended that any external state (like database sequences) be used in the derivation of INSPIRE identifiers.

In a simple example, where one INSPIRE feature is derived from one source feature, all that is required is to copy the source attribute, and set the namespace to a unique string managed by the owner of the source dataset. If the structure of the INSPIRE identifier changes over time, different approaches could be applied to derive a unique and persistent identifier.

```
Group
(  Forall ?X
   (  If
      (
         (* rule conditions here, bind ?id variable to a source id attribute.
           And CO.DP is a unique namespace managed by the owner of the source data. *)
      )
      Then Do
      (  Assert
         (   ?inspireId New()
             ?inspireId # net:InspireId
             ?inspireId[base:localId-> ?id)]
             ?inspireId[base:namespace->"CO.DP."^^xsd:string)]
         )
      )
      ex:foo[ex:count -> 5]
   )
)
```

Where more than one INSPIRE feature is derived from a single source feature, the process can be adapted by

1) Using the outlined process to generate a unique INSPIRE identifier for the source feature (whether or not it will actually be outputted)

2) Generated a set of identifiers for the generated INSPIRE features that use the INSPIRE id of the source feature as a 'namespace'. This namespace need not be placed in nominated namespace attribute; it may merely be an appropriately delimited prefix to the localId.

It is not clear that it would be possible to generate meaningful persistent INSPIRE identifiers if the source datasets do not contain enough information to derive them in a deterministic manner. In such circumstances, it would still be possible to using a randomly generated UUID to assign a unique identifier. However, running the transformation process twice would generate a different set of identifiers.

Case 17: Complex Operations

The following is a complex operation defined using natural language. It comprises a ruleset following by a sequence of actions which are performed if the rule is satisfied. The action states:

"For cadastral boundaries that touch cadastral parcels, if it is either a left or right boundary, then assign the cadastral boundary's reference as the cadastral parcel instance's `cp:boundary` *attribute and, likewise, assign the cadastral parcel's reference as the cadastral boundary instance's* `cp:parcel` *attribute."*

This would be cumbersome to implement using RIF, therefore it is recommended that operations be defined as external functions and imported into the RIF document. See the "black-box" approach described in Section 6.2.3 Spatial Extensions.

Case 18: Intermediate Objects

Mapping definitions can sometimes be simplified through the use of intermediate objects. These allow the mapping process to be broken into a series of transformations, rather than tackled in one go. Although use of intermediate objects introduces the complexity of an extra schema, it results in a more modular mapping definition, which may be easier to design and maintain.

Intermediate objects may be used to merge logically similar classes in the source dataset (such as RIVER_POLYGON and RIVER_LINE) before translation to an INSPIRE class (e.g. Watercourse).

Equally, intermediate objects could be used to separate logically dissimilar objects in the source data (for example, split TRANSPORT_LINES into ROAD_LINKS and RAIL_LINKS) before translation to different INSPIRE classes.

The Schema Transformation Network Service should only output data in the nominated target schema. This means that intermediate objects can be created in an alternative schema without them appearing in the generated dataset. It is not recommended that intermediate objects are generated in the source schema as this may lead to race conditions and/or infinite loops.

Appendix F: Supported Simple Features Functions and Predicates

The following is a listing of the spatial functions and predicates supported by the Schema Transformation Network Service prototype. It illustrates the approach to naming functions and predicates proposed in the main body of the document (see Section 6.2.4). Note that it is up to the implementers of a given transformation engine to implement these functions and predicates and document clearly any that they do not support.

IRI of Function or Predicate
http://www.opengeospatial.org/standards/sfa/IGeometry#GetDimension
http://www.opengeospatial.org/standards/sfa/IGeometry#GetSpatialReference
http://www.opengeospatial.org/standards/sfa/IGeometry#SetSpatialReference
http://www.opengeospatial.org/standards/sfa/IGeometry#IsEmpty
http://www.opengeospatial.org/standards/sfa/IGeometry#IsSimple
http://www.opengeospatial.org/standards/sfa/IGeometry#Envelope
http://www.opengeospatial.org/standards/sfa/IGeometry#Project
http://www.opengeospatial.org/standards/sfa/IGeometry#Extent2D
http://www.opengeospatial.org/standards/sfa/IWks#ExportToWKB
http://www.opengeospatial.org/standards/sfa/IWks#ExportToWKT
http://www.opengeospatial.org/standards/sfa/IWks#ImportFromWKB
http://www.opengeospatial.org/standards/sfa/IWks#ImportFromWKT
http://www.opengeospatial.org/standards/sfa/IGeometryFactory#CreateFromWKB
http://www.opengeospatial.org/standards/sfa/IGeometryFactory#CreateFromWKT
http://www.opengeospatial.org/standards/sfa/IPoint#Coords
http://www.opengeospatial.org/standards/sfa/IPoint#GetX
http://www.opengeospatial.org/standards/sfa/IPoint#GetY
http://www.opengeospatial.org/standards/sfa/ICurve#GetLength
http://www.opengeospatial.org/standards/sfa/ICurve#StartPoint
http://www.opengeospatial.org/standards/sfa/ICurve#EndPoint
http://www.opengeospatial.org/standards/sfa/ICurve#IsClosed
http://www.opengeospatial.org/standards/sfa/ICurve#Value
http://www.opengeospatial.org/standards/sfa/ILineString#GetNumPoints
http://www.opengeospatial.org/standards/sfa/ILineString#Point
http://www.opengeospatial.org/standards/sfa/ISurface#GetArea
http://www.opengeospatial.org/standards/sfa/ISurface#Centroid
http://www.opengeospatial.org/standards/sfa/ISurface#PointOnSurface
http://www.opengeospatial.org/standards/sfa/IGeometryCollection#GetNumGeometries
http://www.opengeospatial.org/standards/sfa/IGeometryCollection#Geometry
http://www.opengeospatial.org/standards/sfa/IPolygon#ExteriorRing
http://www.opengeospatial.org/standards/sfa/IPolygon#GetNumInteriorRings
http://www.opengeospatial.org/standards/sfa/IPolygon#InteriorRing
http://www.opengeospatial.org/standards/sfa/IMultiCurve#GetLength
http://www.opengeospatial.org/standards/sfa/IMultiCurve#IsClosed
http://www.opengeospatial.org/standards/sfa/IMultiSurface#GetArea
http://www.opengeospatial.org/standards/sfa/IMultiSurface#Centroid
http://www.opengeospatial.org/standards/sfa/IMultiSurface#PointOnSurface
http://www.opengeospatial.org/standards/sfa/ISpatialRelation#Equals
http://www.opengeospatial.org/standards/sfa/ISpatialRelation#Touches
http://www.opengeospatial.org/standards/sfa/ISpatialRelation#Intersects
http://www.opengeospatial.org/standards/sfa/ISpatialRelation#Contains
http://www.opengeospatial.org/standards/sfa/ISpatialRelation#Within
http://www.opengeospatial.org/standards/sfa/ISpatialRelation#Disjoint
http://www.opengeospatial.org/standards/sfa/ISpatialRelation#Crosses
http://www.opengeospatial.org/standards/sfa/ISpatialRelation#Overlaps
http://www.opengeospatial.org/standards/sfa/ISpatialRelation2#Relate
http://www.opengeospatial.org/standards/sfa/ISpatialOperator#Distance
http://www.opengeospatial.org/standards/sfa/ISpatialOperator#Boundary
http://www.opengeospatial.org/standards/sfa/ISpatialOperator#Buffer
http://www.opengeospatial.org/standards/sfa/ISpatialOperator#ConvexHull
http://www.opengeospatial.org/standards/sfa/ISpatialOperator#Intersection
http://www.opengeospatial.org/standards/sfa/ISpatialOperator#Union
http://www.opengeospatial.org/standards/sfa/ISpatialOperator#Difference
http://www.opengeospatial.org/standards/sfa/ISpatialOperator#SymmetricDifference

Table 28 Supported and Unsupported Simple Features Functions and Predicates

Appendix G: Definitions, Acronyms, Abbreviations and Initials

Definitions, acronyms, abbreviations and initials are available in the INSPIRE Glossary [2]. The following are also introduced in this document.

Acronyms, Abbreviations and Initials	***Expansions and Definitions***
ATL	*ATLAS Transformation Language*, a model transformation language and toolkit to produce a set of target models from a set of source models.
CRS	*Coordinate Reference System*, a system that defines how georeferenced spatial data relates to real locations on the Earth's surface.
DQ	*Data Quality*: the processes and technologies involved in ensuring the conformance of data values to business requirements and acceptance criteria (see [50]).
ebXML	*Electronic Business using eXtensible markup Language*, a family of XML based standards whose mission is to provide an open, XML-based infrastructure for exchanging business information.
EC	*European Commission*, the executive body of the European Union.
ETL	*Extract, Transform and Load*, a database and data warehousing term referring to the process of migrating data from one storage format to another.
FME	A widely-used spatial ETL solution that enables GIS Professionals to translate, transform, integrate and distribute spatial data.
GIS	*Geographic Information System*, a system that captures, stores, analyses, manages, and presents data that are linked to location.
GML	*Geography Markup Language*, the XML grammar defined by the OGC to express geographical features
gOML	*Geographic Profile of the OML*, a subdialect of OML written by the Humboldt Community for use in its Alignment Editor and Conceptual Schema Transformer applications.
HTTP	*Hypertext Transfer Protocol*, a protocol for distributed, collaborative, hypermedia information systems
INSPIRE	*Infrastructure for Spatial Information in the European Community*, a European Union wide initiative to harmonise spatial data infrastructures in order to support policy and decision making in Europe.
IOC	*INSPIRE Initial Operating Capacity Task Force*, a group commissioned to help and support the implementation of INSPIRE in the Member States of the European Union.
IR	*Implementing Rules*, a set of rules to ensure that the spatial data infrastructures of the Member States of the European Union are compatible and usable in a Community and transboundary context.
ISO	*International Organisation for Standardization*, the world's largest developer and publisher of international standards.
JRC	*Joint Research Centre*, the European Union's scientific and technical research laboratory and an integral part of the European Commission.
LMO	*Legally Mandated Organisation*, an organisation that has or will have a legal

Acronyms, Abbreviations and Initials	*Expansions and Definitions*
	mandate for one or more aspects of INSPIRE.
MOF	*Meta-Object Facility*, an OMG standard for model-driven engineering.
OGC	*Open Geospatial Consortium*, an international, non-profit organization engaged in development of standards for geospatial and location based services.
OMG	*Object Management Group*, an international, open membership, not-for-profit computer industry consortium involved in developing enterprise integration standards for a variety of technologies.
OML	*Ontology Mapping Language*, a modelling language enabling the user to specify correspondences between two ontologies.
QVT	*Query/View/Transform*, a standard for model transformation defined by the OMG.
RDF	*Resource Description Framework*, a general method for conceptual description or modelling of information that is implemented in Web resources.
RIF	*Rule Interchange Format*, a W3C standard for exchanging rules among rule systems, in particular among Web rule engines.
RIF-BLD	*RIF Basic Logic Dialect*, a RIF dialect that enables logic rules to be exchanged between rule systems.
RIF-Core	*RIF Core Dialect*, a common subset of the RIF-BLD and RIF-PRD dialects, based on RIF-DTB 1.0.
RIF-DTB	*RIF Datatypes and Built-Ins*, a RIF dialect containing datatypes, built-in functions and built-in predicates expected to be supported by RIF dialects such as RIF-PRD, RIF-BLD and RIF-Core.
RIF-FLD	*RIF Framework for Logic Dialects*, a framework for specifying the syntax and semantics of RIF logic dialects.
RIF-PRD	*RIF Production Rules Dialect*, a standard XML serialization format for production rule languages.
SDIC	*Spatial Data Interest Community*, an organisation with an interest in improving use of resources for spatial data management and the development and operation of spatial information services.
SOA	*Service-Oriented Architecture*, a system design approach that views components as online services which produce or consume data in relation to one another.
SOAP	A protocol specification for exchanging structured information in the implementation of Web Services in computer networks (original meaning was Simple Object Access Protocol)
SQL	*Structured Query Language*, a database computer language designed for managing data in relational database management systems.
TNS	*Transformation Network Service*, a Web service providing query or data instance transformation services.
UDDI	*Universal Description, Discovery and Integration*, a platform-independent, XML-based registry for publishing business services online.
UML	*Unified Modeling Language*, a standardised general-purpose modelling

Acronyms, Abbreviations and Initials	*Expansions and Definitions*
	language in the field of software engineering.
URL	*Uniform Resource Locator*, an identifier that specifies where a network resource is available and the mechanism for retrieving it.
UUID	*Universally Unique Identifier*, an identifier standard used in software construction, that enables distributed systems to uniquely identify information without significant central coordination.
W3C	The *World-Wide Web Consortium*, an international community of member organisations working to develop Web standards.
WFS	*Web Feature Service*, a specification providing an interface enabling requests for geographical features across the Web using platform-independent calls.
WFS-T	*Web Feature Service (Transactional),* a WFS with additional support for transactions, which are atomic, consistent, isolated and durable pieces of information processing activity.
WS-Addressing	*Web Services Addressing*, a specification providing transport-neutral mechanisms to address Web services and messages.
WSDL	*Web Services Description Language*, an XML format for describing network services.
WS-I	*Web Services Interoperability Organization*, an open industry organization chartered to establish best practices for Web services interoperability.
XMI	*XML Metadata Interchange*, an OMG standard for exchanging metadata information via XML, which is a serialization format for MOF-compliant models.
XML	*eXtensible Markup Language*, a set of rules for encoding documents electronically.
XSLT	*eXtensible Stylesheet Language Transformations*, a language for transforming XML documents into other XML documents.

Table 29 Explanation of Acronyms

Appendix H: References

[1] DIRECTIVE 2007/2/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE), http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2007:108:0001:0014:EN:PDF.

[2] INSPIRE glossary, http://inspire-registry.jrc.ec.europa.eu/registers/GLOSSARY.

[3] Regulation on INSPIRE Network Services (Commission Regulation (EC) No 976/2009 of 19 October 2009), http://ec.europa.eu/transparency/regcomitology/index.cfm?do=Search.getPDF&cI7TwVsORn+kLl9oziBPzRrPh2gD8ZmE8tZUqV9OrP7B7EJR+poTzWZ/2wT/z/JFTr7x0HnynbCJdi/BzR4ZvdPpAur0FOHhej8jYcN49FA=.

[4] Draft amendments to Regulation (EC) No 976/2009, http://ec.europa.eu/transparency/regcomitology/index.cfm?do=Search.getPDF&cI7TwVsORn+kLl9oziBPzRrPh2gD8ZmE8tZUqV9OrP7B7EJR+poTzWZ/2wT/z/JFTr7x0HnynbCJdi/BzR4ZvdPpAur0FOHhej8jYcN49FA=.

[5] Draft Implementing Rules for INSPIRE Transformation Services, http://inspire.jrc.ec.europa.eu/documents/Network_Services/INSPIRE_Draft_Implementing_Rules_Transformation_Services_(version_3.0).pdf.

[6] INSPIRE Network Services Architecture, http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/network/D3_5_INSPIRE_NS_Architecture_v3-0.pdf.

[7] INSPIRE Network Services SOAP Framework, http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/network/INSPIRE_NETWORK_SERVICES_SOAP_Framework.pdf.

[8] INSPIRE Data Specifications, http://inspire.jrc.ec.europa.eu/index.cfm/pageid/2.

[9] INSPIRE Schema Transformation Network Service “State Of The Art Analysis”.

[10] The Ontology Definition Metamodel, http://www.omg.org/spec/ODM/1.0/.

[11] INSPIRE Draft Technical Guidance Coordinate Transformation, http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/network/Draft_Technical_Guidance_Coordinate_Transformation_Services_v1.0.pdf.

[12] INSPIRE Draft Technical Guidance Download Services, http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/network/Draft_Technical_Guidance_Download_Services_v1.0.pdf.

[13] W3C Rule Interchange Format Working Group, http://www.w3.org/2005/rules/wiki/RIF_Working_Group.

[14] OASIS ebXML Registry TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=regrep.

[15] UDDI Specification v. 3, http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3.

[16] Rule Interchange Format Basic Logic Dialect, http://www.w3.org/TR/rif-bld/.

[17] Rule Interchange Format Framework for Logic Dialects, http://www.w3.org/TR/rif-fld/.

[18] Rule Interchange Format Production Rules Dialect, http://www.w3.org/TR/rif-prd/.

[19] Rule Interchange Format Datatypes and Built-Ins, http://www.w3.org/TR/rif-dtb/.

[20] OGC Simple Features Interface Standard (SFS), http://www.opengeospatial.org/standards/sfa.

[21] See http://lists.w3.org/Archives/Public/public-rif-comments/2009Aug/0024.html.

[22] *Logic Foundations of Object-Oriented and Frame-Based Languages* (Kifer M., Lausen G., Wu J.) at http://www.cs.sunysb.edu/~kifer/TechReports/flogic.pdf.

[23] INSPIRE Initial Operating Capability Task Force (IOC TF), http://inspire.jrc.ec.europa.eu/index.cfm/pageid/481.

[24] SOAP Version 1.2, http://www.w3.org/TR/soap/.

[25] WS-Addressing, W3C Member Submission 10 August 2004, http://www.w3.org/Submission/ws-addressing/.

[26] OpenGIS Geography Markup Language, http://www.opengeospatial.org/standards/gml.

[27] INSPIRE Registry, http://inspire-registry.jrc.ec.europa.eu/.

[28] Web Services Interoperability Organisation (WS-I) Basic Profile Version 1.1, http://www.ws-i.org/Profiles/BasicProfile-1.1.html.

[29] Service-Oriented Architecture, http://en.wikipedia.org/wiki/Service-oriented_architecture.

[30] Web Ontology Language – Description Logics (OWL-DL), http://www.w3.org/TR/2004/REC-owl-guide-20040210/#OwlVarieties.

[31] Unified Modeling Language (UML), http://www.uml.org/.

[32] Interactive Instruments' ShapeChange, http://www.interactive-instruments.de/index.php?id=28&L=1.

[33] Electronic Business using eXtensible Markup Language (ebXML), http://www.ebxml.org/.

[34] OGC Web Feature Service (WFS), http://www.opengeospatial.org/standards/wfs.

[35] For WFS-T, see WFS Version 1.1.0, http://portal.opengeospatial.org/files/?artifact_id=8339.

[36] W3C RDF Vocabulary Description Language (RDF Schema) Version 1.0, http://www.w3.org/TR/rdf-schema/.

[37] Web Ontology Language (OWL), http://www.w3.org/TR/owl-features/.

[38] XML Schema Version 1.0, http://www.w3.org/TR/xmlschema-1/.

[39] RIF Overview, http://www.w3.org/TR/rif-overview/.

[40] OMG MetaObject Facility (MOF), http://www.omg.org/mof/.

[41] AtlanMod Transformation Language (ATL), http://www.emn.fr/z-info/atlanmod/index.php/Main_Page.

[42] OMG Query/View/Transformation (QVT) Version 1.0, http://www.omg.org/spec/QVT/1.0/.

[43] XSL Transformation (XSLT) Version 1.0, http://www.w3.org/TR/xslt.

[44] World-Wide Web Consortium (W3C), http://www.w3.org/.

[45] RIF Use Case *Vocabulary Mapping for Data Integration*, http://www.w3.org/TR/rif-ucr/#Vocabulary_Mapping_for_Data_Integration.

[46] RIF Working Group website, http://www.w3.org/2005/rules/wiki/RIF_Working_Group.

[47] *A Classification of Schema Mappings and Analysis of Mapping Tools* (F. Legler, IBM Deutschland Entwicklung GmbH & F. Naumann, Hasso-Plattner-Institut, Potsdam, 2006, see http://www.btw2007.de/paper/p449.pdf.

[48] See RIF-PRD test case available at http://www.w3.org/2005/rules/wiki/Modify_loop.

[49] Rule Interchange Format Core Dialect, http://www.w3.org/TR/rif-core/.

[50] For a general definition of data quality, see http://en.wikipedia.org/wiki/Data_quality

[51] Geographic Profile of the Ontology Mapping Language, see Humboldt project document *A5.2-D3 [3.3] Conceptual Schema Specification and Mapping*, http://community.esdi-humboldt.eu/attachments/50/987-conceptual_schema_specification_and_mapping-fhg-igd-001-final.pdf.

[52] Ontology Mapping Language, http://www.omwg.org/TR/d7/rdf-xml-syntax/.

[53] 1Spatial's Radius Studio Rules Language, see http://www.1spatial.com/products/index.php?ov=2#1269858704773_1/5.

[54] Safe Software's FME Workbench, see http://www.safe.com/products/overview.php.

[55] ISO-19113:2002, Geographic information -- Quality principles, http://www.iso.org/iso/catalogue_detail.htm?csnumber=26018.

[56] INSPIRE Consolidated UML Model, http://inspire.jrc.ec.europa.eu/index.cfm/pageid/541/downloadid/1707.

[57] Microsoft .NET Framework, http://www.microsoft.com/net/.

[58] JAXB 2.0, https://jaxb.dev.java.net/.

[59] Geoserver, http://geoserver.org/display/GEOS/Welcome.

[60] INSPIRE Geoportal, see http://www.inspire-geoportal.eu/.

[61] FreebXML, see http://www.freebxml.org/.

[62] INSPIRE Schema Transformation Network Service Prototype Report.

[63] Accept-Language HTTP Header, see http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html Section 14.4

[64] Xlink: see for a general introduction, http://en.wikipedia.org/wiki/XLink.

[65] Web Services Description Language (WSDL), see http://en.wikipedia.org/wiki/Web_Services_Description_Language.

[66] Google AJAX Search API, http://code.google.com/apis/ajaxsearch/documentation/.

[67] Extended Backus-Naur Form, http://en.wikipedia.org/wiki/Extended_Backus%E2%80%93Naur_Form.

[68] INSPIRE Data Specification on Hydrography – Guidelines, http://inspire.jrc.ec.europa.eu/documents/Data_Specifications/INSPIRE_DataSpecification_HY_v3.0.1.pdf.