infoGrips GmbH Informationssysteme Engineering & Consulting
Technoparkstrasse 1
8005 Zürich
Tel.: 044 / 350 10 10
Fax.: 044 / 350 10 19

Data Gateway Protocol (DGP)

Zusammenfassung

This document describes the Data Gateway Protocol (DGP). The DGP is a system neutral protocol to execute distributed orders over a network of interconnected geodata servers.

Inhaltsverzeichnis

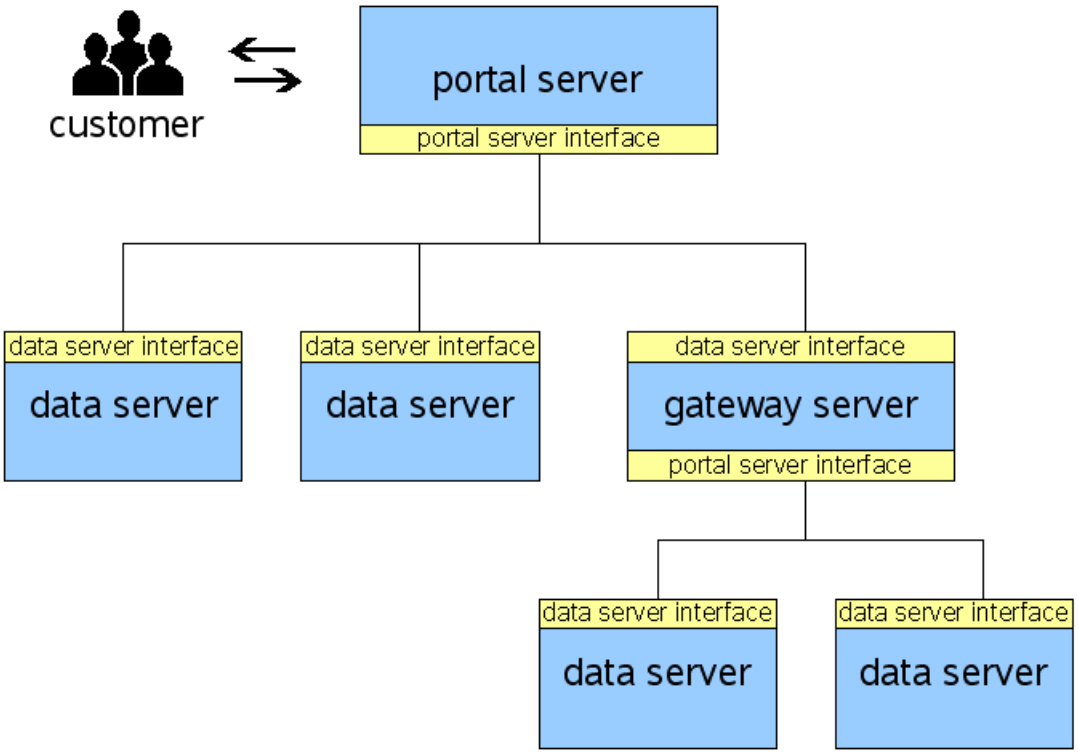
1. Introduction

The DGP (Data Gateway Protocol) is a system neutral protocol based on INTERLIS and SOAP to execute distributed geodata orders over a network of interconnected geodata servers. Due to its system neutral architecture - the DGP is based on INTERLIS and SOAP only - the DGP protocol can be implemented by different system vendors and organizations.

1.1. Network Architecture

To execute data orders over a network of interconnected geodata servers one server has to take the lead to coordinate the activities of the other servers in the network. The leading server is called the **portal server** while the other servers are called **data servers**.

The portal server implements the **portal server interface** while the data servers implement the **data server interface**. A sever can implement both interface types and therefore may act as a **gateway server** in a hierarchical network.



All servers in the network share the same INTERLIS data models. While the data servers contain datasets conforming to the shared INTERLIS data models, the portal server contains only meta data about these datasets. A data server may store datasets internally as files or in a database. No assumption about the internal data representation is made by the DGP.

1.1.1. Portal Server

The following is true about a portal server or gateway servers:

- Is the root (portal server) or an intermediate node (gateway server) in the network.
- Implements the portal server interface.

- May also implement the data server interface to act as an intermediate node (gateway server) in a data network.
- Contains only meta data about the data in the network.
- Distributes price or order requests over the directly connected data servers.
- Merges the partial price or order results of data servers to get the full price information or order result.
- Has a GUI interface to communicate with human users (portal server).

1.1.2. Data Servers

The following statements hold for data servers:

- A data server is a leave node in the data network.
- Each data server contains datasets conforming to the shared INTERLIS data models.
- Processes price or order requests received from a portal server or gateway server.
- Delivers price information or the resulting dataset to the portal server / gateway server.
- Delivers meta data of its datasets to the portal server / gateway server.
- Has no GUI interface.

1.2. Communication between Portal and Data Servers

1.2.1. Communication Objects

Communication between portal and data servers takes place by exchanging communication objects between portal and data servers over SOAP. All communication objects are described by an INTERLIS 2 model (see also appendix A). The following main objects exist:

data product

A data product definition is a well defined transformation of one INTERLIS data model to a final output format (i.e. INTERLIS1 ITF, INTERLIS 2 XTF, DXF, PDF or others). Each data product has a unique name within the data network. A data product may support any number of parameters. At least a range polygon and data selection by topic has to be supported.

order

An order describes the order information for an accepted order. The order dataset is collected by the portal server and contains at least the following information:

- customer data (i.e. name, address, etc.).
- name of the ordered data product.
- complete set of data product parameter values (i.e. range and topics).

With the help of the order dataset a connected data server can process the order by delivering the resulting data set to the portal server.

meta data

Meta data describes a single INTERLIS dataset in a connected data server. In the context of a data network only the following meta data is relevant:

- name of the data model.
- name of the dataset. The dataset name has to be unique for a given data model / data server.
- geographic range covered by the dataset as a closed polygon.

1.2.2. Order Processing Work Flow

Orders are executed as follows:

1. A customer enters the network at the portal server.
2. The portal server shows the user the available data products implemented in the data network.
3. The customer selects the desired data product and enters all required product parameters (i.e. geographic range, topics, etc.).
4. The portal server checks with the meta data, which attached data servers can deliver the requested data product.
5. The portal server executes the price function on any data server and presents the merged price result to the user.
6. The user accepts (next step) or rejects (back to step 2) the order.
7. The order dataset is finalized by the portal server and registered in the order database of the portal sever. The order is send then to any attached data server.
8. The order is processed by the data servers and the result is delivered back to the portal server.
9. Depending on the data product the portal server merges the result of all data servers to produce the final result.
10. The customer is notified by e-mail that the final result is ready for download from the portal server.

1.2.3. Exchange of Meta Data

The central portal server contains only meta data of the connected data servers in the data network. The data servers have to deliver their meta data to the central portal server as follows:

1. When a new INTERLIS dataset becomes available in a data server, the data server notifies the central portal server by sending a meta dataset to the portal server.
2. When an INTERLIS dataset is not available anymore in a data server, the data server sends a delete request to the central portal server. The portal sever deletes the indicated meta dataset.

2. The DGP protocol

2.1. SOAP Encoding

Communication between portal and data servers takes place by exchanging SOAP 1.2 messages over HTTP. The SOAP messages are always grouped as synchronous request / response pairs. A protocol client sends out a request which in turn is answered by a proper response.

Each SOAP message (request or response) contains only a single communication object. The object is send within a single SOAP Body element. The SOAP Header element is not used by the DGP. All possible communication objects are specified by the DGP communication model described in INTERLIS 2 (see also appendix A).

All request messages have to be authenticated. Authentication is done by sending a user / password with each request message. Servers should not accept request, if they are not authenticated correctly.

2.2. Data Server Interface

The following request / response messages have to be implemented by a data server:

2.2.1. CalculatePrice

Request Object

DGP10.DataServer.CalculatePriceRequest.

Response Object

DGP10.DataServer.CalculatePriceResponse.

Response Error Object

DGP10.DataServer.ErrorResponse.

Call Semantics

With CalculatePrice the portal server can request the price of any known data product in the data network. A data server has to process the request and deliver a valid CalculatePriceResponse object or an ErrorResponse. Data servers have take into account that a product has product parameters and that the parameter values may affect the price.

Because the price calculation is done interactively on the portal server the price has to be delivered within 15 seconds by each data server.

Beispiel 1. CalculatePrice Request

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <DGP10.DataServer.CalculatePriceRequest>
      <User>test</User>
      <Password>test</Password>
      <Product>
        <DGP10.DataProduct>
          <ProductName>INTERLIS1</ProductName>
          <ModelName>DM01AVCH24D</ModelName>
          <TopicNames>FixpunkteKategrie1,Liegenschaften</TopicNames>
          <Range>
            <SURFACE>
              <BOUNDARY>
                <POLYLINE>
                  <COORD><C1>480000.000</C1><C2>70000.000</C2></COORD>
                  <COORD><C1>850000.000</C1><C2>70000.000</C2></COORD>
                  <COORD><C1>850000.000</C1><C2>310000.000</C2></COORD>
                  <COORD><C1>480000.000</C1><C2>310000.000</C2></COORD>
                  <COORD><C1>480000.000</C1><C2>70000.000</C2></COORD>
                </POLYLINE>
              </BOUNDARY>
            </SURFACE>
          </Range>
        </DGP10.DataProduct>
      </Product>
    </DGP10.DataServer.CalculatePriceRequest>
  </env:Body>
</env:Envelope>
```

Beispiel 2. CalculatePrice Response

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <DGP10.DataServer.CalculatePriceResponse>
      <Price>85.50</Price>
      <Currency>CHF</Currency>
    </DGP10.DataServer.CalculatePriceResponse>
  </env:Body>
</env:Envelope>
```

2.2.2. ExecuteOrder

Request Object

DGP10.DataServer.ExecuteOrderRequest.

Response Object

DGP10.DataServer.ExecuteOrderResponse

Response Error Object

DGP10.DataServer.ErrorResponse.

Call Semantics

With ExecuteOrder the portal server can order datasets from a connected data server. Each data server has to process the request and deliver a valid ExecuteOrderResponse

object or an ErrorResponse. The ExecuteOrderResponse object contains the requested data as zipped dataset embedded with base64 encoding in the data stream.

There are no response time limits for ExecuteOrderRequest, because a large order request may take a long time to process. The portal server handles this delay by sending the resulting dataset by e-mail to the customer after it has merged all the partial results from the data servers. The customer doesn't have to wait interactively for the finished result.

Beispiel 3. ExecuteOrder Request

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <DGP10.DataServer.ExecuteOrderRequest>
      <User>test</User>
      <Password>test</Password>
      <Order>
        <Customer>
          <Name1>infoGrips GmbH</Name1>
          <Street>Obstgartenstrasse 7</Street>
          <Zip>8035</Zip>
          <City>Z&#252;rich</City>
          <EMail>info@infogrips.ch</EMail>
        </Customer>
        <Product>
          <DGP10.DataProduct>
            <ProductName>INTERLIS1</ProductName>
            <ModelName>DM01AVCH24D</ModelName>
            <TopicNames>FixpunkteKategrie1,Liegenschaften</TopicNames>
            <Range>
              <SURFACE>
                <BOUNDARY>
                  <POLYLINE>
                    <COORD><C1>480000.000</C1><C2>70000.000</C2></COORD>
                    <COORD><C1>850000.000</C1><C2>70000.000</C2></COORD>
                    <COORD><C1>850000.000</C1><C2>310000.000</C2></COORD>
                    <COORD><C1>480000.000</C1><C2>310000.000</C2></COORD>
                    <COORD><C1>480000.000</C1><C2>70000.000</C2></COORD>
                  </POLYLINE>
                </BOUNDARY>
              </SURFACE>
            </Range>
          </DGP10.DataProduct>
        </Product>
      </Order>
    </DGP10.DataServer.ExecuteOrderRequest>
  </env:Body>
</env:Envelope>
```

Beispiel 4. ExecuteOrder Response

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <DGP10.DataServer.ExecuteOrderResponse>
      <Dataset>XTLJILUOLJ808mmkmluOLKJLKJLKLRA...</Dataset>
    </DGP10.DataServer.ExecuteOrderResponse>
  </env:Body>
</env:Envelope>
```

2.3. Portal Server Interface

The following request / response messages have to implemented by the portal server:

2.3.1. MetadataUpload

Request Object

DGP10.PortalServer.MetadataUploadRequest.

Response Object

DGP10.PortalServer.MetadataUploadResponse

Response Error Object

DGP10.PortalServer.ErrorResponse.

Call Semantics

Notifies the portal server that a new dataset is available on the dataserver. The dataset name has to be unique within the datasets of the same data model / data server.

Beispiel 5. MetadataUpload Request

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
     <DGP10.PortalServer.MetadataUploadRequest>
        <User>test</User>
        <Password>test</Password>
        <MetaData>
          <DGP10.MetaData>
            <ModelName>DM01AVCH24D</ModelName>
            <DatasetName>test</DatasetName>
            <Range>
              <SURFACE>
                <BOUNDARY>
                  <POLYLINE>
                    <COORD><C1>480000.000</C1><C2>70000.000</C2></COORD>
                    <COORD><C1>850000.000</C1><C2>70000.000</C2></COORD>
                    <COORD><C1>850000.000</C1><C2>310000.000</C2></COORD>
                    <COORD><C1>480000.000</C1><C2>310000.000</C2></COORD>
                    <COORD><C1>480000.000</C1><C2>70000.000</C2></COORD>
                  </POLYLINE>
                </BOUNDARY>
              </SURFACE>
            </Range>
          </DGP10.MetaData>
        </MetaData>
     </DGP10.PortalServer.MetadataUploadRequest>
  </env:Body>
</env:Envelope>
```

Beispiel 6. MetadataUpload Response

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <DGP10.PortalServer.MetadataUploadResponse>
    </DGP10.PortalServer.MetadataUploadResponse>
  </env:Body>
</SOAP-ENV:Envelope>
```

2.3.2. MetadataDelete

Request Object

DGP10.PortalServer.MetadataDeleteRequest.

Response Object

DGP10.PortalServer.MetadataDeleteResponse

Response Error Object

DGP10.PortalServer.ErrorResponse.

Call Semantics

Requests from the portal server to delete a previously uploaded meta data dataset.

Beispiel 7. MetadataDelete Request

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
     <DGP10.PortalServer.MetadataDeleteRequest>
        <User>test</User>
        <Password>test</Password>
        <ModelName>DM01AVCH24D</ModelName>
        <DatasetName>test</DatasetName>
     </DGP10.PortalServer.MetadataDeleteRequest>
  </env:Body>
</env:Envelope>
```

Beispiel 8. MetadataDelete Response

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <DGP10.PortalServer.MetadataDeleteResponse>
    </DGP10.PortalServer.MetadataDeleteResponse>
  </env:Body>
</env:Envelope>
```

A. DGP Communication Model

All possible DGP objects are described by the following INTERLIS 2 data model:

```
INTERLIS 2.2;

MODEL DGP10 =

  DOMAIN

    SwissCoord = COORD
      480000.000 .. 850000.000,
      70000.000 .. 310000.000,
      ROTATION 2 -> 1;

    STRUCTURE DataProduct =
      !! name of data product, i.e. INTERLIS1, INTERLIS2
      ProductName: MANDATORY TEXT*20;
      !! data model name
      ModelName: MANDATORY TEXT*80;
      !! selected topics as comma separated list or *
      TopicNames: MANDATORY TEXT*1024;
      !! geographic range as polygon
      Range: MANDATORY SURFACE WITH (STRAIGHTS) VERTEX SwissCoord;
    END DataProduct;

    STRUCTURE Customer =
      Name1: MANDATORY TEXT*40;
      Name2: TEXT*40;
```

```
      Street: MANDATORY TEXT*80;
      Zip: MANDATORY TEXT*20;
      City: MANDATORY TEXT*40;
      EMail: MANDATORY TEXT*40;
    END Customer;

    STRUCTURE Order =
      Customer: MANDATORY Customer;
      Product: MANDATORY DataProduct;
    END Order;

    STRUCTURE MetaData =
      ModelName: MANDATORY TEXT*255;
      !! DatasetName has to be unique for a given DataServer / ModelName.
      DatasetName: MANDATORY TEXT*255;
      !! Range is a bounding polygon around the dataset
      Range: MANDATORY SURFACE WITH (STRAIGHTS) VERTEX SwissCoord;
    END MetaData;

    CLASS Request (ABSTRACT) =
      User: MANDATORY TEXT*20;
      Password: MANDATORY TEXT*20;
    END Request;

    CLASS Response (ABSTRACT) =
    END Response;

  TOPIC DataServer =

    !! CalculatePrice Message
    CLASS CalculatePriceRequest EXTENDS Request =
      Product: MANDATORY DataProduct;
    END CalculatePriceRequest;
    CLASS CalculatePriceResponse EXTENDS Response =
      Price: MANDATORY 0.0 .. 1000000.0;
      Currency: MANDATORY TEXT*3; !! always CHF at the moment
    END CalculatePriceResponse;

    !! ExecuteProduct Message
    CLASS ExecuteOrderRequest EXTENDS Request =
      Order: MANDATORY Order;
    END ExecuteOrderRequest;
    CLASS ExecuteOrderResponse EXTENDS Response =
      !! Dataset contains the dataset zipped and base64 encoded.
      !! Dataset will be modelled by BLACKBOX BINARY with INTERLIS 2.3.
      Dataset: MANDATORY TEXT*1024;
    END ExecuteOrderResponse;

    !! Error Messages
    CLASS ErrorResponse EXTENDS Response =
      Message: MANDATORY TEXT*1024;
    END ErrorResponse;

  END DataServer;

  TOPIC PortalServer =

    !! MetadataUpload Message
```

```
    CLASS MetadataUploadRequest EXTENDS Request =
      Metadata: MANDATORY MetaData;
    END MetadataUploadRequest;
    CLASS MetadataUploadResponse EXTENDS Response =
    END MetadataUploadResponse;

    !! MetadataDelete Message
    CLASS MetadataDeleteRequest EXTENDS Request =
      ModelName: MANDATORY TEXT*255;
      DatasetName: MANDATORY TEXT*255;
    END MetadataDeleteRequest;
    CLASS MetadataDeleteResponse EXTENDS Response =
    END MetadataDeleteResponse;

    !! Error Message
    CLASS ErrorResponse EXTENDS Response =
      Message: MANDATORY TEXT*1024;
    END ErrorResponse;

  END PortalServer;

END DGP10.
```

B. Terminology

data network

A data network is a network of interconnected data servers and one central portal server. Each server in a data network knows the same INTERLIS data models.

data model

A data model described with INTERLIS 1 or INTERLIS 2.

data server

A data server contains all necessary datasets to produce a data product. The datasets are stored in the data server as files or in a database.

portal server

A portal server is the center of a network of data servers. While the data servers contain datasets to produce data products, the portal server only contains meta data about the data servers. This meta data helps the portal server to find the proper data servers for a given order.

data product

A data product is a well defined transformation of one or more INTERLIS data models to a final output format (i.e. ITF, XTF, DXF, PDF or others). Each data product has a unique name within the data network. The INTERLIS => Output Format transformation has to be implemented on each data server.

data product parameters

A data product may support any number of parameters. Each data server should at least support topic names and a range polygon as parameters.

price function

The price function calculates the price for a given data product. It therefore supports the same parameters as the data product. While the data product maps one data model to a destination format, the price function maps one data model to a calculated price. More then one data product may share the same price function.

order

An order is a dataset describing the order information for an accepted order. The order is collected by the portal server and has at least the following attributes:

- customer information (i.e. name, address, etc.).
- name of the ordered data product
- complete set of data product parameter values.

With the help of the order dataset an attached data server can calculate the price of an order or process the order by delivering the resulting data set.

meta data

Meta data describes a single INTERLIS dataset in a connected data server. In the context of a data network only the following meta data is relevant:

- name of the data model.
- name of the dataset. The dataset name has to be unique for a given data model.
- geographic range covered by the dataset as a closed polygon.