

|  |  |
| --- | --- |
|  | **imotion** |
|  |  |
|  | Protran 2.x Documentation |
|  |  |
|  | Configuration |
|  |  |
|  | Technical Description |
|  |  |
|  | Version: 5.1 |
|  | State: Draft |
|  | Classification: Internal use only |
|  | Author: WES |
|  | Creation date: 2013-07-03 |
|  | Repository : $/Gorba/Main/Motion/Protran/Documents/TD\_ProtranDocumentation/ TD\_Protran2Documentation.docx |
|  | Gorba AG  Sandackerstrasse  9245 Oberbüren  Switzerland |

**Table of contents**

[1 Introduction 8](#_Toc423078629)

[1.1 Protran Version 8](#_Toc423078630)

[1.2 Purpose 8](#_Toc423078631)

[1.3 Intended Audience and Reading Suggestions 8](#_Toc423078632)

[1.4 Remarks 8](#_Toc423078633)

[2 System Overview 8](#_Toc423078634)

[3 Configurations 9](#_Toc423078635)

[3.1 Global configuration conventions 9](#_Toc423078636)

[3.1.1 Durations and time intervals 9](#_Toc423078637)

[3.2 protran.xml 9](#_Toc423078638)

[3.2.1 Protocol configuration 9](#_Toc423078639)

[3.2.2 Persistence configuration 10](#_Toc423078640)

[3.3 IBIS.XML 12](#_Toc423078641)

[3.3.1 Protran’s Behaviour 12](#_Toc423078642)

[3.3.2 Sources 15](#_Toc423078643)

[3.3.3 IBIS Recording 21](#_Toc423078644)

[3.3.4 Time Synchronization 22](#_Toc423078645)

[3.3.5 IBIS Telegrams 24](#_Toc423078646)

[3.4 vdv301.xml 25](#_Toc423078647)

[3.4.1 Explanation of the general structure 25](#_Toc423078648)

[3.4.2 @XmlDoc(xsd=..\..\..\..\Common\Configuration\Source\Protran\VDV301\vdv301.xsd;xml=..\..\Source\Vdv301\vdv301.xml) 27](#_Toc423078649)

[3.5 io.xml 28](#_Toc423078650)

[3.5.1 Serial port configuration 28](#_Toc423078651)

[3.5.2 Input mapping 29](#_Toc423078652)

[3.5.3 Transformations 29](#_Toc423078653)

[3.6 Arriva.XML 30](#_Toc423078654)

[3.6.1 FTP Tag 30](#_Toc423078655)

[3.6.2 Behaviour Tag 31](#_Toc423078656)

[4 IBIS Telegrams 33](#_Toc423078657)

[4.1 Supported Telegrams 34](#_Toc423078658)

[4.1.1 DS001 35](#_Toc423078659)

[4.1.2 DS001a 35](#_Toc423078660)

[4.1.3 DS002 35](#_Toc423078661)

[4.1.4 DS003 36](#_Toc423078662)

[4.1.5 DS003a 36](#_Toc423078663)

[4.1.6 DS003c 36](#_Toc423078664)

[4.1.7 DS005 37](#_Toc423078665)

[4.1.8 DS006 37](#_Toc423078666)

[4.1.9 DS006a 37](#_Toc423078667)

[4.1.10 DS008 38](#_Toc423078668)

[4.1.11 DS009 38](#_Toc423078669)

[4.1.12 DS010 38](#_Toc423078670)

[4.1.13 DS010b 38](#_Toc423078671)

[4.1.14 DS010j 39](#_Toc423078672)

[4.1.15 DS020 39](#_Toc423078673)

[4.1.16 DS021 39](#_Toc423078674)

[4.1.17 DS021a 40](#_Toc423078675)

[4.1.18 DS021c 41](#_Toc423078676)

[4.1.19 DS036 41](#_Toc423078677)

[4.1.20 DS080 42](#_Toc423078678)

[4.1.21 DS081 42](#_Toc423078679)

[4.1.22 GO001 42](#_Toc423078680)

[4.1.23 GO002 43](#_Toc423078681)

[4.1.24 GO003 43](#_Toc423078682)

[4.1.25 GO004 44](#_Toc423078683)

[4.1.26 GO005 44](#_Toc423078684)

[4.1.27 GO006 45](#_Toc423078685)

[4.1.28 GO007 45](#_Toc423078686)

[4.1.29 HPW074 45](#_Toc423078687)

[4.2 DS001 46](#_Toc423078688)

[4.3 DS001a 46](#_Toc423078689)

[4.4 DS002 47](#_Toc423078690)

[4.5 DS003 47](#_Toc423078691)

[4.6 DS003a 47](#_Toc423078692)

[4.7 DS003c 48](#_Toc423078693)

[4.8 DS005 48](#_Toc423078694)

[4.9 DS006 49](#_Toc423078695)

[4.10 DS006a 50](#_Toc423078696)

[4.11 DS008 51](#_Toc423078697)

[4.12 DS009 51](#_Toc423078698)

[4.13 DS010 52](#_Toc423078699)

[4.14 DS010b 52](#_Toc423078700)

[4.15 DS010j 52](#_Toc423078701)

[4.16 DS020 53](#_Toc423078702)

[4.17 DS021 54](#_Toc423078703)

[4.18 DS021a 57](#_Toc423078704)

[4.18.1 Telegram Collection 58](#_Toc423078705)

[4.18.2 Stop Information 58](#_Toc423078706)

[4.18.3 Destination Information 59](#_Toc423078707)

[4.18.4 Travel times calculation 59](#_Toc423078708)

[4.18.5 Status Values 59](#_Toc423078709)

[4.18.6 DS021a ‘A‘ variation 60](#_Toc423078710)

[4.18.7 DS021A ‘S‘ variation 61](#_Toc423078711)

[4.19 DS021c 62](#_Toc423078712)

[4.19.1 Telegram Collection 63](#_Toc423078713)

[4.19.2 Stop Information 63](#_Toc423078714)

[4.19.3 Special stop index values 63](#_Toc423078715)

[4.19.4 Destination Information 64](#_Toc423078716)

[4.20 DS036 64](#_Toc423078717)

[4.21 DS080 65](#_Toc423078718)

[4.22 DS081 65](#_Toc423078719)

[4.23 GO001 66](#_Toc423078720)

[4.24 GO002 67](#_Toc423078721)

[4.24.1 Telegram Parsing 68](#_Toc423078722)

[4.24.2 Telegram Collection 68](#_Toc423078723)

[4.24.3 Connection Information 69](#_Toc423078724)

[4.25 GO003 70](#_Toc423078725)

[4.26 GO004 71](#_Toc423078726)

[4.27 GO005 72](#_Toc423078727)

[4.27.1 Telegram Collection 75](#_Toc423078728)

[4.27.2 Stop Information 75](#_Toc423078729)

[4.27.3 Destination Information 75](#_Toc423078730)

[4.27.4 Route Buffering 75](#_Toc423078731)

[4.27.5 Route Deletion 76](#_Toc423078732)

[4.28 GO006 76](#_Toc423078733)

[4.29 GO007 77](#_Toc423078734)

[4.29.1 Telegram Collection 77](#_Toc423078735)

[4.29.2 Stop and connection Information 77](#_Toc423078736)

[4.29.3 Destination Information 78](#_Toc423078737)

[4.29.4 Route Change 78](#_Toc423078738)

[4.30 HPW074 78](#_Toc423078739)

[4.31 Common Sub-Tags 80](#_Toc423078740)

[4.31.1 Generic Usage 80](#_Toc423078741)

[4.31.2 IBIS Answers 80](#_Toc423078742)

[5 Telegram’s Transformations 82](#_Toc423078743)

[5.1 Capitalize Transformation 83](#_Toc423078744)

[5.2 ChainRef 84](#_Toc423078745)

[5.3 Integer Transformation 84](#_Toc423078746)

[5.4 Join Transformation 85](#_Toc423078747)

[5.5 LawoString Transformation 85](#_Toc423078748)

[5.6 RegexMapping Transformation 86](#_Toc423078749)

[5.7 RegexDivider Transformation 87](#_Toc423078750)

[5.8 Replace Transformation 88](#_Toc423078751)

[5.9 StringMapping Transformation 88](#_Toc423078752)

[6 Regular Expressions in Telegram Transformations 89](#_Toc423078753)

[6.1 Splitting words 89](#_Toc423078754)

[6.1.1 Advanced example 90](#_Toc423078755)

[6.2 Splitting by special characters 90](#_Toc423078756)

[6.3 Splitting DS021a, DS021c and GO005 90](#_Toc423078757)

[6.3.1 Advanced examples 90](#_Toc423078758)

[6.4 Removing leading or trailing characters 92](#_Toc423078759)

[6.5 Format date or time 92](#_Toc423078760)

[7 Telegram Usage 93](#_Toc423078761)

[7.1 DS001 Example Usage 94](#_Toc423078762)

[7.2 DS005 Example Usage 94](#_Toc423078763)

[7.3 DS006 Example Usage 95](#_Toc423078764)

[7.4 DS010B Example Usage 95](#_Toc423078765)

[7.5 DS020 Example Usage 95](#_Toc423078766)

[7.6 DS021A Example Usage 96](#_Toc423078767)

[7.7 GO003 Example Usage 98](#_Toc423078768)

[7.8 HPW074 Example Usage 99](#_Toc423078769)

[8 Persistence Service 100](#_Toc423078770)

[8.1 Validity Period 100](#_Toc423078771)

[8.2 Information Saved 100](#_Toc423078772)

[8.2.1 DS021a/c, GO005 100](#_Toc423078773)

[8.2.2 GO002 100](#_Toc423078774)

[8.2.3 FTP File List for Abu Dhabi 100](#_Toc423078775)

[9 Arriva departures 101](#_Toc423078776)

[10 Appendix A: Configuration migration 102](#_Toc423078777)

[10.1 Migration from Protran 1.x to 2.x 102](#_Toc423078778)

[10.1.1 cfg.xml to protran.xml 102](#_Toc423078779)

[10.1.2 ibis.xml 102](#_Toc423078780)

[10.1.3 arriva.xml 103](#_Toc423078781)

[10.1.4 AbuDhabi.xml 103](#_Toc423078782)

[10.1.5 New io.xml 103](#_Toc423078783)

**Modification management**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Dept.** | **Modifications** | **State** |
| 0.1 | 2013-07-05 | WES | SW | Initial version based on version 10.0 of Configuration 1.x document  Included chapter about migration from 1.x to 2.x and updated cfg.xml and ibis.xml chapters | draft |
| 0.2 | 2013-09-17 | WES | SW | Added UDPServer source configuration  Added DS003c, DS006a and DS009 | draft |
| 0.3 | 2013-09-24 | WES | SW | Removed code conversion for DS001a | draft |
| 0.4 | 2013-10-22 | WES | SW | Added DeletePassedStops to GO002 | draft |
| 0.5 | 2013-10-24 | WES | SW | Added remarks about DS009a/b | draft |
| 0.6 | 2013-10-25 | WES | SW | Added DS008 | draft |
| 1.1 | 2014-02-05 | RAN/WES | SW | Modified chapter 5.1 about Capitalize transformation | draft |
| 1.2 | 2014-02-21 | RAN | SW | Modified examples in chapters 4.16, 5.3, 5.4, 5.7, 5.8, 5.9, 7.6 | draft |
| 1.3 | 2014-06-02 | WES | SW | Added DS021, removed answers sub-tags in telegrams that don’t have answers | draft |
| 1.4 | 2014-06-17 | WES | SW | Added chapter about vdv301.xml configuration | draft |
| 1.5 | 2014-06-17 | RAN | SW | Minor sentence correction to vdv301.xml chapter | draft |
| 2.1 | 2014-07-23 | RAN | SW | Added details about the telegrams under section 4.1 | draft |
| 2.2 | 2014-08-14 | WES | SW | Updated telegrams details (in ch. 4.1) | draft |
| 2.2 | 2014-08-29 | AMR | PM | Changes from examples | draft |
| 2.3 | 2014-09-01 | RAN | SW | Minor changes to chapter 4.1 | draft |
| 3.1 | 2014-01-27 | WES | SW | Changed IBIS time sync since it is now done through GIOoM (and Hardware Manager) and not Ximple | draft |
| 3.2 | 2015-02-11 | RAN | SW | Added chapter about short GO002 telegram with support for DS010j | draft |
| 3.3 | 2015-04-09 | RAN | SW | Added chapter about GO007 telegram. Updated DS009 telegram | draft |
| 3.4 | 2015-04-09 | WES | SW | Fixed GO007 and referenced it also from DS021 and GO005 | draft |
| 4.1 | 2015-06-19 | EPT | SW | Added “ShowPastStops” to chapters 4.18, 4.19 and 4.26  Updated telegram details of DS036 in chapter 4.1.19 | draft |
| 4.2 | 2015-06-23 | WES | SW | Added chapter for DS036 | draft |
| 5.1 |  | WES | SW | Added section about VDV 301 | draft |

**Review**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Dept.** | **Remarks** |
| 0.1 | 2013-07-08 | RAN | SW | Minor changes/corrections |
| 0.6 | 2014-01-08 | WES | SW | Formal review |
| 2.3 | 2014-11-10 | WES | SW | Reviewed, a few things fixed |
| 3.2 | 2015-04-09 | RAN | SW | Reviewed |
| 4.2 | 2015-06-23 | WES | SW | Reviewed |

**Release**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Dept.** | **Remarks** |
| 1.0 | 2014-01-08 | WES | SW | First version for Protran 2.0.1402 |
| 2.0 | 2014-06-26 | RAN | SW | New release matching Protran 2.2.1426 |
| 3.0 | 2014-11-10 | WES | SW | New release matching Protran 2.4.1445 |
| 4.0 | 2015-04-09 | RAN | SW | New release matching Protran 2.4.1515 |
| 5.0 | 2015-06-23 | WES | SW | New release matching Protran 2.4.1525 |

# Introduction

## Protran Version

C:\Users\wes\AppData\Local\Microsoft\Windows\INetCache\Content.Word\protan.emfThis document covers Protran version **2.4.1525**.

## Purpose

The goal of this document is to describe in detail each Protran parameter in order to configure it properly.

At the end of this document, the reader will have all the details to configure Protran.

## Intended Audience and Reading Suggestions

This document is addressed to product managers or customer project managers that are familiar with this product and are able to install, operate and maintain it.

## Remarks

This document covers all the Protran’s configuration files and parameters for the current version (see above). It’s possible that in future the configuration files will be enhanced but for backward compatibility issues, their contents will never be changed.

# System Overview

Protran is an application capable to connect itself with one or more remote computer through Ethernet or a serial connection. Also, Protran interacts with another Gorba’s application, called MediServer, in order to send information to the Gorba’s media player.

Basically, an overview of the typical scenario is presented below:

Gorba System

Infomedia

Protran

Remote Computer

MediServer

TCP/IP or Serial

Figure 2‑1 Typical System’s Overview

Protran has to be configured correctly in order to establish a valid connection with the external remote computer.

# Configurations

Protran needs different configuration files depending on the characteristics of the remote computer. If the remote computer is reachable by an ibis bus, are required to be configured both the file “**protran.xml**” and the file “**ibis.xml**”.

Other protocols require different configuration files besides “**protran.xml**”. Usually the configuration file is named after the protocol or project (e.g. “**VDV301.xml**”, “**AbuDhabi.xml**”, “**Arriva.xml**”, …)

The file “**dictionary.xml**” is always mandatory to be configured. All the log activities are managed by the file “**NLog.config**” independently from the other configuration files.

## Global configuration conventions

All our configuration files follow some conventions.

### Durations and time intervals

All durations and time intervals are configured using the XSD duration format.

The time interval is specified in the following form "PnYnMnDTnHnMnS" where:

* P indicates the period (required)
* nY indicates the number of years
* nM indicates the number of months
* nD indicates the number of days
* T indicates the start of a time section (required if you are going to specify hours, minutes, or seconds)
* nH indicates the number of hours
* nM indicates the number of minutes
* nS indicates the number of seconds (decimal numbers are allowed)

Examples:

* 1 year = P1Y
* 1 day 6 hours = P1DT6H
* 10 seconds = PT10S
* 400 milliseconds = PT0.4S

## protran.xml

The file “protran.xml” has to be stored in the configuration directory of Protran or next to “Protran.exe”, otherwise Protran will immediately exit with an error message.

The “protran.xml” file contains different XML tags. They can be grouped in:

* Protocol configuration
* Persistence configuration

### Protocol configuration

These tags specify the kind of protocol that Protran has to “translate” (the protocol that “belongs” to the remote computer, indeed).  
Hereafter is shown the referring section in the “cfg.xml”.

<?xml version="1.0"?>

<Protran Version="2.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <Protocols>

    <Protocol Name="IbisProtocol" Enabled="true" description="This is the IBIS protocol"/>

    <Protocol Name="IOProtocol" Enabled="true" description="This is the I/O protocol"/>

  </Protocols>

  <Persistence description="Allows Protran to store/load information in/from the hard disk in order

    to reuse them in case of restart.">

    <IsEnabled desc="Flag that indicates whether the persistence service is enabled or not.

      Admitted values are: true, false.

      Default value: false">false</IsEnabled>

    <PersistenceFile description="Absolute file name for the file that

      will be used by Protran for the stored information.">persistence.xml</PersistenceFile>

    <DefaultValidity description="The default amount of seconds

      for the information's validity. Default is 10 minutes.">PT10M</DefaultValidity>

  </Persistence>

</Protran>

Figure 3‑1 A complete protran.xml file

Hereafter will be provided a description of all of them.

#### <Protocol>

This tag contains all the name required by Protran to deal with the remote board computer. It doesn’t contain any sub-tags, but requires the following attributes:

* Name: the name of the protocol; the following values are currently supported:
  + IbisProtocol
  + VDV301
  + IOProtocol
  + AbuDhabiProtocol
  + ArrivaProtocol
* Enabled: one of “true” or “false”; you don’t need to explicitly enable protocols, but this attribute can be used to disable one protocol for testing reasons

Multiple protocol tags are possible inside the protocol definition. All configured and enabled protocols will be started by Protran.

### Persistence configuration

Basically, these tags tell Protran “where” it can find information about the “Generic View” feature and about the “Persistence” feature. This last feature allows Protran to store/load specific information into/from a file in order to reuse them in case in case of restart.

#### <Persistence><IsEnabled>

Tells if the “Persistence” feature has to be enabled or not.  
The admitted values are: true, false.  
You have to specify “true” to enable it, “false” to disable it.  
By default, the “Persistence” is disabled.  
This XML tag doesn’t support any XML sub-tag. It supports only the attribute “description” in which is provided a brief explanation about its meaning.

#### <Persistence><PersistenceFile>

Specify the absolute file name of the file that will be used by Protran to write the information and to load information at its startup. If the “Persistence” feature is enabled but the file doesn’t exist, Protran will create and use the default one: Persistence.xml (stored in the same directory of the Protran’s executable). This XML tag can accept alphanumeric string.  
This XML tag doesn’t support any XML sub-tag. It supports only the attribute “description” in which is provided a brief explanation about its meaning.

#### <Persistence><DefaultValidity>

Specify the default amount of time (expressed as an XML duration) with which consider valid the information stored in the “Persistence” file. Suppose that some information are stored in the file with a default validity time equal to one minute and suppose to restart Protran after some hours. At its startup Protran will notice that the information are too much outdated and therefore it will reject them.  
The admitted values are positive durations.  
The default value is “PT10M” (10 minutes).  
This XML tag doesn’t support any XML sub-tag. It supports only the attribute “description” in which is provided a brief explanation about its meaning.

## IBIS.XML

Ibis.xml is an additional file for Protran used whenever is specified the **IbisProtocol**.

Attention: it’s strictly required that the file “ibis.xml” is stored in the same directory in which is stored “protran.xml”. Otherwise Protran will exit with an error message.

The file “ibis.xml” can be thought as the container of the following groups of:

* General information about the **Protran’s behavior** (for example, its behavior regarding the parser, timeouts and so on…)
* Information about the **source** where data from the IBIS master is received
* Information about the **recording** capabilities of Protran regarding the IBIS telegrams received from the IBIS master
* Information about how Protran has to deal the **time synchronization** with the IBIS master
* Information about how to react in case of specific **telegrams** coming from the IBIS master.
* Information about the **translation** from the IBIS telegrams to the Gorba’s protocol (XIMPLE).

In the following paragraphs, group by group, will be detailed each tag contained into the file “ibis.xml”. The XML root tag of this file is <Ibis> (case sensitive).

### Protran’s Behaviour

There are some general aspects of Protran that must be defined into the configuration file. Some aspects are related to its way to interact with the IBIS master, others on its “speed” and others about time issues.

A complete example is show below:

<Ibis>

<Behaviour desc="Container of the settings about all the software's behaviours.">

<IbisAddress desc="Values admitted: {1 ... 32} Def. 8">10</IbisAddress>

<ConnectionTimeOut desc="values admitted: {1 ... 255} in seconds. Def. 60">60</ConnectionTimeOut>

<CheckCrc descr="values admitted: {true, false} (case insensitive) Def. true">true</CheckCrc>

<ByteType descr="values admitted: {Ascii7, UnicodeBigEndian}">Ascii7</ByteType>

<ProcessPriority descr="Gets or sets the priority of the Protran process in case IBIS is used.

Values admitted: {AboveNormal, BelowNormal, High, Idle, Normal, RealTime }.

Default value is AboveNormal">AboveNormal</ProcessPriority>

<ConnectionStatusUsedFor Table="SystemStatus" Column="RemotePC" Row="0"/>

    <SpeakersVolume desc="The percentage level to be applied to the speakers.

    Admitted values are integers from 0 to 100.

    The value 0 represents the lowest volume (speakers silenced).

    The value 100 represents the highest volume.

    The default value is 100.">50</SpeakersVolume>

</Behaviour>

</Ibis>

Figure 3‑2 Example of Protran’s generic behaviour configuration in IBIS.XML

In the example above it is specified that Protran has to consider as proper, the IBIS telegrams having an address equal to 10, also it has to consider as “inactive” the IBIS master if this doesn’t send nothing for 60 seconds, also it has to check the CRC code in each IBIS telegram received, the telegrams are ASCII based, also Protran has to run itself with more priority than a normal application and finally, each time Protran has to communicate the IBIS master status to the Gorba’s media player, it has to use the translation rules specified by the triple Table="SystemStatus" Column="RemotePC" Row="0" (for a detailed explanation of the translation rules, see chapter 7).

#### <Behaviour>

This tag contains all the information about the Protran’s general behavior. It’s a container of other sub-tags so it doesn’t require any value.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| Behaviour | * IbisAddres (more than one are allowed) * ConnectionTimeOut * CheckCrc * ByteType * ProcessPriority * ConnectionStatusUsedFor | desc just a little description about this tag. |

Table ‑ Behaviour content

#### <Behaviour><IbisAddress>

This tag contains an integer that represents the IBIS address that Protran has to consider in the IBIS telegrams.  
The admitted values are the integers between 1 and 32.  
The default value is 8.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| IbisAddress | None | desc just a little description about this tag. |

Table ‑ IbisAddress content

#### <Behaviour><ConnectionTimeOut>

This tag contains an integer that represents the amount of seconds after which Protran consider “inactive” the IBIS master in case of complete silence from it.  
The admitted values are the integers between 1 and 255.  
The default value is 60 seconds.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| ConnectionTimeOut | None | desc just a little description about this tag. |

Table ‑ ConnectionTimeOut content

#### <Behaviour><CheckCrc>

This tag contains a string that tells if Protran has to check the CRC for each IBIS telegram received or not.  
The admitted values are: true or false. “True” is to enable to CRC check, “false” is to disable it.  
The default value is true.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| CheckCrc | None | desc just a little description about this tag. |

Table ‑ CheckCrc content

#### <Behaviour><ByteType>

This tag contains a string that tells which kind of encoding is used by the IBIS master to create/read telegrams. For example, if the IBIS master treats the telegrams in Ascii (7 bit) Protran has to do the same, otherwise it will not be able to recognize any telegram.  
The admitted values are: Ascii7, UnicodeBigEndian.  
The default value is Ascii7.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| ByteType | None | desc just a little description about this tag. |

Table ‑ ByteType content

#### <Behaviour><ProcessPriority>

This tag contains a string that represents the priority level of Protran compared to a normal running application. Due to the fact that the IBIS master wants answer within small timing intervals, Protran has to run with more priority. If it is served by the O.S as a normal application, it might be possible that it can’t answer rapidly to the IBIS master.  
The admitted values are: AboveNormal, BelowNormal, High, Idle, Normal, RealTime.  
The default value is AboveNormal.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| ProcessPriority | None | desc just a little description about this tag. |

Table ‑ ProcessPriority content

#### <Behaviour><ConnectionStatusUsedFor>

This tag contains the rules that Protran has to respect whenever it has to inform the Gorba’s media player about changes on the IBIS master’s status. To communicate this kind of information to the Gorba’s media player, Protran has to specify a specific position inside the media player’s database; otherwise the media player will not recognize it well. The position inside this database is uniquely identified by the triple {Table, Column, Row} and these are the only things to configure for this tag (a more detailed explanation about the translation in a database’s coordinate, see the chapter 0).

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| ConnectionStatusUsedFor | None | * Table * Column * Row   Table and Column contain a string, Row contains an integer bigger or equal than 0, embraced or not embraced by curly brackets. |

Table ‑ ConnectionStatus content

### Sources

The following sources are available:

* Simulation: reads IBIS telegrams from a pre-recorded log file (several formats are supported) and simulates thus the behavior of Protran with an IBIS master
* Serial Port: reads IBIS telegrams as they arrive from the IBIS master through the Wagenbus interface
* UDP Server: creates a UDP/IP server that listens to telegrams received from the IBIS master. This feature is experimental.

#### IBIS Simulation

Protran is capable to simulate what an IBIS master did with it in a previously recorded session. Basically, a simulation consists in loading a file in which Protran has stored all the telegrams received from the IBIS master or loading a file with the extension “.PRO.CSV” or “.ISM”. These last two file types are created by other applications (for example, WBMonitor.exe creates the .PRO.CSV), and Protran just use them to simulate a conversation with a virtual IBIS master.

The configurable tags for this feature are the following:

* <Sources><Simulation><Active>
* <Sources><Simulation><SimulationFile>
* <Sources><Simulation><InitialDelay>
* <Sources><Simulation><IntervalBetweenTelegrams>
* <Sources><Simulation><TimesToRepeat>

A complete example is shown below:

<Sources>

<Simulation desc="Container of all the information about the IBIS simulation.">

<SimulationFile desc="Free text (case insensitive) Def.  
 ibis.log">./ibisSimulation.log</SimulationFile>

<InitialDelay desc="The initial delay">PT5S</InitialDelay>

<IntervalBetweenTelegrams desc="Time between sending two telegrams.  
      No value means that the simulation will follow specific time stamps. Def. empty  
 ">PT1S</IntervalBetweenTelegrams>

<TimesToRepeat desc="Values admitted: {0, max int} 0 means infinite times. Def.   
 1">2</TimesToRepeat>

</Simulation>

</Sources>

Figure 3‑3 Example of simulation configuration in IBIS.XML

In the example above, is active a simulation based on file “ibisSimulation.log”. Protran will search for the file next to ibis.xml. The simulation starts after 5 seconds after the user’s confirmation (pressing ENTER on the console), also the simulation will respect the same timing used during the recorded session, the simulation will be repeated 2 times and doesn’t respect the Gismo standard.

##### <Sources><Simulation>

This tag contains all the information about a simulation. It’s a container of other sub-tags so it doesn’t require any value.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| Simulation | * SimulationFile * InitialDelay * IntervalBetweenTelegrams * TimesToRepeat | desc just a little description about this tag. |

Table ‑ Simulation content

##### <Sources><Simulation><SimulationFile>

This tag contains the absolute path of the file that has to be loaded. If the path is relative, it is expected to be relative to the ibis.xml config file.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| SimulationFile | None | desc just a little description about this tag. |

Table ‑ SimulationFile content

##### <Sources><Simulation><InitialDelay>

This tag contains an XML duration that represents the amount of time to wait before the real execution of the simulation.  
The admitted values are positive durations.  
The default value is PT0S.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| InitialDelay | None | desc just a little description about this tag. |

Table ‑ InitialDelay content

##### <Sources><Simulation><IntervalBetweenTelegrams>

This tag contains an XML duration that represents the amount of time to wait between a simulated IBIS telegram and the next one.  
The admitted values are positive durations. If this element is not present (or empty), Protran will use the timestamp stored in the simulation file (if compliant with its parsing rules).  
The default value is emtpy.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| IntervalBetweenTelegrams | None | desc just a little description about this tag. |

Table ‑ IntervalBetweenTelegrams content

##### <Sources><Simulation><TimesToRepeat>

This tag contains an integer that represents the amount of times that the simulation has to be repeated before the program’s termination.  
The admitted values are the integer between 0 and 2147483647. If 0 is specified, the simulation will be repeated infinitely.  
The default value is 1.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| TimesToRepeat | None | desc just a little description about this tag. |

Table ‑ TimesToRepeat content

#### Serial Port

The serial port configuration is completely covered by the following tags:

* <ComPort>
* <BaudRate>
* <DataBits>
* <StopBits>
* <Parity>
* <RetryCount>
* <SerialPortReopen>

A complete example is shown below:

<SerialPort desc="Container of the settings about the channel with the IBIS master.">

<ComPort desc="Serial port's name. Free text (case insensitive) Def. COM1">COM1</ComPort>

<BaudRate desc="Values admitted: {1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200} Def. 1200">1200</BaudRate>

<DataBits desc="Values admitted: {5, 6, 7, 8} Def. 7">7</DataBits>

<StopBits desc="Values admitted: {None, One, Two, OnePointFive} Def. Two">Two</StopBits>

<Parity desc="Values admitted: {Odd, Even, Mark, Space, None} (case insensitive) Def. Even">Even</Parity>

    <RetryCount desc="Tells how many attempts Protran has to do to open the serial port, if it is currently busy. Values admitted: an interger between 0 and 255. 0 means that Protran will not retry. Default value: 0">10</RetryCount>

 <SerialPortReopen desc="Attempt to reopen serial port due to errors {FrameOnly, All, None}. Default value: FrameOnly">FrameOnly</SerialPortReopen>

</SerialPort>

Figure 3‑4 Example of serial port configuration in IBIS.XML

In the example above was specified the COM1, the baud rate 1200, 7 bits, 2 bit for the stop, and an even parity.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| SerialPort | * ComPort * BaudRate * DataBits * StopBits * Parity * RetryCount * SerialPortReopen | desc just a little description about this tag. |

Table ‑ SerialPort content

##### <Sources><SerialPort><ComPort>

This tag contains the name of the serial port to which Protran has to receive/send IBIS telegram from/to the IBIS master.  
The default value is COM1.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| ComPort | None | desc just a little description about this tag. |

Table ‑ ComPort content

##### <Sources><SerialPort><BaudRate>

This tag contains an integer that represents the baud rate of the serial device.  
The admitted values are: 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200. The default value is 1200.

Do not use different baud rate, otherwise Protran might not work properly.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| BaudRate | None | desc just a little description about this tag. |

Table ‑ BaudRate content

##### <Sources><SerialPort><DataBits>

This tag contains an integer that represents the number of bits that compose a single information unit. The admitted values are: 5, 6, 7, 8.  
The default value is 7.

Do not use different data bits, otherwise Protran might not work properly.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| DataBits | None | desc just a little description about this tag. |

Table ‑ DataBits content

##### <Sources><SerialPort><StopBits>

This tag contains a string that represents the number of bits used to indicate the termination of single information. The admitted values are: None, One, Two, OnePointFive. The default value is Two.

Do not use different stop bits otherwise Protran might not work properly.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| StopBits | None | desc just a little description about this tag. |

Table ‑ StopBits content

##### <Sources><SerialPort><Parity>

This tag contains a string that represents the kind of parity used by the serial driver. The admitted values are: Odd, Even, Mark, Space, None.  
The default value is Even.

Do not use different parity otherwise Protran might not work properly.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| Parity | None | desc just a little description about this tag. |

Table ‑ Parity content

##### <Sources><SerialPort><RetryCount>

This tag contains an integer that represent the number of attemtps that Protran has to do to open the serial port, if it is currently busy.

The admitted values are the integers between 0 and 255. 0 means that Protran will not retry to re-open the serial port after the first failure.  
The default value is 0.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| RetryCount | None | desc just a little description about this tag. |

Table ‑ RetryCount content

##### <Sources><SerialPort><SerialPortReopen>

This tag contains information about under what serial port error conditions must Protran try to reopen the serial port. Errors on the serial port can be due to frame, parity or both. Protran can be configured to reopen serial port upon only frame errors (FrameOnly), all errors (All) or No errors (None). If configured as “None”, then Protran will not reopen serial port when errors are detected. It is expected to recover on its own.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| SerialPortReopen | None | desc just a little description about this tag.  Default value: FrameOnly |

Figure ‑ SerialPortReopen content

#### UDP Server

The IBIS UDP Server configuration is completely covered by the following tags:

* <LocalPort>
* <ReceiveFormat>
* <SendFormat>

A complete example is shown below:

<UDPServer desc="Use a UDP/IP server to receive IBIS telegrams">

  <LocalPort desc="Local UDP port, default: 47555">47555</LocalPort>

  <ReceiveFormat desc="Expected format of received IBIS telegrams.

    Values admitted: {Full, NoChecksum, NoFooter} (case insensitive) Def. Full">NoChecksum</ReceiveFormat>

  <SendFormat desc="Format of sent IBIS telegrams.

    Values admitted: {Full, NoChecksum, NoFooter} (case insensitive) Def. Full">NoFooter</SendFormat>

</UDPServer>

Figure 3‑6 Example of UDP server configuration in IBIS.XML

In the example above we specified the local UDP port to be 47555. The received telegrams contain the line-feed but no checksum; the sent telegrams shouldn’t contain any of the footer fields.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| UDPServer | * LocalPort * ReceiveFormat * SendFormat | desc just a little description about this tag. |

Table ‑ SerialPort content

##### <Sources><UDPServer><LocalPort>

This tag contains the local UDP port address on which Protran has to receive/send IBIS telegram from/to the IBIS master.  
The default value is 47555.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| LocalPort | None | desc just a little description about this tag. |

Table ‑ LocalPort content

##### <Sources><UDPServer><ReceiveFormat>

This tag contains the expected format for the reception of IBIS telegrams over UDP.

The following values are allowed:

* **Full**: The telegram contains header, payload, CR and checksum.
* **NoChecksum**: The telegram contains header, payload and CR but no checksum.
* **NoFooter**: The telegram contains header and payload but no CR and no checksum.

The default value is “Full”.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| ReceiveFormat | None | desc just a little description about this tag. |

##### <Sources><UDPServer><SendFormat>

This tag contains the expected format for sending IBIS telegrams over UDP.

The following values are allowed:

* **Full**: The telegram contains header, payload, CR and checksum.
* **NoChecksum**: The telegram contains header, payload and CR but no checksum.
* **NoFooter**: The telegram contains header and payload but no CR and no checksum.

The default value is “Full”.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| ReceiveFormat | None | desc just a little description about this tag. |

### IBIS Recording

Protran is capable to record in a specific file all the telegrams received from the IBIS master during a communication with this last. In this way, it’s possible to “re-load” the communication for debug purposes, post processing analysis and so on.

The configurable tags for this Protran’s feature are the following:

* <Recording><Active>
* <Recording><Format>
* <Recording><FileAbsPath>

A complete example is shown below:

<Recording>

<Active desc="Values admitted: {true, false} (case insensitive) Def. false">true</Active>

<Format desc="The file format for the recording.   
 Values admitted: {Protran,Gismo}">Protran</Format>  
 <FileAbsPath desc="Free text (case insensitive) Def. .\ibis.log">.\ibis.log</FileAbsPath>

</Recording>

Figure 3‑7 Example of recording configuration in IBIS.XML

In the example above the IBIS recording is active, the file produced doesn’t respect the “Gismo standard” and the file in which will be recorded all the IBIS telegram is called “ibis.log” and will be stored in the directory in which is also stored the application Protran.exe.

#### <Recording>

This tag contains all the information about the Protran’s recording capabilities with the IBIS master. It’s a container of other sub-tags so it doesn’t require any value.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| Recording | * Active * Format * FileAbsPath | None |

Table ‑ IbisRecordingConfig content

#### <Recording><Active>

This tag contains a string that tells is the recording feature has to be enabled or not.  
The admitted values are: true, false.  
The default value is false.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| Active | None | desc just a little description about this tag. |

Table ‑ Active content

#### <Recording><Format>

This tag contains a string that tells the format to use when recording to a file.  
The admitted values are: “Protran”, “Gismo”.  
The default value is “Protran”.

Attention: actually Protran doesn’t support the Gismo standard.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| Format | None | desc just a little description about this tag. |

Table ‑ ForGismo content

#### <Recording><FileAbsPath>

This tag contains a string that represents the absolute path of the file that will contain the recorded IBIS telegrams.

Attention: if the file specified already exists, the file will be overwritten (Protran doesn’t write on it in append mode).

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| FileAbsPath | None | desc just a little description about this tag. |

Table ‑ FileAbsPath content

### Time Synchronization

Inside the system in which Protran is running, there might be the need to synchronize the time through all the entities involved in the system itself. Protran is capable to recognize the date/time IBIS telegrams and forward the reference date/time to all the other entities. To do that it needs some configuration parameters that are the following:

* <TimeSync>
* <TimeSync><InitialDelay>
* <TimeSync><WaitTelegrams>
* <TimeSync><Tolerance>

A complete example is shown below:

<Ibis>

<TimeSync Enabled="true" desc="Container of the settings about the date-time synchronization process.">

<InitialDelay desc="Time to wait before the time sync is started.">PT10S</InitialDelay>

<WaitTelegrams desc="Number of telegrams to check before the time is considered valid.">3</WaitTelegrams>

<Tolerance desc="Tolerance above which the difference has to be for the time to be synchronized">PT0S</Tolerance>

</TimeSync>

</Ibis>

Figure 3‑8 Example of time synchronization configuration in IBIS.XML

In the example above the time synchronization feature is enabled, Protran starts to synchronize the date and time after 10 seconds of its startup, also it waits for 3 valid IBIS telegrams before really synchronizing the date/time through Hardware Manager.

#### <TimeSync>

This tag contains the set of information required by Protran to perform the date and time synchronization process.  
The admitted values are: true, false. To enable this feature must be set to “true” the attribute “Enabled”, otherwise “false” to disable it.  
The default value is true.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| TimeSync | * InitialDelay * WaitTelegrams * Tolerance | Enabled A flag to tell if time synchronization is enabled.  desc just a little description about this tag. |

Table ‑ IbisTimeSynch content

#### <TimeSync><InitialDelay>

This tag contains an XML duration that represents the amount of time that Protran will wait before starting the date and time synchronization process.  
The admitted values are positive durations.  
The default value is “PT10S” (10 seconds).

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| InitialDelay | None | desc just a little description about this tag. |

Table ‑ InitialDelay content

#### <TimeSync><WaitTelegrams>

This tag contains an integer that represents the amount of valid IBIS telegrams received from the IBIS master regarding the date and time, before starting the synchronization process.  
The admitted values are the integers between 0 and 65535.  
The default value is 3 telegrams.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| WaitTelegrams | None | desc just a little description about this tag. |

Table ‑ WaitTelegrams content

#### <TimeSync><Tolerance>

This tag contains an XML duration that represents the delta within which Protran doesn’t treat a time telegram. For example, if the current time is set to 12:00:00 and the tolerance is 10 seconds, Protran discards all the IBIS time telegrams having time between 11:59:50 and 12:00:10 because those times are within the admitted tolerance. The admitted values are positive durations.  
The default value is “PT0S” (0 seconds).

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| Tolerance | None | desc just a little description about this tag. |

Table ‑ Tolerance content

### IBIS Telegrams

Protran is capable to recognize several telegrams coming from the IBIS master.

For each telegram, Protran can

* Send back to the IBIS master an answer telegram
* Manipulate the payload
* Use the manipulated payload for specific purposes

Before seeing in details the three points above, in the next chapter will be explained what a telegram is and all their attributes and values. Also, the last two previous points are extremely important and for this reason they will be explained separately (see chapters 4 and 7).

## vdv301.xml

vdv301.xml is an additional file for Protran used whenever the **VDV301** protocol is specified.

Attention: it’s strictly required that the file “vdv301.xml” is stored in the same directory as “protran.xml”. Otherwise Protran will exit with an error message.

The file “vdv301.xml” has the following groups of configuration:

* The **services** used by Protran to obtain information.
* A mapping of VDV 301 **language** definitions to Ximple language definitions.
* Information about the **translation** from the VDV 301 data to Ximple.

More information about VDV 301 can also be found in chapter 5.

### Explanation of the general structure

This chapter explains the general structure of the vdv301.xml configuration file without covering all elements of the structure. Please refer to the next chapters for a detailed overview of the file structure.

#### Services

The “Services” section of the configuration file contains one element for each service to subscribe to.

According to VDV 301, services can be discovered automatically using DNS-SD; therefore it is not necessary to specify where the service(s) can be found. If service discovery is not available or not working as expected, it is possible to specify the “Host”, “Port” and “Path” parameters manually; if they are defined, automatic service discovery is disabled for the given service.

Inside each service element, there are elements available for all operations defined by VDV 301-2 (e.g. “CustomerInformationService” has an operation called “GetAllData”, therefore there is a “GetAllData” element inside the “CustomerInformationService”). Protran will always use the Subscribe<operation> and Unsubscribe<operation> operations to receive data from a remote service. Currently there is no polling implemented.

The structure inside each operation element is a replica of the actual VDV 301 structure. Every “leaf” element (not containing any children) of the standard is represented by an element name that allows to transform the data (if needed) and then put it into a Ximple cell. It is possible to have multiple elements of the same name if there is the need to store the same data into different Ximple cells (e.g. when different transformations are needed for the same input data).

#### Languages

The language mapping allows for the “InternationalTextType” VDV 301 structures to be mapped to Ximple languages. Usually there is only one language available in the VDV 301 data structure, so it is enough to map it to the “default” language of Ximple.

If a language is received from the VDV 301 service but there is no language mapping for the given language, the data is ignored.

Given the following mapping (in vdv301.xml):

|  |
| --- |
| <Languages>    <Language VDV301="de" Ximple="default" />  </Languages> |

the output for (VDV 301 data received from a remote service):

|  |
| --- |
| <StopName>  <Value>K.-Adenauer-Platz</Value>    <Language>de</Language>  </StopName> |

would be (Ximple output sent to Infomedia):

|  |
| --- |
| <Ximple Version="2.0.0">    <Cells>      <Cell Language="0" Table="…" Column="…" Row="…">K.-Adenauer-Platz</Cell>    </Cells>  </Ximple> |

In the above example the mapping from “default” to “0” is done using dictionary.xml:

|  |
| --- |
| <Languages>    <Language Index="0" Name="default" Description="Default language" />  <Languages> |

#### Transformations

If a value received from a VDV 301 service has to be modified, transformations can be used. Contrary to the IBIS protocol (see above and below) there is no need to have a “default” transformation for a value; if no transformation is given, the value is simply passed to Infomedia using it in Ximple exactly as received.

The same transformations are available in VDV 301 as for other protocols. Please refer to chapter “5 Telegram’s Transformations” below for more details.

### @XmlDoc(xsd=..\..\..\..\Common\Configuration\Source\Protran\VDV301\vdv301.xsd;xml=..\..\Source\Vdv301\vdv301.xml)

## io.xml

The I/O protocol is responsible for two things:

* reading and setting serial port I/Os on the old Gorba Topboxes (PM600, PM800 and Atom)
* converting any I/O from the system to Ximple cells

To enable this protocol, you need to add it to “**protran.xml**” in the section <Protran><Protocols>:

<Protocol Name="IOProtocol" Enabled="true" description="This is the I/O protocol"/>

The following figure shows a sample “**io.xml**” file:

<?xml version="1.0" encoding="utf-8" ?>

<IO>

  <SerialPorts>

    <SerialPort Name="COM1">

      <RTS>Speaker</RTS>

      <CTS>SpecialInput</CTS>

    </SerialPort>

    <SerialPort Name="COM2">

      <CTS>StopRequest</CTS>

    </SerialPort>

  </SerialPorts>

  <Inputs>

    <Input Name="StopRequest" TransfRef="Default" Enabled="true">

      <UsedFor Table="SystemStatus" Column="StopRequestedState" Row="0"/>

    </Input>

    <Input Name="SpecialInput" TransfRef="Default" Enabled="true">

      <UsedFor Table="SystemStatus" Column="SpecialInput" Row="0"/>

    </Input>

  </Inputs>

  <Transformations>

    <Chain id="Default" />

  </Transformations>

</IO>

Figure 3‑9 A complete io.xml file

### Serial port configuration

This is configured in the tags inside <IO><SerialPorts>.

If you want to use serial port pins as I/Os in the system (also other applications like Renderer will be able to access them), you need to configure the port and the name you want to give to the I/O.

Serial ports can be shared between the IBIS protocol in Protran and the I/O protocol. This is required if the same port provides IBIS data and I/Os (e.g. enabling the speaker through the RTS pin on the old Gorba Topboxes).

The above example file contains the usual configuration for the old Gorba Topboxes.

#### <IO><SerialPorts><SerialPort>

This tag can contain all input and output pins available on a serial port with the name mapped for it. For simplicity all ports are also provided as an I/O with the name “<serialport>.<pin>” (e.g. “COM1.RTS”, “COM2.CTS”, …). The following pins are available:

* <RTS> (output)
* <CTS> (input)
* <DTR> (output)
* <DSR> (input)

### Input mapping

Inputs from anywhere in the Medi network can be used to generate Ximple cells. Usually those inputs are from the local machine (but a different process), but it is possible to gather inputs from a different machine (see below).

Each input is configured independently.

#### <IO><Inputs><Input>

This tag has the following attributes:

* Enabled (optional), enables or disables the handling of the input (default: "true")
* Unit (optional), defines the name of the unit (machine/PC) where the I/O is located (default: empty, this means the local unit is used)
* Application (optional), defines the name of the application where the I/O is located (default: empty, this means all application on the configured Unit are queried to find the I/O)
* Name (mandatory), the name of the input, this must match the name of the input in the “providing” application. If the I/O is from a serial port, this must be the same as name given to the pin (<IO><SerialPorts><SerialPort><…>), see 3.5.1.1.
* TransfRef (mandatory), the transformation (see below) used to convert the I/O value to a Ximple cell value

This tag has a single inner element:

* <UsedFor> The generic coordinate to fill with the transformed value. See 4.31.1 for more information.

### Transformations

The transformations are configured exactly the same way as in ibis.xml. See chapter 5 for more information.

## Arriva.XML

The file “arriva.xml” has to be stored in the same directory on which is stored the file Protran.exe. The Protran’s “arriva.xml” file contains different XML tags. They can be grouped in:

* ComputerName tags
* FTP tags
* Behaviour tags

An example of configuration is the following:

<?xml version="1.0" encoding="utf-8"?>

<Arriva>

<FTP Descr="Information for the FTP protocl.">

<PollingEnabled>true</PollingEnabled>

<SourceDirectory

Descr="local directory with file for connections.">C:\Departures\Gorba\Xml</SourceDirectory>

<Filename Descr="File to use for connections.">departures.xml</Filename>

</FTP>

<Behaviour>

 <ConnectionsEnabled>true</ConnectionsEnabled>

 <ConnectionSource desc="connection source for transfer set.

values admitted: {ArrivaProtocol, Ftp}">Ftp</ConnectionSource>

 <MaxDepartures

desc="maximum connections in the departures file.

Values admitted {1; 100}. Default value: 100">20

</MaxDepartures>

 </Behaviour>

</Arriva>

### FTP Tag

This tag contains all the information about the Arriva protocol’s FTP connection. It’s a container of other sub-tags so it doesn’t require any value.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| FTP | * PollingEnabled * SourceDirectory * Filename | Descr just a little description about this tag. |

Table ‑ FTP content

#### PollingEnabled

This tag contains a boolean that represents whether polling is enabled using FTP for Arriva protocol in Protran.  
The admitted values are the true or false.  
The default value is true.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| PollingEnabled | None | None |

Table ‑ PollingEnabled content

#### SourceDirectory

Specify the absolute file path of the file that will be used by Protran to find the connections file sent by the Ftp client.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| Descr | yes | Just a little description about this tag. | A string of full path for file. Default: empty |

Table ‑ SourceDirectory content

#### Filename

Specify the name of the file that will be used by Protran to verify connections file sent by the Ftp client.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| Descr | yes | Just a little description about this tag. | A string of name of the file. Default: empty |

Table ‑ Filename content

### Behaviour Tag

This tag contains all the information about the Arriva protocol’s behavior w.r.t the connections. It’s a container of other sub-tags so it doesn’t require any value.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| Behaviour | * ConnectionsEnabled * ConnectionSource * MaxDepartures | None |

Table ‑ Behaviour content

#### ConnectionsEnabled

This tag contains a boolean that represents the whether processing of connections sent by Ftp client is enabled for Arriva protocol in Protran.  
The admitted values are the true or false.  
The default value is true.

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| ConnectionsEnabled | None | None |

Table ‑ ConnectionsEnabled content

#### ConnectionSource

Specify the source of the connections information for Arriva in Protran. The source can be either Arriva Protocol or the Ftp client.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| Descr | Yes | Just a little description about this tag. | A string. Default: Ftp |

Table ‑ ConnectionSource content

#### MaxDepartures

Specify the maximum number of departures allowed in a connections file sent by the Ftp client to Protran.

The admitted values are the 1 to 100. The default value is 100.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| Descr | No | Just a little description about this tag. | An integer. Default: 100 |

# IBIS Telegrams

An IBIS telegram actually is a set of bytes sent from the IBIS master. It contains a header, a payload and a trailer as shown below:

Header

Payload

Trailer

Figure 4‑1 General Structure of an IBIS Telegram

Depending on the header, Protran recognizes or rejects the IBIS telegrams coming from the IBIS master. Some telegrams contain, within the payload, the additional information called “IBIS address” that specifies that the telegram is addressed only to a targeted IBIS slave (the others on the same bus have to reject it). Also, if the CRC code contained into the trailer is wrong, Protran rejects the telegram (if this behavior is enabled, see paragraph 3.3.1.4) otherwise it starts to parse and analyze the telegram itself. The parse and the analysis phases are done depending on how the telegram is configured in the file ibis.xml. Here below will be explained all the possible attributes and sub-tags usable for each telegram.

## Supported Telegrams

Currently the following telegrams are supported by Protran:

|  |  |  |
| --- | --- | --- |
| **Telegram** | **Signature** | **Short Description** |
| DS001 | l | Line number (lowercase Lima) |
| DS001a | lE | Special line number (lowercase Lima) |
| DS002 | k | Run number |
| DS003 | z | Destination number |
| DS003a | zA | Destination |
| DS003c | zI | Next stop (n \* 4 characters, alpha-numeric) (uppercase India) |
| DS005 | u | Time of day |
| DS006 | d | Calendar date |
| DS006a | dU | Date and time in “ddMMyyyyHHmmss” format |
| DS008 | n | Wagon address |
| DS009 | v | Next stop (16 to 24 characters, alpha-numeric – includes DS009a and b) |
| DS010 | x | Current stop index, used with GO005 |
| DS010b | xI | Current stop index, used with DS021a (uppercase India) |
| DS010j | x | Current stop index combined with status, used with DS021c |
| DS020 | a | Status inquiry |
| DS021 | aA | Destination (addressed) |
| DS021a | aL | Stop list |
| DS021c | aX | Stop list |
| DS036 | hP | Audio announcement |
| DS080 | bT | Open door |
| DS081 | bM | Door has closed |
| GO001 | xE | Events from CU (currently used for “100m before stop”) |
| GO002 | aU | Connection information |
| GO003 | aB | Stop list (Gorba 16-bit version) |
| GO004 | aM | Text messages |
| GO005 | aA | Stop list |
| GO006 | z | Alphanumeric line number |
| GO007 | aA | Stop list with line number |
| HPW074 | sN | Special text (popup) index |

Note: We strongly suggest **only enabling the telegrams that are actually used** for a given customer. Some telegrams even have the same header and therefore enabling both could cause unexpected behavior.

### DS001

Telegram for line number.

**Signature:** l

**Full Telegram:** lZZZ<CR><FP>

Where:

ZZZ three numeric digits for the line number

**Examples:** l010

l005

**Variants:** Protran supports line numbers with 3 to 6 digits

### DS001a

Telegram for special line number

**Signature:** lE

**Full Telegram:** lEZZ<CR><FP>

Where,

ZZ two numeric digits that specify the code for the special line character

These are used as code for symbols with code conversion.

**Examples:** lE12

lE05

**Variants:** Protran supports special line numbers with 2 to 4 digits

### DS002

Telegram for run number

**Signature:** k

**Full Telegram:** kZZ<CR><FP>

Where,

ZZ two numeric digits with leading zeros for the run number

**Examples:** k10

k25

**Variants:** Protran supports run numbers with 2 or 3 digits

### DS003

Telegram for destination number

**Signature:** z

**Full Telegram:** zZZZ<CR><FP>

Where,

ZZZ three numeric characters with leading zeros that specify the code for the for the destination text

**Examples:** z123

z055

**Variants:** Protran supports destination numbers with 3 or 4 digits

### DS003a

Telegram for destination text

**Signature:** zA

**Full Telegram:** zAHnC<CR><FP>

Where,

H Length of the data in blocks of 16 characters

Example: zA5 – means that there are 80 characters of data following zA

nC The destination text padded with spaces

The interpretation of data can be different – the usual variant for zA4:

* 16 characters for the front display, 1st line ¦ 16 characters for the front display, 2nd line
* 16 characters for the side display, 1st line ¦ characters for the side display, 2nd line

**Example:** zA2Neuchatel˽˽˽˽˽˽˽Solothurn˽˽˽˽˽˽˽

**Variants:** Protran supports any number of characters (it doesn’t have to be a multiple of 16)

### DS003c

Telegram for Next stop

**Signature:** zI

**Full Telegram:** zIHnC<CR><FP>

Where,

H Length of the data in blocks of 4 characters

Example: zI4 – means that there are 16 characters of data following zI

nC The next stop name padded with spaces

**Example:** zI3Neuchatel˽˽˽

**Variants:** Protran supports any number of characters (it doesn’t have to be a multiple of 4)

### DS005

Telegram for time

**Signature:** u

**Full Telegram:** uZZZZ<CR><FP>

Where,

Z1Z2 two numeric digits for the hour

Z3Z4 two numeric digits for minutes

**Examples:** u1000

u1230

### DS006

Telegram for date

**Signature:** d

**Full Telegram:** dZZZZZ<CR><FP>

Where,

Z1Z2 two numeric digits for the day

Z3Z4 two numeric digits for month

Z5 one numeric digit for the year

**Examples:** d26014

d01124

**Variants:** Protran supports 5 or 6 digits; if 6 digits are defined, Z5Z6 are used as the two numeric digits for the year

### DS006a

Telegram for date and time

**Signature:** dU

**Full Telegram:** dUZZZZZZZZZZZZZZ<CR><FP>

Where,

Z1Z2 two numeric digits for the day

Z3Z4 two numeric digits for month

Z5Z6Z7Z8 four numeric digits for the year

Z9Z10 two numeric digits for the hour

Z11Z12 two numeric digits for minutes

Z13Z14 two numeric digits for seconds

**Example:** dU23072014152719

### DS008

Telegram for wagon address

**Signature:** n

**Full Telegram:** nHHH<CR><FP>

Where,

HHH three hex digits for the wagon address

**Example:** n123

### DS009

Telegram for stop name

**Signature:** v

**Full Telegram:** v16C<CR><FP>

Where,

16C the stop name padded with spaces

**Example:** vNeuchatel˽˽˽˽˽˽˽

**Variants:** Protran supports 16 to 24 characters; the variant with 20 characters (v20C) is also referred to as **DS009a** and the variant with 24 characters (v24C) as **DS009b**.

### DS010

Telegram for current stop index. Used in conjunction with GO005

**Signature:** x

**Full Telegram:** xZZZZ<CR><FP>

Where,

ZZZZ 4 digit number with leading zeros for the stop index (0001 ... 0999)

**Examples:** x0001

x0005

**Variants:** Protran supports DS010 stop indexes with 4 or 5 digits

### DS010b

Telegram for current stop index. Used in conjunction with DS021a. Also used when handling DS080, GO001 and GO002.

**Signature:** xI

**Full Telegram:** xIZZ<CR><FP>

Where,

ZZ 2 digit number with leading zeros for the stop index (01 ... 99)

**Examples:** xI01

xI99

**Variants:** Protran supports DS010b stop indexes with 2 or 3 digits

### DS010j

Telegram for current stop index combined with status. Used in conjunction with DS021c

**Signature:** x

**Full Telegram:** xSZZZ<CR><FP>

Where,

S Status (number 0 … 9)

* 2 – All stops must be displayed
* 3 – Current stop must be suppressed

ZZZ Stop index (000 ... 999)

**Examples:** x2001

X3009

### DS020

Status enquiry telegram. It contains only the address of the display which is going to be asked for the status

**Signature:** a

**Full Telegram:** aH<CR><FP>

Where,

H Ibis address in hex

**Examples:** a8

a:

### DS021

Telegram for Destination for specific address

**Signature:** aA

**Full Telegram:** aAHHnC<CR><FP>

Where,

H1 Related IBIS address

H2 Length of the data in blocks of 16 characters

nC Destination name

**Examples:** aA81Neuchatel˽˽˽˽˽˽˽

aA:1Solothurn˽˽˽˽˽˽˽

### DS021a

Telegram for stop list

**Signature:** aL

**Full Telegram:** aLH1H2H3nC

Where,

aL telegram identifier  
H1 address of the route path display to be fed  
H2H3 factors for the length of data (n = 16\*H2 + H3)  
nC user data

The data (nC) are interpreted in the following way (values in angle brackets are hex values):  
<03>II<04>mC<05>kC<06>4Z  
  
II: stop index - 01 … 99: stop index within the actual pattern; 99: end of data   
mC: stop name (0 … 40 characters, usually 20 characters)  
kC: additional text information for changeover   
 (0 … 40 characters, usually 20 characters)  
4Z time information for changeover (4 decimal characters; e.g. 0940)

**Example:** aL800<03>01<04>neuchatel<05><0D><8D>

**Variants:** Protran can be configured to use any other delimiters instead of <03>, <04>, <05> and <06>.  
The “additional text information” and “time information for changeover” blocks including their relative preceding delimiter are optional.

### DS021c

Telegram for stop list

**Signature:** aX

**Full Telegram:** aXHZ<03>ZZZ<04>nC<05>nC<CR><FP>

Where,

aX telegram header

H IBIS address, to which TFT the telegram belongs to

Z status flag

* 0: initialisation 🡪 Protran deletes all received DS021b data received so far
* 1: valid data, to be stored internally
* 2: indication for the last record of the DS021b telegram and switching the answer for DS120 to 1
* Telegrams with other status flags are ignored by Protran

<03> delimiter

ZZZ stop indexes: 001 ... 099 (these are the only valid stop indexes!)

* 100: start text (where does the bus start) - currently there is no dictionary item to support this
* 101: destination text (alternative to the last stop in the route) -> telegram configuration must support the possibility to link to the related dictionary item (DestinationName)
* 102: ASCII line number (alternative to the l telegram - this alternative support alphanumeric line numbers) -> telegram configuration must support the possibility to link to the related dictionary item (Line)
* 103 ... 199: reserved for future usage

<04> delimiter

nC stop name or special texts (depending on the xxx)

<05> delimiter (optional, only available if additional information are following)

nC additional information (e.g. via infos or changeover texts)

**Example:** aX81<03>001<04>Neuchatel<05><0D><98>

**Variants:** Protran can be configured to use any other delimiters instead of <03>, <04> and <05>.

### DS036

Telegram for audio announcement.

**Signature:** hP

**Full Telegram:** hPnZ<CR><FP>

Where,

nZ Number of the next stop. The minimum value for “n” is 4.

**Examples:** hP0025

hP00140345

### DS080

Telegram to open door

**Signature:** bT

**Full Telegram:** bT<CR><FP>

**Example:** bT

### DS081

Telegram to close door

**Signature:** bM

**Full Telegram:** bM<CR><FP>

**Example:** bM

### GO001

Telegram for events from CU (currently used for “100m before stop”)

**Signature:** xE

**Full Telegram:** xEHZ<CR><FP>

Where,

H Ibis address

Z Event number. Currently only event number 1 is supported

**Example:** xE81<OD><C6>

### GO002

Telegram for connection information

**Signature:** aU

**Full Telegram:** aUXXXSSRPLLLLLLUUUUMMVVVVnC<CR><FP>

Where,

XXX Length calculated from HH to nC including nC

SS stop index

R Row number

P pictogram name

LLLLL line number

UUUU departure time

MM track number

VVVV time deviation

nC name of the connection. Maximum 30 characters

**Example:** aU2080111555551010010000Neuchatel<OD><80>

**Variants:** The length of each field can be freely configured in Protran, the lengths shown here are the respective default values.  
Protran supports short telegrams (aUXXX<CR><FP> and aUXXXSS<CR><FP>) for clearing data.

### GO003

Telegram for stop list

**Signature:** aB

**Full Telegram:** aBHHnC<CR><FP>

Where,

H1 IBIS address

H2 length of the data

nC user data; stops are separated by <0A>

**Example:** aB800stop1<0A>stop2<0A>stop3<0A><OD><E1>

**Variants:** Protran supports any number of characters (the length field is ignored)

### GO004

Telegram for text messages

**Signature:** aM

**Full Telegram:** aMHZZZZZZZZZZZZnC<CR><FP>

Where,

H IBIS address

Z1Z2 message index: row 0, row 1, row 2, …

Z3Z4 message type

Z5Z6Z7Z8 start validity seconds since midnight

Z9Z10Z11Z12 end validity seconds since midnight

nC message text:

* the text before the 1st <10> is treated as message title
* the text after the 1st <10> is treated as message text
* further <10> are translated internally to BBCode [br] (line break)

**Example:** aM8010111010180new message<OD><D8>

### GO005

Telegram for stop list

**Signature:** aA

**Full Telegram:** aAHHH<03>llllxxxx<04>nC<CR><FP>

Where,

H1 related IBIS address

H2H3 data length => H2 \* 16 + H3 \* 4 = number of characters after H3 up to (excluding) <CR>

<03> fixed delimiter (hex 03h)

llll alphanumeric line number (4 character)

xxxx stop index (number with leading zeros - 4 digits)

<04> fixed delimiter (hex 04h)

nC stop name

**Example:** aA810<03>A2340001<04>stop 1<0A> <OD><CA>

**Variants:** Protran can be configured to use another delimiter instead of <04> (<03> can’t be changed).  
Protran supports any number of characters (it doesn’t have to be a multiple of 4; the data length field is ignored)

### GO006

Telegram for alphanumeric line number

**Signature:** z

**Full Telegram:** z0CCCC<CR><FP>

Where,

0 fixed digit zero (“0”)

CCCC 4 character alphanumeric line number

**Example:** Z0A123

**Variants:** Protran supports alphanumeric line numbers with 3 to 5 characters

### GO007

Telegram for stop list with line number.

**Signature:** aA

**Full Telegram:** aAHHHnC<CR><FP>

Where,

H1 IBIS address

H2H3 length of the data

nC user data; line number separated by <03>; stops are separated by <04>;   
connection information separated by <05>

**Example:** aA800<03>lineNumber<04>stop1<05>connection<04>stop2<05>connection<04>stop3<05>connection<04><OD><E1>

### HPW074

Telegram for special text (pop-up)

**Signature:** sN

**Full Telegram:** sNHZZ<CR><FP>

Where,

H IBIS address

ZZ Text index (to be looked up in a separate file)

**Examples:** sN801

sN:05

## DS001

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑1 DS001 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑2 DS001 sub-tags

The “UsedFor” attribute should always be present (unlike in Protran 1.x).

## DS001a

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑3 DS001a attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑4 DS001a sub-tags

There is no special handling of DS001a. If you need “code conversion”, please configure it in Infomedia (unlike in Protran 1.x).

## DS002

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be a number. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑5 DS002 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑6 DS002 sub-tags

## DS003

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be a number. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑7 DS003 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑8 DS003 sub-tags

## DS003a

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑9 DS003a attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑10 DS003a sub-tags

## DS003c

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑11 DS003c attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑12 DS003c sub-tags

## DS005

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑13 DS005 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑14 DS005 sub-tags

This telegram’s “UsedFor” should not be present in this section if you intend on using Time Sync (see chapter 3.3.4); but be aware that if you don’t configure the DS005 telegram at all (or set its “Enabled” attribute to false), Time Sync will not work.

## DS006

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑15 DS006 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |
| InitialYear | no | Defines the initial year for year calculation. Default value: 2014 |
| OutputFormat | no | Defines the format for outputting the date. Default value: dd.MM.yyyy |

Table 4‑16 DS006 sub-tags

Protran expects DS006 to be either in “ddMMy” (VDV standard) or “ddMMyy” format.

When Protran gets a DS006, it will parse the content (if it is 5 or 6 characters long) and then format the result using the “OutputFormat”. The following characters are allowed in the output format:

* “d”: Day of month, one or two digits
* “dd”: Day of month, two digits, padding with “0”
* “M”: Month of the year, one or two digits
* “MM”: Month of the year, two digits, padding with “0”
* “yy”: Year, two digits
* “yyyy”: Year, four digits
* More options can be found here: <http://www.csharp-examples.net/string-format-datetime/>

The “InitialYear” is used to calculate the missing digits of the year. The resulting value will always be greater than or equal to the defined initial year. Examples:

|  |  |  |
| --- | --- | --- |
| **D006 year** | **InitialYear** | **Output year** |
| 1 | 2012 | 2021 |
| 2 | 2012 | 2012 |
| 4 | 2012 | 2014 |
| 11 | 2012 | 2111 |
| 13 | 2012 | 2013 |
| 1 | 2015 | 2021 |
| 2 | 2015 | 2022 |
| 4 | 2015 | 2024 |
| 11 | 2015 | 2111 |
| 13 | 2015 | 2113 |

Table 4‑17 DS006 InitialYear examples

## DS006a

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑18 DS006a attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the **formatted** payload (see OutputFormat below) of this telegram is used (see chapter 4.31.1). |
| OutputFormat | no | Defines the format for outputting the date. Default value: dd.MM.yyyy |

Table 4‑19 DS006a sub-tags

Protran expects DS006a to be in “ddMMyyyyHHmmss” format.

When Protran gets a DS006a, it will parse the content (if it is 14 characters long) and then format the result using the “OutputFormat”. The following character sequences are allowed in the output format:

* Date (normal usage):
  + “d”: Day of month, one or two digits
  + “dd”: Day of month, two digits, padding with “0”
  + “M”: Month of the year, one or two digits
  + “MM”: Month of the year, two digits, padding with “0”
  + “yy”: Year, two digits
  + “yyyy”: Year, four digits
* Time (use with caution):
  + “h”: 12-hour time format, one or two digits
  + “hh”: 12-hour time format, two digits, padding with “0”
  + “H”: 24-hour time format, one or two digits
  + “HH”: 24-hour time format, two digits, padding with “0”
  + “m”: minutes, one or two digits
  + “mm”: minutes, two digits, padding with “0”
  + “s”: seconds, one or two digits
  + “ss”: seconds, two digits, padding with “0”
* More options can be found here: <http://www.csharp-examples.net/string-format-datetime/>

## DS008

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑20 DS008 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑21 DS008 sub-tags

## DS009

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑22 DS009 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑23 DS009 sub-tags

Protran supports DS009 with a payload of 16 to 24 characters. This means, you have to use DS009 in the configuration if you need to handle DS009a or DS009b.

This telegram’s “UsedFor” is usually left empty when this telegram is used to handle GO007.

## DS010

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | no | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑24 DS010 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑25 DS010 sub-tags

This telegram’s “UsedFor” is usually left empty since this telegram is only used to handle GO005.

## DS010b

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be a number. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑26 DS010b attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑27 DS010b sub-tags

This telegram’s “UsedFor” is usually left empty since this telegram is only used to handle DS021a.

## DS010j

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be a number. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑28 DS010j attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑29 DS010j sub-tags

This telegram’s “UsedFor” is usually left empty since this telegram is only used to handle DS021c.

## DS020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑30 DS020 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| Answer | Yes | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |

Table 4‑31 DS020 sub-tags

This telegram’s “UsedFor” is usually left empty since this telegram is only used to send IBIS answers with the status to the master. Therefore the “Answer” sub-tag should always be configured correctly, see chapter 4.31.2.

## DS021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑32 DS021 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |
| Answer | no | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |

Table 4‑33 DS021 sub-tags

Be aware that this telegram has the same header as GO005 and

GO007, so always make sure to only enable DS021 or GO005 or

GO007 but never more than one of these at the same time.

## DS021a

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be an array of strings. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑34 DS021a attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| FlushNumberOfStations | no | Number of stops to collect before sending an update to Infomedia. Default: 5 |
| FlushTimeout | no | Time to wait before finishing the collection of stops. Default: PT30S |
| HideLastStop | no | Hide the last stop of the stop list. Default: false |
| HideDestinationBelow | no | Hide the destination (see UsedForDestination and so on) if less than this number of stations is shown. Default: 0 (disabled) |
| ShowPastStops | no | Put past stops into generic cells with negative row numbers. Default: false |
| FirstStopIndexValue | no | Value of the first stop coming from the IBIS master. Default value: 0 |
| EndingStopValue | no | Value of the end marker for the stop list. This is usually 99 or 999. Default value: 99 |
| DeleteRouteIndexValue | no | Value of the index to delete a route. Values admitted {-1, 0}. Default value: -1 |
| AbsoluteTimeFormat | no | Format of the UsedForAbsoluteTime and UsedForDestinationAbsoluteTime.  Meaningful formats: 'HH:mm' (24-hour format), 'H:mm' (1-digit hour), 'hh:mm' (12-hour format)  Default value: 'HH:mm' |
| UsedFor | no | Usage of the first part in DS021a (see also chapter 4.31.1, placeholder allowed). |
| UsedForTransfers | no | Usage of the second part in DS021a (see also chapter 4.31.1, placeholder allowed). |
| UsedForTransferSymbols | no | Usage of the third part in DS021a (see also chapter 4.31.1, placeholder allowed). |
| UsedForRelativeTime | no | Usage of the fourth part in DS021a (see also chapter 4.31.1, placeholder allowed). |
| UsedForAbsoluteTime | no | Usage of the fourth part in DS021a (see also chapter 4.31.1, placeholder allowed). |
| UsedForDestination | no | Usage of the first part in the last entry of DS021a (see also chapter 4.31.1, no placeholder) |
| UsedForDestinationTransfers | no | Usage of the second part in the last entry of DS021a (see also chapter 4.31.1, no placeholder) |
| UsedForDestinationTransferSymbols | no | Usage of the third part in the last entry of DS021a (see also chapter 4.31.1, no placeholder) |
| UsedForDestinationRelativeTime | no | Usage of the fourth part in DS021a (see also chapter 4.31.1, placeholder allowed). |
| UsedForDestinationAbsoluteTime | no | Usage of the fourth part in DS021a (see also chapter 4.31.1, placeholder allowed). |
| Answer | no | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |
| Connection | no | DS021a ‘A’ variation (see chapter 0). |

Table 4‑35 DS021a sub-tags

The handling of stop list telegrams can be configured very specifically for each customer. Therefore you have a lot of possible parameters. In the following paragraphs you find an explanation of the algorithm and how it uses the given parameters.

All the usage sections of DS021 sub-telegrams such as DS021A, DS021C and GO005 have an additional attribute “FromBlock” to signify if the particular usage is to be used and at what position in the telegram.

### Telegram Collection

Protran collects the information retrieved from DS021a and caches it. To prevent the stop list from “building up” one line at the time, Protran will only send the stop list information to Infomedia in the following cases:

1. When a DS021a with a stop index equal to FirstStopIndexValue is received, the stop list is cleared and empty cells are sent to Infomedia (“full clear”)
2. A multiple of FlushNumberOfStations has been collected (“intermediate flush”)
3. A DS021a with a stop index equal to EndingStopValue is received (“last flush”)
4. The last telegram was not yet received, but Protran didn’t receive a DS021a for FlushTimeout seconds (“timeout flush”)
5. A DS010b with a stop index different from the previous one is received (“index update flush”)

After a “last” or “timeout” flush Protran does not collect any DS021a anymore until it receives again a first telegram (see case 1 above).

It is suggested to set FlushNumberOfStations to the number of stops shown in the stop list (“Perlschnur”) in Infomedia.

### Stop Information

DS021a is split into four/five columns usually using <04>, <05>, # and $ as delimiters:

1. Stop index (2 or 3 digits) followed by <04>
2. Stop name followed by <05> (UsedFor)
3. Transfer information followed by “#” (UsedForTransfers)
4. Transfer symbol list delimited by “;” (UsedForTransferSymbols)
5. If present, travel times are followed by “$” (UsedForRelativeTime, UsedForAbsoluteTime)

For each DS021a telegram this information is stored and sent to Infomedia according to the configured usage.

If the customer wishes not to display the last stop since this is the same information shown in the destination (see below), it is possible to always filter out the last stop by setting HideLastStop to true.

If the customer wishes to display past stops using negative row numbers it is possible to enable the feature by setting ShowPastStops to true.

### Destination Information

The last stop of the stop list can be used as destination, if the IBIS master does not provide this information otherwise (e.g. DS003a). The information is retrieved exactly the same way as the stop information (see above) and is configured with UsedForDestination, UsedForDestinationTransfers, UsedForDestinationTransferSymbols, UsedForDestinationRelativeTime and UsedForDestinationAbsoluteTime respectively.

If the customer wishes not to display the destination when it is already visible as the last entry in the stop list, the destination can be hidden by setting HideDestinationBelow to the number of stops visible in the stop list **plus one**. For example if your “Perlschnur” contains 5 stops plus the destination, you set the value to 6 and then Protran will hide all destination information if there are less than 6 stops in the stop list. This behavior is disabled by setting the value to 0 or removing the sub-tag.

### Travel times calculation

The travel time to be displayed for each stop and destination is calculated based on the following algorithm. Each telegram contains a travel time value. When the stops and destination have to be displayed, Protran calculates the relative travel times for stop n by cumulating the travel times between the stop of current stop index to stop n. The absolute travel time is calculated by cumulating the travel times between the stop of current stop index to stop n w.r.t the absolute time of the system. The travel times for the stops and destination are recalculated and sent for display after each stop index change.

### Status Values

For the Answer configuration (see chapter 4.31.2) the following status values can be used. These values are set depending on the state of the DS021a collection.

* Ok - Default state if everything is OK.
* IncorrectRecord This answer is immediately sent to the master when a DS021a with an invalid structure is received.

In addition to an immediate answer to DS021a, also the system state (DS020) is updated:

* NoData - If a DS021a is received after a final or timeout flush (see above) that has an index value different from FirstStopIndexValue.
* MissingData - If in the sequence of DS021a telegrams a stop index is missing

### DS021a ‘A‘ variation

|  |  |  |
| --- | --- | --- |
| **Tag name** | **Sub-tags allowed** | **Attributes allowed** |
| Connection | * UserFor * UsedForStopName * UsedForLineNumber * UsedForDepartureTime | Enabled |

Table 4‑36 Connection content

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table ‑ DS021a ‘A’ variant attribute

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1, placeholder allowed). |
| UsedForStopName | no | Defines how the StopName of this telegram is used (see chapter 4.31.1, placeholder allowed). |
| UsedForLineNumber | no | Defines how the Linenumber of this telegram is used (see chapter 4.31.1, placeholder allowed). |
| UsedForDepartureTime | no | Defines how the Departure time of this telegram is used (see chapter 4.31.1, placeholder allowed). |
| LineNumberFormat | no | String format for the line number. |
| ShowForNextStopOnly | no | Show connection information only for the next stop. Possible values: true, false |

Table 4‑38 DS021a ‘A’ variant sub-tags

Protran collects the connection records retrieved from DS021a ‘A’ variant telegram and caches it. Protran will only send the stop list information to Infomedia in the following cases:

1. The last connection info for one stop was not yet received, but Protran didn’t receive a DS021a for timeout seconds (“timeout flush”).
2. An update of connection info for a stop was received and Protran sends the complete set of connection info for the stop after the timeout.
3. A DS010b with a different stop index is received and Protran has connection information available for the “next” stop. If no connection information is available, the connection information is cleared.
4. Whenever Protran receives a DS021a with the stop index equal to first stop index “A0”, it clears all cached data and sends empty cells to Infomedia.
5. Whenever Protran receives a DS021a with the connection record with only spaces, it sends the complete set of connection info for the stop with Null values for that connection record.

DS021a payload is split into four columns using “;” as delimiter:

1. Stop index (UsedForStopName)
2. Line number (UsedForLineNumber)
3. Time to change (UsedForDepartureTime)
4. Destination (UsedFor)

For each DS021a telegram this information is stored and sent to Infomedia according to the configured usage.

The line number (“LineNumberFormat”) can refer to an image file to be shown in Infomedia. To define the file name, a format string can be provided. In the format string “{0}” is replaced with the text coming from the telegram.

When the connection info for a stop is sent, it has to be sorted based on the following

1. Connection records with “Time to change” = 0 are deleted and not sent to Infomedia.
2. Connection records are sorted based on “Time to change” with the record with minimum “Time to change” placed on the top row.

When there are gaps in the row indexes for the connection records, Protran must transfer the connection set with continuous row indexes to Infomedia.

### DS021A ‘S‘ variation

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedForText | no | Defines how the payload of this telegram is used (see chapter 4.31.1, placeholder allowed). |

Table 4‑39 DS021a ‘S’ variant sub-tag

Protran collects the text retrieved from DS021a ‘S’ variant telegram and caches it. Protran concatenates the text received from indexes (S1 to S9) into a single text to be sent to Infomedia. Protran will only send the payload text to Infomedia in the following cases:

1. The text with index S9 was not yet received, but Protran didn’t receive a DS021a for timeout seconds (“timeout flush”).
2. An update of text for an index was received and Protran sends the complete concatenated text.
3. Whenever Protran receives a DS021a with the stop index equal to first stop index “S0”, it deletes the complete concatenated text and sends empty cells to Infomedia.
4. If the text payload for an index contains only spaces, Protran replaces the text at the index with spaces and send the concatenated text to Infomedia.

## DS021c

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be an array of strings. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑40 DS021c attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| FlushNumberOfStations | no | Number of stops to collect before sending an update to Infomedia. Default: 5 |
| FlushTimeout | no | Time to wait before finishing the collection of stops. Default: PT30S |
| HideLastStop | no | Hide the last stop of the stop list. Default: false |
| HideDestinationBelow | no | Hide the destination (see UsedForDestination and so on) if less than this number of stations is shown. Default: 0 (disabled) |
| ShowPastStops | no | Put past stops into generic cells with negative row numbers. Default: false |
| TakeDestinationFromLastStop | no | Take the destination (see UsedForDestination and so on) from index “101” or from the last stop. Default: false (use index 101) |
| UsedFor | no | Usage of the first part in DS021c (see chapter 4.31.1, placeholder allowed). |
| UsedForTransfers | no | Usage of the second part in DS021c (see chapter 4.31.1, placeholder allowed). |
| UsedForTransferSymbols | no | Usage of the third part in DS021c (see chapter 4.31.1, placeholder allowed). |
| UsedForDestination | no | Usage of the first part in the last entry of DS021c (see chapter 4.31.1, no placeholder) |
| UsedForDestinationTransfers | no | Usage of the second part in the last entry of DS021c (see chapter 4.31.1, no placeholder) |
| UsedForDestinationTransferSymbols | no | Usage of the third part in the last entry of DS021c (see chapter 4.31.1, no placeholder) |
| AsciiLineNumberUsedFor | no | Usage of contents of DS021c with index “102” (see chapter 4.31.1, no placeholder) |
| Answer | no | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |

Table 4‑41 DS021c sub-tags

The handling of stop list telegrams can be configured very specifically for each customer. Therefore you have a lot of possible parameters. In the following paragraphs you find an explanation of the algorithm and how it uses the given parameters.

DS021c is handled similar to DS021a; for easier readability this chapter still contains all information needed for DS021c and therefore contains a lot of duplication from the previous chapter.

### Telegram Collection

Protran collects the information retrieved from DS021c and caches it. To prevent the stop list from “building up” one line at the time, Protran will only send the stop list information to Infomedia in the following cases:

1. When a DS021c with a status value of “0” is received, the stop list is cleared (“full clear”)
2. A multiple of FlushNumberOfStations has been collected (“intermediate flush”)
3. A DS021c with a status value of “2” is received (“last flush”)
4. The last telegram was not yet received, but Protran didn’t receive a DS021c for FlushTimeout seconds (“timeout flush”)
5. A DS010j with status “2” and a stop index different from the previous one is received (“index update flush”)
6. When a DS010j with status “3” is received, only the current stop information is cleared.
7. When a DS010j with the value “0” is received, all the stops shown on the Perlschnur are cleared but not deleted.

After a “last” or “timeout” flush Protran does not collect any DS021c anymore until it receives again a telegram with status “0” (see case 1 above).

It is suggested to set FlushNumberOfStations to the number of stops shown in the stop list (“Perlschnur”) in Infomedia.

### Stop Information

DS021c is split into up to five columns usually using <03>, <04>, <05> and # as delimiters:

1. Status code followed by <03>
2. Stop index (3 digits) followed by <04>
3. Stop name followed by <05> (UsedFor)
4. Transfer information followed by “#” (UsedForTransfers)
5. Transfer symbol list delimited by “;” (UsedForTransferSymbols)

For each DS021c telegram this information is stored and sent to Infomedia according to the configured usage.

If the customer wishes not to display the last stop since this is the same information shown in the destination (see below), it is possible to always filter out the last stop by setting HideLastStop to true.

If the customer wishes to display past stops using negative row numbers it is possible to enable the feature by setting ShowPastStops to true.

### Special stop index values

The following stop index values in DS021c are handled specially:

* 100: The start stop is currently ignored
* 101: The destination information is used as described in section 0 below
* 102: The ASCII line number is used if AsciiLineNumberUsedFor is configured
* Any index below 1 or above 102 is ignored

### Destination Information

The stop with index 101 can be used as destination, if the IBIS master does not provide this information otherwise (e.g. DS003a). The information is retrieved exactly the same way as the stop information (see above) and is configured using UsedForDestination, UsedForDestinationTransfers and UsedForDestinationTransferSymbols respectively. If the customer does not provide stop index 101 or he prefers to use the last stop as destination information (same as DS021a), set TakeDestinationFromLastStop to true.

If the customer wishes not to display the destination when it is already visible as the last entry in the stop list, the destination can be hidden by setting HideDestinationBelow to the number of stops visible in the stop list **plus one**. For example if your “Perlschnur” contains 5 stops plus the destination, you set the value to 6 and then Protran will hide all destination information if there are less than 6 stops in the stop list. This behavior is disabled by setting the value to 0 or removing the sub-tag.

## DS036

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** | |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be a number. | | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | | true, false |

Table 4‑42 DS036 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how this telegram is used (see chapter 4.31.1). |
| Answer | no | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |
| AutoReset | no | If this value is true, the value of the given Ximple cell is reverted to an empty immediately after setting it. Default: true |

Table 4‑43 DS036 sub-tags

When a DS036 is received, Protran sends the received value in the cell configured by “UsedFor”.

If “AutoReset” is set to true (default), approximately 0.2 seconds after receiving the DS036, Protran will automatically send another cell with an empty value (“”) in the cell configured by “UsedFor”. This feature should be enabled when the value from DS036 is used in Infomedia to trigger an event cycle. Like this, the following improved behavior can be observed:

* The event cycle is re-triggered even if exactly the same value is sent again from the board computer in a DS036 (every DS036 will trigger the event cycle)
* When Protran or the Audio Renderer restarts, it will not replay the last received announcement index (DS036)

## DS080

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑44 DS080 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| OpenValue | no | Defines the value to be sent when the DS080 is received. Default value: 1 |
| CloseValue | no | Defines the value to be sent when a DS010b with different index is received. Default value: 0 |
| ResetWithDS010b | no | Defines if DS010b should be used to trigger sending the close value. Possible values: true, false |
| UsedFor | no | Defines how this telegram is used (see chapter 4.31.1). |

Table 4‑45 DS080 sub-tags

This telegram does not have a payload. The values put into the Ximple cell defined by “UsedFor” is defined by “OpenValue” and “CloseValue”.

The “CloseValue” is only sent if “ResetWithDS010b” is set to true and a DS010b is received with an index different than the previous. Otherwise the value can only be reset by DS081 (see chapter 4.22 below).

## DS081

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑46 DS081 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| Value | no | Defines the value to be sent when the DS081 is received. Default value: 0 |
| UsedFor | no | Defines how this telegram is used (see chapter 4.31.1). |

Table 4‑47 DS081 sub-tags

This telegram does not have a payload. The value put into the Ximple cell defined by “UsedFor” is defined by “Value”.

This telegram is usually used in conjunction with DS080 (see chapter 4.22 above).

## GO001

The GO001 telegram is used in special projects to send events from the IBIS master. Currently only address 0, event code 1 is implemented (see telegram specification for more information).

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be a number. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑48 GO001 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1). |

Table 4‑49 GO001 sub-tags

When a GO001 is received in with the right event code, Protran sends value “1” in the cell configured by “UsedFor”.

When a DS010b is received with an index different than the previous Protran sends value “0” in the cell configured by “UsedFor”.

## GO002

The GO002 telegram is used to send connection information about future stops from the IBIS master.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑50 GO002 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| CheckLength | no | Check the length when parsing the telegram. Possible values: true, false |
| StopIndexSize | no | Size of the stop index part in the telegram. Default: 2 |
| RowNumberSize | no | Size of the row number part in the telegram. Default: 1 |
| PictogramSize | no | Size of the pictogram part in the telegram. Default: 1 |
| LineNumberSize | no | Size of the line number part in the telegram. Default: 5 |
| TrackNumberSize | no | Size of the track number part in the telegram. Default: 2 |
| ScheduleDeviationSize | no | Size of the schedule deviation part in the telegram.  Default: 4 |
| FirstStopIndex | no | Index of the first stop in the DS021a telegram. Default: 1 |
| FirstRowIndex | no | Index of the first row in the GO002 telegram. Default: 1 |
| LastRowIndex | no | Index of the last row in the GO002 telegram. Default: 9 |
| PictogramFormat | no | String format for the pictogram. |
| LineNumberFormat | no | String format for the line number. |
| ScheduleDeviation | yes | Schedule deviation information, see Table 4‑52. |
| DeletePassedStops | no | Delete connection information as soon as the stop index increases. Possible values: true, false |
| ShowForNextStopOnly | no | Show connection information only for the next stop. Possible values: true, false |
| UsedFor | no | Defines how the destination name of this telegram is used (see also chapter 4.31.1, placeholder allowed). |
| UsedForPictogram | no | Defines how the pictogram of this telegram is used (see also chapter 4.31.1, placeholder allowed). |
| UsedForLineNumber | no | Defines how the line number of this telegram is used (see also chapter 4.31.1, placeholder allowed). |
| UsedForDepartureTime | no | Defines how the departure time of this telegram is used (see also chapter 4.31.1, placeholder allowed). |
| UsedForTrackNumber | no | Defines how the track number of this telegram is used (see also chapter 4.31.1, placeholder allowed). |
| UsedForScheduleDeviation | no | Defines how the schedule deviation of this telegram is used (see also chapter 4.31.1, placeholder allowed). |
| Answer | no | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |

Table 4‑51 GO002 sub-tags

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| OnTime | yes | Value to be sent when schedule deviation is 0. |
| Ahead | yes | String format for value to be sent when schedule deviation is negative. |
| Delayed | yes | String format for value to be sent when schedule deviation is positive. |

Table 4‑52 ScheduleDeviation sub-tags

The handling of connection telegrams can be configured very specifically for each customer. Therefore you have a lot of possible parameters. In the following paragraphs you find an explanation of the algorithm and how it uses the given parameters.

### Telegram Parsing

The GO002 telegram has the following fields (with an example):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Header** | **Data length** | **Stop index** | **Row index** | **Pictogram** | **Line num.** | **Departure** | **Track** | **Schedule deviation** | **Destination** |
| aU | 999 | 03 | 1 | 1 | L0750 | 1425 | M1 | +002 | ST. GALLEN |

When parsing the telegram, the data length can be verified and the telegram discarded if the length is not correct. To enable this, set “CheckLength” to true.

The length (number of characters) of each field of the telegram can be configured (only if necessary). The following parameters allow overriding the default length for each field: “StopIndexSize”, “RowNumberSize”, “PictogramSize”, “LineNumberSize”, “TrackNumberSize” and “ScheduleDeviationSize”. The default values can be found in Table 4‑51.

#### Short GO002 Telegram

Apart from GO002 telegrams with the complete information as specified above, Protran supports

* Empty GO002 telegrams (data length = 000): aUXXX<CR><FP>
* GO002 telegrams with only the Data Length and Stop Index (data length = 002): aUXXXSS<CR><FP>

### Telegram Collection

Protran collects the information retrieved from GO002 and caches it. All connection information about the current route is stored, but only the information about the “next” (see below) stop is sent to Infomedia.

In this section we refer to “next” stop with the following conditions:

* The current DS010b or DS010j stop index is equal to the stop index sent in GO002
* The current DS010b or DS010j stop index is less than the stop index sent in GO002 **and** there is no other connection information available **and** “ShowForNextStopOnly” is set to “false” (default).

Whenever Protran receives a DS021a with the stop index equal to “FirstStopIndex”, it clears all cached data and sends empty cells to Infomedia.

To prevent the connection information from “building up” one line at the time, Protran will only send the connection information in the following cases:

1. A GO002 with a row index equal to “LastRowIndex” is received and the stop index refers to the “next” stop.
2. The currently shown connection list is updated with one GO002 with the same stop index.
3. A DS010b or DS010j with a different stop index is received and Protran has connection information available for the “next” stop. If no connection information is available, the connection information is cleared.

The first row of the connection table can be referred to with “0” or “1”, this can be configured with “FirstRowIndex”.

Connection information for the last stop (and all previous stops) is deleted if “DeletePassedStops” is set to “true” and the stop index changes (backwards or forwards). This guarantees that connection information for a given stop is only shown until that stop was reached for the first time. Resetting the index to a lower value will not bring back “old” connection information.

Both the pictogram (“PictogramFormat”) as well as the line number (“LineNumberFormat”) can refer to an image file to be shown in Infomedia. To define the file name, a format string can be provided. In the format string “{0}” is replaced with the text coming from the telegram. Example:

Configuring “D:\Infomedia\Layout\Symbols\Picto\_{0}.png” for “PictogramFormat” will result in the following value being sent to Infomedia if the Pictogram string in the telegram is “2”:  
“D:\Infomedia\Layout\Symbols\Picto\_2.png”

The same formatting applies to schedule deviations, but here three different formats can be defined:

* “ScheduleDeviation 🡪 OnTime”: Fixed string that is shown when the connection is on time.
* “ScheduleDeviation 🡪 Ahead”: Format string that is shown when the connection is ahead
* “ScheduleDeviation 🡪 Delayed”: Format string that is shown when the connection is delayed

#### Empty GO002 telegram

Upon reception of an empty GO002 telegram, Protran deletes all the connection information available for all the stops.

#### GO002 telegram with Stop Index

Upon reception of a GO002 telegram with only a Stop Index, Protran deletes all the connection information associated with that Stop Index.

### Connection Information

The different fields of the telegram can be used to fill different Ximple cells:

* Destination: “UsedFor”, value is sent through the configured transformation chain (see “TransfRef” above)
* Pictogram: “UsedForPictogram”, value is formatted with “PictogramFormat”
* Line number: “UsedForLineNumber”, value is formatted with “LineNumberFormat”
* Departure time: “UsedForDepartureTime”, value is always formatted as “HH:mm”
* Track number: “UsedForTrackNumber”, value is sent as-is
* Schedule deviation: “UsedForScheduleDeviation”, value is formatted according to the rules defined above

## GO003

The GO003 telegram is used in special projects to send next stop information for multiple languages in a single telegram.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be an array of strings. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑53 GO003 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1, placeholder allowed). |
| Answer | no | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |

Table 4‑54 GO003 sub-tags

When a GO003 is received it is transformed into multiple cells with the configured transformation chain (“TransfRef”) and then sent to Infomedia in the cells defined with “UsedFor”.

## GO004

The GO004 telegram is used to send multiple text messages to the Topbox.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be an array of strings. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑55 GO004 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Defines how the payload of this telegram is used (see chapter 4.31.1, placeholder allowed). |
| UsedForTitle | no | Defines how the title of this telegram is used (see chapter 4.31.1, placeholder allowed). |
| UsedForType | no | Defines how the message type of this telegram is used (see chapter 4.31.1, placeholder allowed). |
| Answer | no | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |

Table 4‑56 GO004 sub-tags

When a GO004 is received it is checked for its validity looking at the time range field. Only messages that are currently valid will be sent out by Protran.

The message field has to be split into multiple parts with the given transformation chain. The first of these parts will be used to fill the UsedForTitle cell, the remaining parts are concatenated using “[br]” and then used to fill the UsedFor cell.

The UsedForType cell is directly taken from the telegram; leading zeros are removed.

The given cells’ rows are always filled from top to bottom without gaps. If a message is removed (or becomes invalid), the following messages are moved up by one row. This guarantees that always the top-most rows contain valid messages and thus simplifies cycle management. The rows are always ordered by the message index from the GO004 telegram.

## GO005

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be an array of strings. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑57 GO005 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| FlushNumberOfStations | no | Number of stops to collect before sending an update to Infomedia. Default: 5 |
| FlushTimeout | no | Time to wait before finishing the collection of stops. Default: PT30S |
| HideLastStop | no | Hide the last stop of the stop list. Default: false |
| HideDestinationBelow | no | Hide the destination (see UsedForDestination) if less than this number of stations is shown. Default: 0 (disabled) |
| ShowPastStops | no | Put past stops into generic cells with negative row numbers. Default: false |
| UsedFor | no | Usage of the stop name in GO005 (see chapter 4.31.1, placeholder allowed). |
| UsedForDestination | no | Usage of the stop name in the last entry of GO005 (see chapter 4.31.1, no placeholder) |
| AsciiLineNumberUsedFor | no | Usage of contents of the line number of the current stop in GO005 (see chapter 4.31.1, no placeholder) |
| Answer | no | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |
| BufferNextRoute | no | Buffers a newly arriving route until a new stop index with a smaller value is received. Default: true |
| HideNextStopForIndex | no | Hides the next stop if the given index is received. Default: 999 |
| DeleteRoute | no | This flag tells Protran whether it has to take care about the route’s Run number value and the stop index to clear the stops. The values admitted are: true, false. The default value is true. If enabled, Protran will clear the stops only if the current route’s Run number value is equal to 0 **and** the stop index received is equal to zero. If not enabled, Protran behaves as usual. This parameter is currently useful only for the STOAG customer. |
|  |  |  |

Table 4‑58 GO005 sub-tags

The handling of stop list telegrams can be configured very specifically for each customer. Therefore you have a lot of possible parameters. In the following paragraphs you find an explanation of the algorithm and how it uses the given parameters.

GO005 is handled similar to DS021a; for easier readability this chapter still contains all information needed for GO005 and therefore contains a lot of duplication from the other chapter.

Be aware that this telegram has the same header as DS021 and

GO007, so always make sure to only enable DS021 or GO005 or

GO007 but never more than one of these at the same time.

### Telegram Collection

Protran collects the information retrieved from GO005 and caches it. To prevent the stop list from “building up” one line at the time, Protran will only send the stop list information to Infomedia in the following cases:

1. A multiple of FlushNumberOfStations has been collected (“intermediate flush”)
2. A GO005 without <04> is received (“last flush”)
3. The last telegram was not yet received, but Protran didn’t receive a GO005 for FlushTimeout seconds (“timeout flush”)
4. A DS010 with a stop index different from the previous one is received (“index update flush”)

After a “last” or “timeout” flush Protran does not collect any GO005 anymore until it receives again telegram with index “1”.

It is suggested to set FlushNumberOfStations to the number of stops shown in the stop list (“Perlschnur”) in Infomedia.

### Stop Information

GO005 is split into two columns usually using <04> as delimiter:

1. Line number (4 digits, AsciiLineNumberUsedFor) and index (4 digits) followed by <04>
2. Stop name followed by <LF> and padding spaces (UsedFor)

For each GO005 telegram this information is stored and sent to Infomedia according to the configured usage. The line number (AsciiLineNumberUsedFor) is always updated to the line number for the current stop.

If the customer wishes not to display the last stop since this is the same information shown in the destination (see below), it is possible to always filter out the last stop by setting HideLastStop to true.

Some customers send a special stop index (using DS010) to tell that the bus is currently at the stop with open doors. In this case the current stop is hidden. This stop index is set using HideNextStopForIndex, by default the value is 999. To disable this feature, set the value to 0.

If the customer wishes to display past stops using negative row numbers it is possible to enable the feature by setting ShowPastStops to true.

### Destination Information

The last stop of the stop list can be used as destination, if the IBIS master does not provide this information otherwise (e.g. DS003a). The information is retrieved exactly the same way as the stop information (see above) and is configured using UsedForDestination.

If the customer wishes not to display the destination when it is already visible as the last entry in the stop list, the destination can be hidden by setting HideDestinationBelow to the number of stops visible in the stop list **plus one**. For example if your “Perlschnur” contains 5 stops plus the destination, you set the value to 6 and then Protran will hide all destination information if there are less than 6 stops in the stop list. This behavior is disabled by setting the value to 0 or removing the sub-tag.

### Route Buffering

Certain IBIS masters send the route information before the bus reaches the final destination. Therefore you can configure Protran to cache the newly arriving route without sending it immediately to Infomedia by setting BufferNextRoute to true. The old route will still be handled as before (e.g. DS010 index updates) until a stop index (DS010) with a lower index (excluding the special value of HideNextStopForIndex) is received. At this point the whole stop list is updated at once.

### Route Deletion

This customer STOAG, populates the various stops of one (or more) route as desired by sending GO005 telegrams. To delete a route, Protran has to receive the route’s Run number equal to 0 **and** the stop index for the route equal to 0. Only when these conditions are both valid, Protran is allowed to clear the stops. The feature to delete a route is enabled or disabled by the configuration parameter DeleteRoute.

## GO006

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be a string. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑59 GO006 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| UsedFor | no | Usage of the line value (alphanumeric) in GO006 (see chapter 4.31.1, placeholder allowed). |
| Answer | no | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |

Table 4‑60 GO006 sub-tags

This telegram should be used only for the AbuDhabi project. Its meaning is related to the route’s line number. The IBIS master in the AbuDhabi project sends, for this kind of telegram, always 5 characters in the payload: the first and the last have to be always ignored and the remaining compose the line number (an alphanumeric value).

## GO007

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be an array of strings. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table ‑ GO007 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| HideLastStop | no | Hide the last stop of the stop list. Default: false |
| HideDestinationBelow | no | Hide the destination (see UsedForDestination) if less than this number of stations is shown. Default: 0 (disabled) |
| ShowPastStops | no | Put past stops into generic cells with negative row numbers. Default: false |
| UsedFor | no | Usage of the stop name in GO007 |
| UsedForTransfers | no | Usage of the connection information for each stop in GO007 |
| UsedForDestination | no | Usage of the stop name in the last entry of GO007 |
| UsedForDestinationTransfers | no | Usage of the connection information for the stop name in the last entry of GO007 |
| UsedForLineNumber | no | Usage of contents of the line number of GO007 |
| Answer | no | Defines if an answer should be sent when this telegram is received. |

Table 4‑62 GO007 sub-tags

The handling of stop list telegrams can be configured very specifically for each customer. Therefore you have a lot of possible parameters. In the following paragraphs you find an explanation of the algorithm and how it uses the given parameters.

GO007 has the same header as DS021 and GO005, so always make sure to only enable DS021 or GO005 or GO007 but never more than one of these at the same time.

### Telegram Collection

Protran collects the information retrieved from GO007 and sends only the line number to Infomedia at once. Protran will only send the stop list information to Infomedia when a DS009 with a stop name which is present in the list of stops received from GO007 is received.

### Stop and connection Information

GO007 is split into multiple columns usually using <03> and <04> as delimiter:

1. Line number (4 digits, UsedForLineNumber) followed by <03>
2. First stop information (UsedFor and UsedForTransfers) followed by <04>
3. Additional stops followed by <04> (except for the last stop)

For each GO007 telegram this information is stored and sent to Infomedia according to the configured usage upon reception of DS009 telegram. The line number (UsedForLineNumber) is always updated when a new GO007 telegram is received.

The stop name (UsedFor) and the transfer information (UsedForTransfers) are split internally by Protran using <05> in the second and subsequent columns. If no <05> is present in the stop information, the entire string is taken as the stop name and the transfer information is left empty.

If the customer wishes not to display the last stop since this is the same information shown in the destination (see below), it is possible to always filter out the last stop by setting HideLastStop to true.

If the customer wishes to display past stops using negative row numbers it is possible to enable the feature by setting ShowPastStops to true.

### Destination Information

The last stop of the stop list can be used as destination. The information is retrieved exactly the same way as the stop information (see above) and is configured using UsedForDestination and UsedForDestinationTransfers.

If the customer wishes not to display the destination information when it is already visible as the last entry in the stop list, the destination can be hidden by setting HideDestinationBelow to the number of stops visible in the stop list **plus one**. For example if your “Perlschnur” contains 5 stops plus the destination, you set the value to 6 and then Protran will hide all destination information if there are less than 6 stops in the stop list. This behavior is disabled by setting the value to 0 or removing the sub-tag.

### Route Change

To change the route, Protran has to receive a new GO007 telegram with stops list. The old stops list is only replaced by the new stops list when a stop name is received from DS009 telegram (not when the GO007 telegram is received).

## HPW074

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| TransfRef | yes | Refers to the **chain of transformations** that Protran will use to analyze the telegram’s payload (see chapter 5). The output of the transformation chain must be a number. | A string with custom value. Default: empty |
| Enabled | yes | Tells if the telegram has to be recognized by Protran or immediately discarded. | true, false |

Table 4‑63 HPW074 attributes

|  |  |  |
| --- | --- | --- |
| **Sub-tags** | **Mandatory** | **Meaning** |
| SpecialTextFile | no | Path of the CSV file containing the special texts. Default: D:\infomedia\layout\specialtext.csv |
| Encoding | no | Encoding of the CSV file. Default UTF-8 |
| UsedFor | no | Defines how the resulting special text is used (see chapter 4.31.1). |
| Answer | no | Defines if an answer should be sent when this telegram is received (see chapter 4.31.2). |

Table 4‑64 HPW074 sub-tags

When a HPW074 telegram is received, the index from the telegram is used to look up a special text in the configured special text file (“SpecialTextFile”) which needs to have the following format:

1;Hello;World;

2;Important;Don’t forget your luggage;

3;Beware of thieves;Keep an eye on your luggage;

Figure 4‑2 Example special text file

If the index is found, the text on the same row as the index is transmitted to Infomeda in the cell defined by “UsedFor”. All semicolons (“;”) are replaced with newline BBCode (“[br]”).

If the index doesn’t exist, an empty cell is sent to Infomedia instead.

## Common Sub-Tags

### Generic Usage

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag’s attributes** | **Mandatory** | **Meaning** | **Values admitted (default value)** |
| Language | no | Refers to the generic language of the cell to be filled. | A string with custom value. Default: 0 |
| Table | yes | Refers to the generic table of the cell to be filled. | A string with custom value. Default: empty |
| Column | yes | Refers to the generic column of the cell to be filled. | A string with custom value. Default: empty |
| ColumnOffset | no | The offset of placeholder column number. | Integer. Default: 1 |
| Row | no | Refers to the generic row of the cell to be filled. | Integer. Default: 0 |
| RowOffset | no | The offset of placeholder row number. | Integer. Default: 0 |

Table 4‑65 Generic usage attributes

Generic usage tags are used in ibis.xml whenever a generic cell has to be referenced.

The “Table” and “Column” attributes can either be indexes (numbers) or names. If a name is specified, it will be looked up in the dictionary.xml to be translated to an index.

The optional “Row” attribute is the row number of the generic cell that will be filled.

Indexes can be direct or with a placeholder. When a row (or a column) has to be different for each element, the placeholder “{0}” can be used. It is then replaced with the according index from the telegram (see each telegram for more details). The offset of the index (its first value) can be defined with “ColumnOffset” and “RowOffset” respectively.

### IBIS Answers

In this chapter is detailed how to configure Protran to send back an answer to the IBIS master depending on specific received telegrams. All the telegrams are contained into the tag <Telegrams>.

Here below is shown a good example:

<Ibis>

<Telegrams>

<DS020 Enabled="true">

<Answer>

<DS120 Enabled="true">

<Response Status="NoData">3</Response>

<Response Status="MissingData">4</Response>

<DefaultResponse>0</DefaultResponse>

</DS120>

</Answer>

</DS020>

</Telegrams>

</Ibis>

Figure 4‑3 Example of telegrams configuration in IBIS.XML

In the example above, you can see that Protran is configured to detect only one IBIS telegram: the DS020. It is enabled and whenever Protran receives that telegram, it has to react sending back to the IBIS master an answer. The answer is made by another telegram: the DS120.

Again keeping on sight the example above, the content of the answer is:

* Set to 3, if Protran is currently on the state “**NoData**”
* Set to 4, if Protran is currently on the state “**MissingData**”
* Set to 0, if Protran is currently on a state different from “**NoData**” and from “**MissingData**.

The state:

* “**NoData**” means that Protran has never received data from the IBIS master.
* “**MissingData**” means that Protran has received some data from the IBIS master but not all of them.
* “**IncorrectRecord**” means that Protran has just received a telegram with invalid data inside.

So, to customize Protran to send an answer to the IBIS master whenever a specific telegram is detected, the steps to do are:

1. Create a tag with the telegram’s name to detect and to which reply with an answer

<DS020></DS020>

1. Enable the telegram

<DS020 Enabled="true"></DS020>

1. Add an answer to the telegram

<DS020 Enabled="true">

<Answer></Answer>

</DS020>

1. Specify the telegram that has to be used as answer (DS120)

<DS020 Enabled="true">

<Answer>

<DS120>

</DS120>

</Answer>

</DS020>

1. Enable the answer

<DS020 Enabled="true">

<Answer>

<DS120 Enabled="true">

</DS120>

</Answer>

</DS020>

1. Give the response value to the telegram used as answer depending on the Protran’s status.

<DS020 Enabled="true">

<Answer>

<DS120 Enabled="true">

<Response Status="IncorrectRecord">3</Response>

<DefaultResponse>0</DefaultResponse>

</DS120>

</Answer>

</DS020>

Following the steps above, Protran is correctly configured to detect the DS020 and to send back to the IBIS master the DS120 as answer with a specific value.

# VDV 301 – IBIS-IP

The VDV 301 (or “IBIS over IP”) protocol is a standard by the Verband Deutscher Verkehrsunternehmen and was first published in 2014. The goal of the protocol is to replace the outdated VDV 300 protocol, also known as IBIS or Wagenbus (see above).

This manual assumes that you have good knowledge of the working of VDV 301; therefore it is suggested to read the standard (VDV 301-1 and VDV 301-2) or at least the “IBIS-IP Primer” to understand the protocol.

The following chapters will describe use cases of VDV 301 that might be required in projects with the relevant configuration. It is important to note that each use case only covers its own configuration, so most often the different configurations must be combined to have a complete valid customer configuration. All use cases concentrate on the service, operation and data item configuration as well as the suggested transformations; all other parts of the configuration file are left out.

## Stop List

The stop list is provided by CustomerInformationService.GetAllData and will be automatically put into rows by Protran. Therefore the configuration is absolutely straight forward:

    <CustomerInformationService>  
      <GetAllData Subscribe="true">  
        <TripInformation>  
          <StopSequence>  
            <StopPoint>  
              <StopName Enabled="true" Table="Stops" Column="StopName" Row="{0}" />  
            </StopPoint>  
          </StopSequence>  
        </TripInformation>  
      </GetAllData>  
    </CustomerInformationService>

Using the placeholder “{0}” for the row number will put the stops into the right row according to the current stop index (GetAllData > CurrentStopIndex). The stop index itself is not required to be configured, this is taken care of by the handling of StopSequence in Protran.

## Destination Information

The destination information is well hidden inside the display content of the current stop. This was done by VDV 301 so that different content for the exterior sign(s) can be shown depending on the current stop. For this reason, the configuration goes deeper into the XML tree:

    <CustomerInformationService>  
      <GetAllData Subscribe="true">  
        <TripInformation>  
          <StopSequence>  
            <StopPoint>  
              <DisplayContent>  
                <Destination>  
                  <DestinationName Enabled="true" Table="Destination"   
 Column="DestinationName" Row="0" />  
                </Destination>  
              </DisplayContent>  
            </StopPoint>  
          </StopSequence>  
        </TripInformation>  
      </GetAllData>  
    </CustomerInformationService>

## Line Information

The line information exists in three different elements in VDV 301: LineNumber, LineName or LineShortName. Depending on the project one of the three (or multiple) have to be used for the line number. Like the destination information (see chapter 5.2), the line number is also sent in the display content:

    <CustomerInformationService>  
      <GetAllData Subscribe="true">  
        <TripInformation>  
          <StopSequence>  
            <StopPoint>  
              <DisplayContent>  
                <LineInformation>  
                  <LineNumber Enabled="true" Table="Route" Column="Line" Row="0" />  
                </LineInformation>  
              </DisplayContent>  
            </StopPoint>  
          </StopSequence>  
        </TripInformation>  
      </GetAllData>  
    </CustomerInformationService>

## Connection Information

Connection information usually consists of the following information:

* Destination where the connection is going
* Platform where the connection is leaving
* Time when the connection is leaving
* Type of transport (e.g. is it a bus or a train)

All this information (and much more) is available for every stop in CustomerInformationService. GetAllData. The configuration can be done as follows:

    <CustomerInformationService>  
      <GetAllData Subscribe="true">  
        <TripInformation>  
          <StopSequence>  
            <StopPoint>  
              <StopName Enabled="true" Table="Stops" Column="StopName" Row="{0}" />  
              <DisplayContent>  
                <Destination>  
                  <DestinationName Enabled="true" Table="Destination"  
  Column="DestinationName" />  
                </Destination>  
              </DisplayContent>  
              <Connection>  
                <DisplayContent>  
                  <Destination>  
                    <DestinationName Enabled="true" Table="Connections"  
  Column="ConnectionDestinationName" Row="{0}"/>  
                  </Destination>  
                </DisplayContent>  
                <Platform Enabled="true" Table="Connections"  
  Column="ConnectionPlatform" Row="{0}"/>  
                <ExpectedDepatureTime Enabled="true" DateTimeFormat="HH:mm"  
 Table="Connections" Column="ConnectionTime"  
  Row="{0}"/>  
                <TransportMode>  
                  <Name Enabled="true" Table="Connections"  
  Column="ConnectionTransportType" Row="{0}"/>  
                </TransportMode>  
              </Connection>  
            </StopPoint>  
          </StopSequence>  
        </TripInformation>  
      </GetAllData>  
    </CustomerInformationService>

Protran will always use the information from the next stop (independent whether there is any connection information).

## Vehicle States

VDV 301 provides information about several states of the system. Most of those are sent as enumerations and therefore need to be converted. The example below uses the following state information and forwards them in the respective Ximple cells to Infomedia:

* Door status (open / closed)
* Stop request
* Exit side (mostly used in trains)

The service configuration can be done as follows:

    <CustomerInformationService>  
      <GetAllData Subscribe="true">  
        <DoorState Enabled="true" Table="SystemStatus" Column="DoorStatus"  
 TransfRef="DoorStatus" />  
        <VehicleStopRequested Enabled="true" Table="SystemStatus"  
 Column="StopRequestedState"/>  
        <ExitSide Enabled="false" Table="SystemStatus" Column="ExitSide"  
 TransfRef="ExitSide"/>  
      </GetAllData>  
    </CustomerInformationService>

In addition these transformations are required to map the VDV 301 enumeration values to Ximple cell values:

    <Chain id="DoorStatus">  
      <Replace>  
        <Mapping from="AllDoorsClosed" to="0"/>  
        <Mapping from="DoorsOpen" to="1"/>  
        <Mapping from="SingleDoorOpen" to="1"/>  
        <Mapping from="SingleDoorClosed" to="1"/>  
      </Replace>  
    </Chain>

    <Chain id="ExitSide">  
      <Replace>  
        <Mapping from="unknown" to="0"/>  
        <Mapping from="right" to="1"/>  
        <Mapping from="left" to="2"/>  
        <Mapping from="both" to="3"/>  
      </Replace>  
    </Chain>

The stop request (VehicleStopRequested) doesn’t require a mapping since Boolean values are understood by Infomedia. Of course the mapping “ExitSide” is only an example and would require the Infomedia project to be configured accordingly.

# Telegram’s Transformations

Sometimes it is required to manipulate the payload of a received telegram in order to change it for custom purposes. Let’s consider the time telegram DS005. The payload of it is composed by 4 digits; the first 2 are referring to one hour and the last 2 are referring to the minutes. For example, suppose we receive “1234” from the IBIS master. XIMPLE requires that the hours are separated from the minutes with a “:” character: “12:34”. This is a payload transformation.

Protran, correctly configured, can apply a lot of transformations to the telegram’s payload; **one or more than one** **following the specific order in which they are written in the configuration file** (top-down). After one transformation, the obtained result can still be manipulated by Protran, applying then a second transformation, a third and so on. For this reason, the desired transformations to be applied for a telegram must be defined into a “chain”. Below is presented how to configure Protran to apply the transformation about the time seen before:

It’s up to you to decide the name of the “TransfRef“.

 <Telegrams>

    <DS005 TransfRef="Time" Enabled="true">

    </DS005>

    <DS006 TransfRef="Date" Enabled="true">

      <UsedFor Table="System" Column="Date" Row="0" />

    </DS006>

</Telegrams>

 <Transformations>

    <Chain id="Time">

      <RegexMapping desc="Format HHmm to HH:mm">

        <Mapping from="^(\d\d)(\d\d)$" to="$1:$2"/>

      </RegexMapping>

    </Chain>

 </Transformations>

Figure 5‑1 Example Of Transformation For DS005

As you can see in the example above, a telegram can have another attribute: “**TransfRef**”. Literally, it means “reference to a transformation chain” and accepts free strings. It’s up to you to give a name to the “**TransfRef**” attribute and also it’s up to you that the name you’ve specified matches with a really existing chain, otherwise Protran will not be able to handle the telegram. An explanation of the XML above will be given later in the paragraph 5.5

Here below are presented all the possible transformations currently implemented in Protran:

* Capitalize
* ChainRef
* Integer
* Join
* LawoString
* RegexMapping
* RegexDivider
* Replace
* StringMapping

An explanation will be given for each of them in the following paragraphs.

Attention: don’t use different names from the following names (case sensitive):

<Capitalize>

<Integer>

<Join>

<RegexDivider>

<RegexMapping>

<StringMapping>

Figure 5‑2 Syntax of the available transformations in Protran

## Capitalize Transformation

The transformation allows the characters in a string to be converted to a different case based on the selected mode.

If there are strings that mustn’t be capitalized, exceptions can be added to this transformation.

Capitalize has an optional attribute “Mode” that can modify its behavior. The default mode is “UpperLower”. The possible modes are:

* UpperLower: makes the first character uppercase and the rest of the string lowercase
* UpperOnly: makes the first character uppercase and doesn’t change the rest of the string
* LowerOnly: Doesn’t change the first character, but makes the rest of the string lowercase

Since in general you want to capitalize words, you should always combine Capitalize with RegexDivider and Join. A complete example is shown below:

 <Telegrams>

    <DS003a TransfRef="MakeCapital" Enabled="true">

    </DS003a>

</Telegrams>

 <Transformations>

    <Chain id="MakeCapital">

      <RegexDivider desc="Split on non-word characters">

        <Regex>(\W+)</Regex>

      </RegexDivider>

      <Capitalize>

        <Exceptions>

          <string>von</string>

          <string>nach</string>

          <string>via</string>

          <string>SBB</string>

        </Exceptions>

      </Capitalize>

<Join/>

    </Chain>

 </Transformations>

Figure 5‑3 Example of Capitalize Transformation

With a transformation as the one shown above, a string like “sbb: zurich nach bern VIA LUZERN” will give the output: “SBB: Zurich nach Bern via Luzern”.

## ChainRef

ChainRef is actually not a transformation by itself, but refers to another chain in the configuration file. This allows for chains to be reused within other chains. Set the TransfRef attribute of the ChainRef to an id of another chain. Be careful about recursion since this would crash Protran at start-up without a meaningful error message.

    <Chain id="MyChain">

<!-- some transformations -->

      <ChainRef TransfRef="FullReplacement"/>

    </Chain>

    <Chain id="FullReplacement">

<!-— some other transformations -->

    </Chain>

Figure 5‑4 Example of Integer Transformation

## Integer Transformation

This kind of transformation gets an integer starting from a string. For example, the string “-1234” will return the integer -1234. The string “abcd” will make fail the transformation because it doesn’t contain only digits.

A complete example is shown below:

 <Telegrams>

    <DS002 TransfRef=" MakeInteger " Enabled="true">

    </DS002>

</Telegrams>

 <Transformations>

<Chain id="MakeInteger">

     <Integer />

    </Chain>

 </Transformations>

Figure 5‑5 Example of Integer Transformation

## Join Transformation

This kind of transformation takes an array of strings and combines them to a single string. For example, the strings “Hello”, “world” and “!”will be combined into “Hello world !”. In this example, the used separator character is the white space “ “. The default value for the separator is an empty string.

A complete example is shown below:

 <Telegrams>

    <DS003a TransfRef="Destination" Enabled="true">

    </DS003a>

</Telegrams>

 <Transformations>

    <Chain id="MakeJoin">

      <Join>

       <Separator>\*</Separator>

      </Join>

    </Chain>

 </Transformations>

Figure 5‑6 Example of Join Transformation

Given the input “Hello”, “world”, “!”, the output of the above transformation will be: “Hello\*world\*!”.

## LawoString Transformation

This kind of transformation takes an array of bytes (a raw telegram) and converts it using the Lawo string conversion rules:

* <01> (only relevant for the next character): +60hex
* <0B> (relevant for all next characters):  
  +60hex (the logic must be disabled in case there is again a <0B> available)
* <02> (only relevant for the next character): +A0hex
* <0C> (relevant for all next characters):  
  +A0hex (the logic must be disabled in case there is again a <0C> available)
* <06> control of the small/caps letter (toggling logic):   
  40hex…5Fhex will be 60hex…7Fhex meaning A..Z will become a..z

For example, the telegram contents “E<06>ICHBERG <06>H<01>$<06>RDLI” will be converted to “Eichberg Härdli”.

Since the input of this transformation is a raw telegram, it has to be the first transformation in a chain.

The conversion of values >128 (7Fhex) is done using the specified code page. Default is 1252 which is (for our purposes) identical to ISO-8859-1. Some board computers use the “OEM (Multilingual Latin I + Euro)” code page which has number 858.

A complete example configuration is shown below:

 <Telegrams>

    <DS021c TransfRef="Lawo" Enabled="true">

    </DS021c>

</Telegrams>

 <Transformations>

    <Chain id="Lawo">

      <LawoString>

       <CodePage>858</CodePage>

      </LawoString>

    </Chain>

 </Transformations>

Figure 5‑7 Example of LawoString Transformation

## RegexMapping Transformation

This kind of transformation applies a regular expression in order to change the format of the original string into a different one, following some rules.

A complete example is shown below:

 <Telegrams>

    <DS005 TransfRef="Time" Enabled="true">

    </DS005>

</Telegrams>

 <Transformations>

    <Chain id="Time">

      <RegexMapping desc="Format HHmm to HH:mm">

        <Mapping from="^(\d\d)(\d\d)$" to="$1:$2"/>

      </RegexMapping>

    </Chain>

 </Transformations>

Figure 5‑8 Example Of Transformation For DS005

This transformation at first studies the telegram’s payload if it matches with the “**from**” format and if so changes the original format according to “**to**”. The regular expressions, in computer science, is a powerful formal language used to specify, recognize string of text, such as particular characters, words or patterns of characters.

Chapter 6 provides more information about regular expressions, so hereafter will be given only a brief description of its usage in the example above.

<RegexMapping desc="Format HHmm to HH:mm">

<Mapping from="^(\d\d)(\d\d)$" to="$1:$2"/>

</RegexMapping>

Figure 5‑9 Example of RegexMapping

A **RegexMapping** is composed by one or more **Mapping**. The one above accepts each string having 4 consequently digits. The first 2 digits are grouped logically together in a group and the last 2 digits are grouped logically into a separate second group. The mapping takes the first 2 digits, than adds a “:” and then adds the last 2 digits.

## RegexDivider Transformation

This kind of transformation divides a string in multiple tokens basing on a specific regular expression’s rule defined in the sub-tag <Regex>.

A complete example is shown below:

 <Telegrams>

    <GO003 TransfRef="MakeRegDivide" Enabled="true">

    </GO003>

</Telegrams>

 <Transformations>

<Chain id="MakeRegDivide">

      <RegexDivider desc="Divides at # characters">

        <Regex>\u0023</Regex>

      </RegexDivider>

    </Chain>

 </Transformations>

Figure 5‑10 Example of RegexDivider Transformation

Basically, each time is encountered the character ‘\u0023’ (#), the transformation will divide the string. For the input string “Hello#world#!” the output is an array made by:

* Hello
* World
* !

You can find more information about regular expressions in chapter 6. For advanced users, the optional attribute “Options” allows for setting Regular Expression options. For possible values and their meaning, refer to:  
<http://msdn.microsoft.com/en-us/library/system.text.regularexpressions.regexoptions.aspx>

## Replace Transformation

This kind of transformation replaces the whole string with another string. The order with which the replacement is performed is **top-down**.

There is an optional attribute “CaseSensitive” which is by default “false”; if you set it to “true” the string must also match in case, not just in characters.

In most cases you want to use this after splitting the string with RegexDivider.

A complete example including a RegexDivider and a Join is shown below:

 <Telegrams>

   <DS003a TransfRef="MakeReplace" Enabled="true">

    </DS003a>

</Telegrams>

 <Transformations>

<Chain id="MakeReplace">

      <RegexDivider>

        <Regex>(\W+)</Regex>

      </RegexDivider>

      <Replace CaseSensitive="true">

        <Mapping from="Bhf" to="Bahnhof"/>

        <Mapping from="Stn" to="Station"/>

      </Replace>

      <Join/>

    </Chain>

 </Transformations>

Figure 5‑11 Example of Replace Transformation

## StringMapping Transformation

This kind of transformation replaces tokens within a string with other tokens. The order with which the replacement is performed is **top-down**.

A complete example for the standard IBIS Umlaut conversion is shown below:

 <Telegrams>

   <DS003a TransfRef="MakeStringMap" Enabled="true">

    </DS003a>

</Telegrams>

 <Transformations>

<Chain id="MakeStringMap">

      <StringMapping desc="IBIS conversion">

        <Mapping from="{" to="ä"/>

        <Mapping from="[" to="Ä"/>

        <Mapping from="|" to="ö"/>

        <Mapping from="\" to="Ö"/>

        <Mapping from="}" to="ü"/>

        <Mapping from="]" to="Ü"/>

        <Mapping from="~" to="ß"/>

      </StringMapping>

    </Chain>

 </Transformations>

Figure 5‑12 Example of Integer Transformation

Each time is encountered the character “{“, it will be replaced with “ä”, the character “}“ will be replaced with “ü” and so on.

A complete example for the French accents conversion is shown below:

 <Telegrams>

   <DS003a TransfRef="MakeStringMap" Enabled="true">

    </DS003a>

</Telegrams>

 <Transformations>

<Chain id="MakeStringMap">

      <StringMapping desc="IBIS conversion">

<Mapping from="~" to="é"/>

<Mapping from="$" to="è"/>

<Mapping from="?" to="ê"/>

<Mapping from="=" to="ë"/>

<Mapping from="&amp;" to="à"/>

<Mapping from="\_" to="â"/>

<Mapping from="+" to="ï"/>

<Mapping from="^" to="ç"/>

<Mapping from="&gt;" to="ô"/>

<Mapping from="%" to="ù"/>

<Mapping from="&lt;" to="û"/>

      </StringMapping>

    </Chain>

 </Transformations>

Figure 5‑13 Example of string mapping for French accents

# Regular Expressions in Telegram Transformations

Regular expressions are used in two cases in transformations:

1. RegexMapping
2. RegexDivider

The “RegexMapping” does a normal Regular Expression Replace operation. The “RegexDivider” does a normal Regular Expression Split operation.

Regular expressions are the most powerful way to modify strings. The whole “vocabulary” useable in “ibis.xml” is specified by Microsoft:

.NET Framework Regular Expressions: <http://msdn.microsoft.com/en-us/library/hs600312>

Regular Expression Language - **Quick Reference**: <http://msdn.microsoft.com/en-us/library/az24scfc>

This chapter can’t go through all details of Regular Expressions, but we rather concentrate on popular usage scenarios. In some advanced examples we show the more advanced users how to use the magic of regular expressions in more complex scenarios.

## Splitting words

When splitting a string into words, you can simply split on “non-word” characters (“\W”). Since we want to split on one or more non-word characters, we add the “+” sign and to “capture” (see below) the characters between the words, we put the expression in parenthesis:

<RegexDivider>

<Regex>(\W+)</Regex>

</RegexDivider>

Figure 6‑1 RegexDivider Transformation for word splitting

The capturing of the expression is required because in the resulting string we want to have the “non-word” characters back in their old place. If we left out the parenthesis, we would get a list of all words, but when joining, we wouldn’t know how to join the words back together (or all non-word characters would simply be lost when joining again).

### Advanced example

As defined in the following reference article, “\W” refers to everything but characters, digits and underscores: <http://msdn.microsoft.com/en-us/library/20bw873z#NonWordCharacter>

If you want to split words by underscores as well, you can define it with an extended character class:

<RegexDivider>

<Regex>([\W\_]+)</Regex>

</RegexDivider>

Figure 6‑2 RegexDivider Transformation for splitting including underscores

Of course any Regular Expression can be used if a more complex behavior is required.

## Splitting by special characters

Some characters like single space, newline and non-printable characters can’t be written in XML configurations. Here Regular Expressions help to describe these transformations in “ibis.xml”.

For example, splitting by newline (ASCII character 10 – or 0A in hexadecimal) can be done as follows:

<RegexDivider>

<Regex>\u000A</Regex>

</RegexDivider>

Figure 6‑3 RegexDivider Transformation for splitting by newlines

## Splitting DS021a, DS021c and GO005

These three telegrams contain different “columns” that are usually divided by ASCII 03, 04, 05 and the hash “#” character. A custom character class allows this splitting:

<RegexDivider>

<Regex>[\u0003-\u0005#]</Regex>

</RegexDivider>

Figure 6‑4 RegexDivider Transformation for splitting DS021a

Example:

aL800 <03>**01**<04>**BAHNHOF**<05>**INFORMATION**#**9;21;24;SBB**

|0| |1 | | 2 | | 3 | 4 |

Figure 6‑5 DS021a telegram divided into 5 parts with RegexDivider

### Advanced examples

Depending on the customer DS021a/b might be sent a bit differently:

* The ASCII 05 character is missing
* Multiple hash characters are used instead of a single one
* The transfer symbols are delimited with colons (“:”) instead of semicolons (“;”)

Thanks to Regular Expressions, this can be configured at your will, but these configurations get of course more complicated. Let’s investigate each case below.

#### ASCII 05 character is missing

Instead of getting the necessary five columns, we only get four, if we split by “[\u0003-\u0005#]” as shown above. Example:

aL800 <03>**01**<04>**BAHNHOF**#**9;21;24;SBB**

|0| |1 | | 2 | 3 |

Figure 6‑6 DS021a telegram wrongly divided into 4 parts with RegexDivider

The easiest way to get around this is by replacing the hash with a double hash. Then the divider will create an additional element between the stop name and the transfer symbols containing an empty string:

aL800 <03>**01**<04>**BAHNHOF**# #**9;21;24;SBB**

|0| |1 | | 2 |3| 4 |

Figure 6‑7 DS021a telegram wrongly divided into 4 parts with RegexDivider

To achieve this, we replace every hash character with two of them using a RegexMapping; since hash should only be available once in the telegram, this works:

<RegexMapping>

<Mapping from="#" to="##"/>

</RegexMapping>

<RegexDivider>

<Regex>[\u0003-\u0005#]</Regex>

</RegexDivider>

Figure 6‑8 Transformation chain for splitting DS021a with missing <05>

#### Multiple hash characters

Instead of getting the necessary five columns, we get six or more, if we split by “[\u0003-\u0005#]” as shown above. Example:

aL800 <03>**01**<04>**BAHNHOF**<05>**INFORMATION**# # #**9;21;24;SBB**

|0| |1 | | 2 | | 3 |4|5| 6 |

Figure 6‑9 DS021a telegram wrongly divided into too many parts with RegexDivider

This problem is the opposite of what is described in section 6.3.1.1. The solution is thus also just the inverse: before splitting, we replace multiple (one or more) occurrences of hash with a single one:

<RegexMapping>

<Mapping from="#+" to="#"/>

</RegexMapping>

<RegexDivider>

<Regex>[\u0003-\u0005#]</Regex>

</RegexDivider>

Figure 6‑10 Transformation chain for splitting DS021a with multiple hash characters

#### Transfer symbols are delimited with colons

To solve this issue, one might be tempted to just replace colons with semicolons using either a simple RegexMapping or even just a StringMapping. Unfortunately this would also replace colons in the stop name or transfer information, not just the transfer symbols.

The solution lies in a so-called “positive look-behind” (written as “(?<=…)”). With that we can define “replace a colon only when it is somewhere after the <05> character”:

<RegexMapping>

<Mapping from="(?&lt;=\u0005.\*):" to=";"/>

</RegexMapping>

<RegexDivider>

<Regex>[\u0003-\u0005#]</Regex>

</RegexDivider>

Figure 6‑11 Transformation chain for splitting DS021a and replacing “:” with “;” in the transfer symbols

N.B.: the “<” character of the regular expression positive look-behind has to be XML-escaped since it is not a valid character in an attribute; therefore you see the “&lt;” in the above example.

## Removing leading or trailing characters

A common case is to remove leading zeros from DS001:

<RegexMapping>

<Mapping from="^0+" to=""/>

</RegexMapping>

Figure 6‑12 RegexMapping Transformation for removing leading zeros

Of course also any other leading or trailing character (sequence) can be replaced using mappings.

To remove trailing spaces use the following mapping:

<RegexMapping>

<Mapping from=" +$" to=""/>

</RegexMapping>

Figure 6‑13 RegexMapping Transformation for removing trailing spaces

## Format date or time

DS005 and DS006 provide date and time information in a compact format which is not compatible with Infomedia.

For the time in DS005, it is enough to introduce a colon between the hours and minutes:

<RegexMapping>

<Mapping from="^(\d\d)(\d\d)$" to="$1:$2"/>

</RegexMapping>

Figure 6‑14 RegexMapping Transformation for DS005

The “$1” and “$2” expressions in the “to” attribute refer to the respective expressions in parenthesis in the “from”: the first two and the second two digits.

If DS006 is sent as 6-digit (2 digits day, 2 digits month and 2 digits year), it can easily be transformed as well with a mapping:

<RegexMapping>

<Mapping from="^(\d\d)(\d\d)(\d\d)$" to="$1.$2.20$3"/>

</RegexMapping>

Figure 6‑15 RegexMapping Transformation for DS006

Here we simply add the dots and “20” to get a four digit year.

# Telegram Usage

The telegram’s payload is interpreted by Protran, translated into XIMPLE and then sent to the Gorba’s media player. Suppose for example that the IBIS master sends the telegram DS001 with the destination code “003”. Protran receives it, parses it and sends it with XIMPLE to the media player. How this last application knows that “003” refers to the journey’s destination code? To answer to this question we need to consider in conjunction the files:

* ibis.xml
* dictionary.xml

The file **dictionary.xml** is out of the scope of this document, so, for the rest of this chapter we will consider its content as fixed and not changeable.

For each available telegram in Protran, there is the possibility to “add” to them additional information: how the telegram’s payload will be used by the media player. From the point of view of the Gorba’s media player, all the “showable things” (strings, images, video, scrolling texts…) have one 3D-coordinate. This coordinate mustn’t be intended as a graphical coordinate on the screen (x,y,z), rather to a coordinate in the media player’s database. This concept is explained with the following figure:

MediServer

Protran

    <DS001a Enabled="true">

      <UsedFor Table="**Infos**" Column="Line symbol" Row="0" />

    </DS001a>

    <DS021 Enabled="true">

      <UsedFor Table="**Stops**" Column="Stop city" Row="0" />

    </DS021>

Table For the **Stops**

Table For **Images**

Table For **Infos**

Infomedia‘s Database

Figure 7‑1 Example of Telegram Usages

First of all, you have to keep in mind that the media player has internally a database. This database is composed by tables, and each table is composed by rows and columns. “To give a **usage**”to a telegram it means basically to send its payload into a specific cell of that database. In **ibis.xml** you have to configure the table, the column and the row and whenever Protran receives a specific telegram, it will forward its payload to the media player specifying the desired cell.

One note: in the example above, you can see that was used the Table="**Stops**" and the Table="**Infos**". Also was used the Column="Stop city" and the Column="Line symbol". All those names are been used because they are used in the **dictionary.xml** file. This means that you can’t use any name for the table and neither for the columns. You must respect the names specified in **dictionary.xml** (this file is fixed, not configurable). It is also possible to specify the table or column number instead of its name.

To do this, there’s the need to add (at least) the sub-tag <UsedFor> with the attributes Table, Column and Row. Due to the importance of this configuration phase, following will be presented an example for each telegram in a real existing ibis.xml configuration file.

## DS001 Example Usage

This telegram contains information about the journey’s line number, so the ibis.xml file was configured to use it for the line number. Be aware that if you are using Code Conversions (see below), you need to leave the DS001 tag enabled but empty (no “UsedFor”).

    <DS001 TransfRef="Line" Enabled="true">

    <UsedFor Table="Trip" Column="Line" Row="0" />

    </DS001>

Figure 7‑2 Example of DS001 Usage

## DS005 Example Usage

This telegram contains information about the time. When Protran receives the time from the IBIS master it has also to synchronize the media player’s time. But this feature is already completely fulfilled by the <TimeSync> configuration (see paragraph 3.3.4), so the DS005 was left empty about the usage.

<DS005 TransfRef="Time" Enabled="true">

  <!-- We handle the time now with the IbisTimeSync above, so ignore this

    <UsedFor Table="System" Column="Time" Row="0" />

   -->

</DS005>

Figure 7‑3 Example of DS005 Usage

## DS006 Example Usage

This telegram contains information about the date. Inside the media player’s database there is a coordinate tasked to contain exactly this kind of information. So, this telegram was configured to point to that coordinate.

    <DS006 TransfRef="Date" Enabled="true">

      <UsedFor Table="System" Column="Date" Row="0" />

    </DS006>

Figure 7‑4 Example of DS005 Usage

In this case was used the coordinate: Table="System" Column="Date" Row="0"

The name of the table and the column and the number used for the row are expressed in **dictionary.xml**, so in ibis.xml we needed to match all of them.

## DS010B Example Usage

This telegram contains information about the actual index of the actual stop. This kind of information is also contained within the telegram DS021A and the specific configuration of this example, was decided to leave empty the DS010B’s usage in favor of the DS021A.

    <DS010b TransfRef="Number" Enabled="true">

      <!-- no configuration is required for this telegram since it is handled by the DS021a handler -->

    </DS010b>

Figure 7‑5 Example of DS010B Usage

## DS020 Example Usage

This telegram is the only telegram that Protran is allowed to send back to the IBIS master. It represents the Protran’s status answer and doesn’t have any meaning about strings or pictures to be shown by the media player. Upon start-up, if no route information has been received by telegrams DS021A, DS021C or GO005, then an answer of “NoData” is sent by Protran to the IBIS master. Route information is considered from persistence too before sending the answer. If persistence has a route stored, then normal answer is sent else “NoData” answer is sent back to IBIS master.

    <DS020 Enabled="true">

      <Answer>

        <DS120 Enabled="true"/>

      </Answer>

    </DS020>

Figure 7‑6 Example of DS020 Usage

## DS021A Example Usage

Also this telegram contains information about the stops that currently must be shown by the media player but the behavior of this telegram is quite different from the DS021 one, because Protran has to “collect” a specific amount of stops before sending them, and/or it has also to wait for a specific amount of time before sending them as well (a sort of timeout for a “flush”) and/or forcedly send all of them as soon as it receives a specific stop value. All this information about the telegram’s behavior can be shown with the following example:

<DS021a TransfRef="Stops021a" Enabled="true">

   <UsedFor Table="Stops" Column="StopName" Row="{0}" FromBlock="0" />

   <UsedForTransferSymbols Table="Stops" Column="StopTransferSymbols" Row="{0}" FromBlock="1" /> <!-- default FromBlock is 2 -->

   <UsedForRelativeTime Table="Stops" Column="StopTime" Row="{0}" FromBlock="2" /> <!-- default FromBlock is 3 -->

   <UsedForAbsoluteTime Table="Stops" Column="StopInfo" Row="{0}" FromBlock="2" /> <!-- default FromBlock is 3 -->

  <UsedForDestination Table="Destination" Column="DestinationName" Row="0" />

   <UsedForDestinationTransferSymbols Table="Destination" Column="DestinationTransferSymbols" Row="0" />

   <UsedForDestinationRelativeTime Table="Destination" Column="DestinationTime" Row="0" />

   <UsedForDestinationAbsoluteTime Table="Destination" Column="DestinationInfo" Row="0" />

  <UsedForText Table="PassengerMessages" Column="MessageText" Row="{0}" />

  <Connection Enabled="false">

      <UsedFor Table="Connections" Column="ConnectionDestinationName" Row="{0}" />

      <UsedForStopName Table="Connections" Column="ConnectionInfo" Row="0" />

      <UsedForLineNumber Table="Connections" Column="ConnectionLineNumber" Row="{0}" />

      <UsedForDepartureTime Table="Connections" Column="ConnectionTime" Row="{0}" />

     <LineNumberFormat>D:\Infomedia\Symbols\L{0}.png</LineNumberFormat>

      <ShowForNextStopOnly>false</ShowForNextStopOnly>

    </Connection>

    <Answer>

      <DS120 Enabled="true">

       <!-- possible Status values: Ok, NoData, MissingData, IncorrectRecord -->

       <Response Status="IncorrectRecord">6</Response>

       <DefaultResponse>0</DefaultResponse>

      </DS120>

    </Answer>

    <FlushNumberOfStations

        desc="Number of stations to collect before sending all of them to the media player.       Values admitted: a positive integer value {1; 65535}.

        Default value: 5">6</FlushNumberOfStations>

    <FlushTimeout

        desc="Amount of time to wait before forcing a send of all the current

        stations collected. Default value: 30 seconds">PT10S</FlushTimeout>

    <FirstStopIndexValue

        desc="The value of the index of the first stop.

        Values admitted {0, 1}. Default value: 0">1</FirstStopIndexValue>

    <EndingStopValue

        desc="The value of the stop that has to be considered as the end of a sequence of stops

        (and not really a proper stop as the others).

        Values admitted {99 or 999}. Default value: 99">99</EndingStopValue>

    <HideLastStop

        desc="Tells if the last stop should be shown or not.

        You might want to hide the last stop if you prefer not to show the destination

        name twice when aproaching the destination.

        Values admitted: {true, false}.

        Default is false.">false</HideLastStop>

    <HideDestinationBelow

        desc="the threshold below which the destination has to be hidden.

        Set this value to one more than the number of stops visible in your

        Perlschnur to hide the destination information (see UsedForDestination, ...)

        if the last stop is visible in the Perlschnur.">6</HideDestinationBelow>

    <ShowPastStops>true</ShowPastStops>

    <DeleteRouteIndexValue

        desc="The value of the index to delete a route.

        Values admitted {-1, 0}. Default value: -1">0</DeleteRouteIndexValue>

    <AbsoluteTimeFormat

        desc="Format of the UsedForAbsoluteTime and UsedForDestinationAbsoluteTime.

        Meaningful formats: 'HH:mm' (24-hour format), 'H:mm' (1-digit hour), 'hh:mm' (12-hour format)

        See http://msdn.microsoft.com/en-us/library/8kb3ddd4.aspx for more formats.

        Default value: 'HH:mm'">HH:mm</AbsoluteTimeFormat>

</DS021a>

Figure 7‑7 Example of DS021A Usage

The sub-tags:

* <FlushNumberOfStations>
* <FlushTimeout>
* <FirstStopIndexValue>
* <EndingStopValue>

are self-explaining.

Suppose Protran has received 3 stations with also some information about the symbols to be shown near the stops themselves. The right media player’s table must be filled as in the following example:

This information must be addressed to the Table="Stops" in the following way (again, all the columns are taken exactly from the **dictionary.xml** file):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stop City | Stop Name | Stop Arrival | Transfers | Transfer symbols |
| Biel/Bienne |  |  | Stop 1 | B;1;2 |
| Bern |  |  | Stop 2 | T;3;4 |
| Basel |  |  | Stop 3 | B;T;5;6 |
|  |  |  |  |  |
|  |  |  |  |  |

<UsedFor Table="Stops" Column="Stop city" Row="{0}" />

<UsedForTransfers Table="Stops" Column="Transfers" Row="{0}" />

<UsedForTransferSymbols Table="Stops" Column="Transfer symbols" Row="{0}" />  
Protran will calculate the right row because for that three information was used Row="{0}" (with the parenthesis).

You can notice that the telegram DS0021 is used also for the journey’s destination:

<UsedForDestination Table="Trip" Column="Destination city" Row="0" />

According to the file **dictionary.xml**, the Table="Trip"is made by only 1 row and 5 columns:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Destination city | Destination stop name | Destination time | Line | Region |
|  |  |  |  |  |

Just because this table has only one row, here was specified Row="0" (without parenthesis).

## GO003 Example Usage

This telegram contains information about all the stops that currently must be shown by the media player. It can contain in its payload one or more stations, and once Protran has parsed it, it has to send all of them in one single shot to the media player. This telegram was configured as shown below:

    <GO003 TransfRef="Stops" Enabled="true">

      <UsedFor Table="Stops" Column="Stop city" Row="{0}" />

      <Answer>

        <DS120 Enabled="true"/>

      </Answer>

    </GO003>

Figure 7‑8 Example of GO003 Usage

Attention: in the example above, it’s expressed only one line with the tag <UsedFor> but before we have told that there might be more than one station in the telegram’s payload. Suppose Protran has received 3 stations:

* Biel/Bienne
* Bern
* Basel

This information must be addressed to the Table="Stops" in the following way (all the columns are taken exactly from the **dictionary.xml** file):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stop City | Stop Name | Stop Arrival | Transfers | Transfer symbols |
| Biel/Bienne |  |  |  |  |
| Bern |  |  |  |  |
| Basel |  |  |  |  |

Using here the Row="{0}" (with the parenthesis), Protran fills automatically the row number (0 for Biel/Bienne, 1 for Bern and 2 for Basel). If would had been specified Row="0" (without the parenthesis) would had be taken only the row number 0.

As you can see, whenever Protran receives a DS021, it has also to reply to the IBIS master with a status answer.

## HPW074 Example Usage

This telegram is used to send strings with special meaning to the media player. These strings are not contained inside the IBIS telegram’s payload, but into a specific file instead (the .csv extension is mandatory and has to be encoded in UTF-8). When the media player receives these strings, it has to show them as top-most. Due to this particular property, was created a new table into the media player’s database and this telegram has to refer to a precise coordinate in it. According to **dictionary.xml**, this table is called “Infos” and is made by 2 columns and only 1 row.

Suppose that the special string is “Hello world !”, we would obtain the following results:

|  |  |
| --- | --- |
| Remote PC | Popup |
|  | Hello world ! |

and the right way to configure Protran is:

    <HPW074 TransfRef="Number" Enabled="true">

      <UsedFor Table="Infos" Column="Popup" Row="0" />

      <SpecialTextFile>C:\specialtext.csv</SpecialTextFile>

    </HPW074>

Figure 7‑9 Example of HPW074 Usage

# Persistence Service

Usually there’s the need to show updated information on the media player not only during the normal system’s execution but also after system reboots and/or software crashes. Suppose for example that on the media player are shown the stops of a specific “Perlschnur” and suppose also that the system has to be rebooted. The reboot takes some minutes and when everything is completely restarted is not guaranteed that the most updated information are immediately re-shown on media player. This is because Protran needs to collect information before sending them to the player. Take for example the IBIS telegrams DS021a about the stops and the GO002 for the connections. Following the functional specifications about those telegrams, Protran is not allowed to send to the player the information as soon as it receives them but rather it has to collect them and only after specific circumstances it can flush them to the media player. The time required to collect the information depends by the IBIS master and obviously may lead to an additional delay on showing the stops and connections to the passengers. For this reasons in Protran is active a service called “Persistence”: during its activity, Protran not only collects the information in its memory but also stores them into a file called “Persistence.xml”. Every time Protran starts, it checks about the existence of that file and if it exists, it populates its memory with the information taken from that file as they were coming from the IBIS master. In this way it can reestablish immediately the situation left before the reboot/crash.

## Validity Period

As mentioned before, Protran stores information in the Persistence.xml file and loads them every time it starts. Suppose that between the last Protran’s execution and the next one there is a pause of one night. Obviously, in this case the information stored in the Persistence.xml file is not anymore valid because too old and therefore it doesn’t make sense to show them immediately on Infomedia.

Currently the validity period is 10 minutes and it starts from the time the data is stored in the file (i.e. when the application shuts down).

## Information Saved

Protran is able to accomplish the persistence service for the information coming from the telegrams DS021a, DS021c, GO002 and GO005.

With newer versions, other information will be added.

### DS021a/c, GO005

All stop information configured in the UsedFor (see chapters 0 and 0) parameters as well as the current stop index (from DS010, DS010b or DS010j respectively) are stored. Therefore Protran also remembers whether it has missing records and reports it with the system status MissingData or Ok.

### GO002

All GO002 telegrams are stored and reloaded when the application restarts. Protran does not automatically send any data, but waits until receiving the first DS010b after the restart. Protran assumes Infomedia was restarted at the same time and thus doesn’t clear any information on the connection screen.

### FTP File List for Abu Dhabi

For the AbuDhabi project, all the files downloaded from the specific FTP server are stored in the Persistence.xml file. This is to avoid downloading multiple times a file already downloaded in the past.

# Arriva departures

The connections for a stop for the customer Arriva is sent to Protran via an FTP client. Protran receives an xml file “departures.xml” with all the information of various connections for a stop. The validity of the file is dependent on the device ID and the expiration time of the file. If the file is not intended for the computer, it is discarded. If the time of arrival of file is past the expiration time of the connections, then the file is discarded. The departures.xml is processed and the connections are sent to the player as a ximple.

# Appendix A: Configuration migration

## Migration from Protran 1.x to 2.x

The configuration file formats of Protran have changed for Protran 2.x. The new configuration file format is not compatible with the old one and Protran 2.x can’t read Protran 1.x configuration files.

When you upgrade an existing project, please make sure you update the configuration files according to the guidelines in this chapter.

### cfg.xml to protran.xml

The general Protran configuration has been renamed from cfg.xml to protran.xml and has been simplified.

The main tag should now contain the version of the config file structure (currently “2.0.0”):

<Protran Version="2.0.0">

The “Protocols” section can be stripped down by simply placing a single tag for each used protocol. See chapter 3.2.1.1 for the correct syntax.

Please keep in mind that you will also have to enable the “IOProtocol” if you are using the “Stop Requested” (or “Special Input”) digital I/Os of the old Gorba Topboxes (PM600, PM800 and Atom).

The <Persistence> section stays the same, except for the default validity which is now configured as an XML duration (using the “PTxx” syntax).

### ibis.xml

The IBIS protocol configuration underwent also a rather big refactoring to make it better understandable. Please verify your ibis.xml against the latest ibis.xsd to make sure your file structure is up-to-date.

* The <IBIS><Behaviour> was moved to the beginning of the file since it is valid for all kinds of IBIS sources
  + The <IBIS><Behaviour><SpeakersVolume> is no more needed since the speaker and volume is directly controlled by the renderer
* All sources were moved under <IBIS><Sources>:
  + <IBIS><SerialPorts> is moved inside <IBIS><Sources> and only one serial port is allowed. The second serial port was used for I/O only and is now replaced by the I/O protocol, see below. For the same reason the <SerialPort><CtsUsedFor> and <SerialPort><UsedForIbis> tags are no more needed.
  + <IBIS><Simulation> is moved inside <IBIS><Sources> and the <Simulation><Active> was removed (replaced by <IBIS><Sources Active="…">)
  + <IBIS><UdpServer> is moved inside <IBIS><Sources>
* <IBIS><RecordingConfig> is renamed to <IBIS><Recording>
  + The elements inside <IBIS><RecordingConfig><IbisRecordingConfig> are now directly in <IBIS><Recording>
  + The flag <ForGismo> is replaced by the <Format> element which allows future extension with other file formats (currently only “Protran” is supported)
* The elements inside <IBIS><TimeSync><IbisTimeSync> are now directly in <IBIS><TimeSync>, the Enabled="…" attribute is moved to the parent as well
* All <…UsedFor> and <UsedFor…> elements in the entire configuration now support the Language="…" attribute to select the dictionary language (in addition to table, column and row) – default value is 0.
* All durations are now configured as an XML duration (using the “PTxx” syntax):
  + <IBIS><Behaviour><ConnectionTimeOut>
  + <IBIS><Simulation><InitialDelay>
  + <IBIS><Simulation><IntervalBetweenTelegrams>
  + <IBIS><TimeSync><InitialDelay>
  + <IBIS><TimeSync><Tolerance>
  + <IBIS><Telegrams><DS021a><FlushTimeout>
  + <IBIS><Telegrams><DS021c><FlushTimeout>
  + <IBIS><Telegrams><GO005><FlushTimeout>
* The special handling for DS001a has been removed since code conversion is now handled in Infomedia.
  + <IBIS><Telegrams><DS001a><UsedForText> is no longer supported
  + <IBIS><Telegrams><DS001a><CodeConversionFile> is no longer supported
  + Because DS001 is no more implicitly handled together with DS001a, the tag <IBIS><Telegrams><DS001> should now contain again a <UsedFor>

### arriva.xml

The following elements are removed since they were obsolete:

* The <Arriva><ComputerName> is now taken from the system automatically
* The <Arriva><FTP><PollTime> is no more used since the file changes are not observed by polling but rather by listening directly to changes in the file

A new section <OBU> was added to replace the remote server configuration previously available in cfg.xml (“ArrivaProtocol” section). The following tags must be configured with the information that was in cfg.xml:

  <OBU Descr="Container of all information useful to connect to Arriva OBU.">

    <RemoteIP Descr="IP address of the OBU">192.168.0.1</RemoteIP>

    <RemotePort Descr="TCP port on the OBU">35001</RemotePort>

  </OBU>

Figure 10‑1 New OBU section in arriva.xml

### AbuDhabi.xml

Since I/Os are now handled by the I/O protocol, the following configuration parameters are no more needed:

* <AbuDhabi><IBIS><CtsUsedFor> is now replaced by using the local I/O called “StopRequest”
* <AbuDhabi><IBIS><SpecialInputUsedFor> is now replaced by using the local I/O called “SpecialInput”

The parameter <AbuDhabi><ISM><Behaviour><PollTime> is now configured as an XML duration (using the “PTxx” syntax).

All xxxUsedFor elements now support the Language="…" attribute to select the dictionary language (in addition to table, column and row)

### New io.xml

The new I/O protocol allows to configure I/Os independent of the communication protocol used (e.g. Arriva protocol + stop request using the digital I/Os of the old Gorba Topboxes (PM600, PM800 and Atom).

See above for more information about the migration to this protocol and its configuration.