Using 4.8.1 Datalink

Arno Rouschen
Blaise Team
IBUC 2009 - Riga, Latvia

Overview

- Datalink Concepts
- Generic BOI files
- Using Data Centre
  - Playing with data; Exploring features
- BOI 4.8.1 Data Storage
  - Stream versus table data
- BOI files and Data Servers
  - Pitfalls and recommendations
- Versioning and migration

Blaise Datalink Concepts

- Based on Microsoft OLE DB technology
- Blaise OLE DB Interface file
  - Data file within Blaise
  - Defines link to data that is stored elsewhere
    - Oracle, SQL Server, MySQL, MS Access ...
  - Can be used with all Blaise tools, like DEP, Blaise Internet, Manipula, Data Viewer

Blaise Datalink History

- Blaise 4.5 / BCP 1 (2001)
  - Existing table
- Blaise 4.6 / BCP 2 (2003)
  - Existing table / based on Blaise data model
- Blaise 4.7 Enterprise (2005)
  - Customizable BOI files, select statements, referencing other data files
- Blaise 4.8 Enterprise
  - 4.8.0 (2007): JOINKEY support, Data Server
  - 4.8.1 (2008): Generic storage, versioning
  - 4.8.2 (2009): Bulk insert, synchronization

Blaise OLE DB Interface File

- Data file type available in Blaise
  - Extension .BOI
- Contains logical definitions; no data
  - Connection information
  - Database tables to be accessed
  - Field mappings to be used
  - Runtime settings and more
- Has an associated Blaise meta file (.bmi)

Blaise OLE DB Tools

- Blaise OLE DB Toolbox (since 4.6)
  - Blaise OLE DB Mapping Wizard
  - Blaise OLE DB Interface Create Wizard
  - Blaise OLE DB Tables Create Wizard
- Blaise OLE DB Command Builder (4.6)
- Blaise OLE DB Workshop (4.7)

- Blaise Data Centre (4.8.1)

BOI file types

- Simple (data only) BOI file
  - Contains just one table definition
  - Can be used for lookups or exports
- Full BOI file
  - Created based on Blaise data model
  - BDB equivalent; stores all data, including status, remarks, error information, etc.
  - Used to store questionnaire data in external db
- BOI referencing another data file
  - Links to BDB, another BOI or an ASCII file
  - Used within Blaise Internet for security reasons

Demo 1: Creating a BOI file for an existing database table

Data Partition Types

- Determines structure how Blaise data will be stored in the OLE DB data source
  - Flat, no blocks
  - Flat, blocks; table for each block type
  - In depth; field, status, value
  - In depth text; field; status; value as text
  - Stream; stream of data per record

Demo 2: Creating a BOI file for a Blaise data model

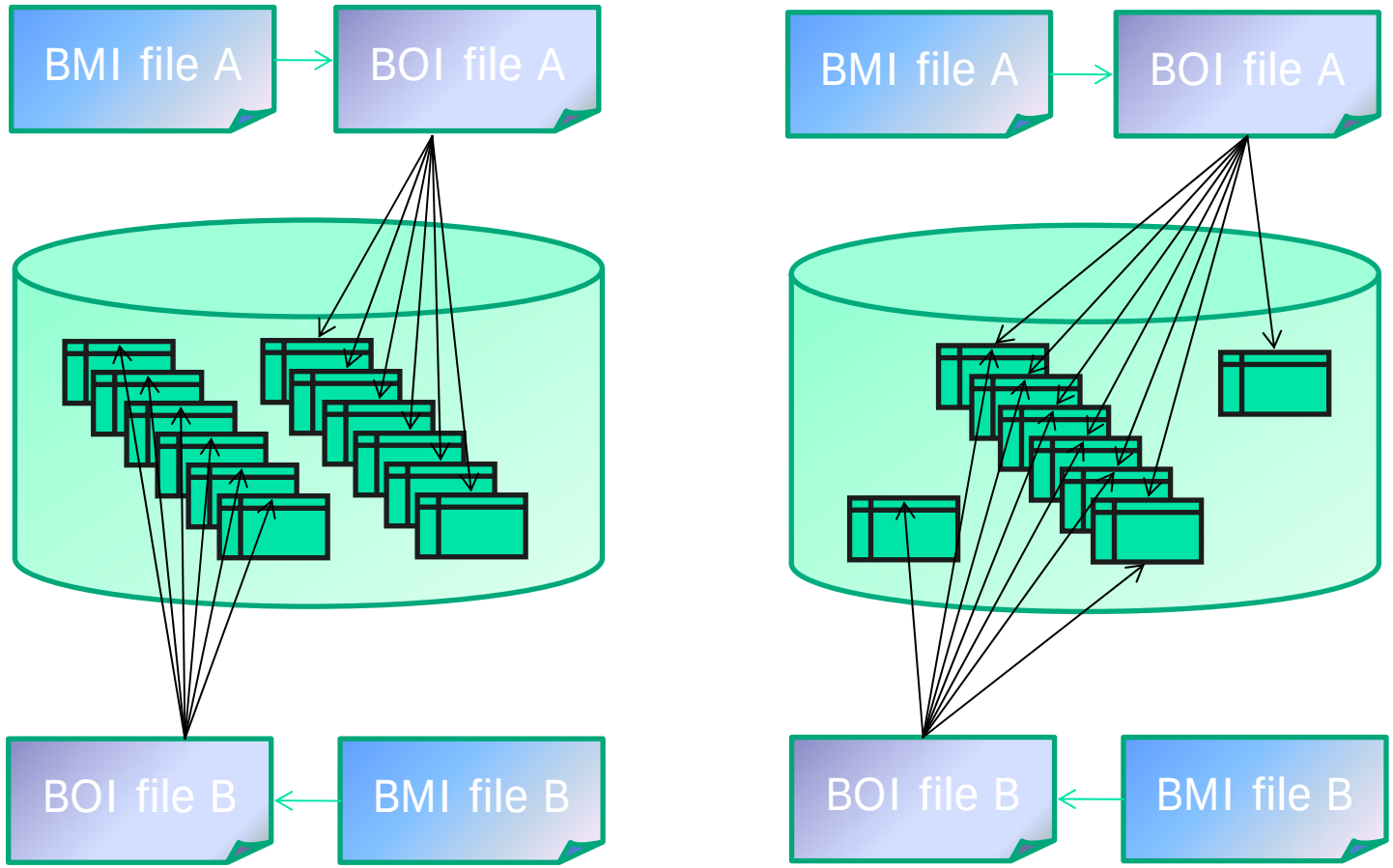
Generic BOI files (4.8.1)

- Goals
  - Store questionnaire data of multiple surveys in a centralized input data store
  - Share database tables as much as possible between surveys
  - Less tables in database; reduce administration burden
  - Generic table structures; uniform data access
  - Provide an easy interface to existing in house systems

Generic BOI files – Non-generic vs Generic

- Non-generic
  - Tailor made for data model
  - Each data model has own set of tables
  - Database tables are not reusable
  - Optimized table and column structure
  - Primary key of data model
- Generic
  - Reuse database tables as much as possible
  - Have fixed predefined tables and columns
  - Common primary key
  - Support for meta and data versioning

Generic BOI files – Table Access



Non-generic

Generic

Generic BOI files – Tables and Data Partition Types

BOI Table Type	Flat	In Depth	Stream
Dictionary Information	☑	☑	☑
Case Information	☑	☑	☑
Form Information	☑	☑	☑
Key Information	☑	☑	☑
ID Information	☑	☑	☑
Remarks	☑	☑	
Open questions	☑	☑	
Flat data (Non-Generic)	☑		

Generic BOI files – Concepts

- Common primary key in all tables
  - JOINKEY
  - DMKEY
  - BEGINSTAMP
- Fixed predefined table structures
  - Column widths are set to maximum
  - Tables contain all available columns

Generic BOI files – Versioning

- Meta and data versioning
- Data versioning occurs on the record level
- Versioning columns
  - BEGINSTAMP
  - ENDSTAMP
    - Date time column type in database
    - Fixed string in Blaise
    - ‘YYYYMMDD HH:MM:SS’

Generic BOI files – Record states

- Actual record
  - ENDSTAMP filled with ‘99991231 00:00:00’
  - Updateable
- Historical record
  - ENDSTAMP filled with ‘real’ timestamp
  - Not updateable

Blaise Data Centre

- Provides uniform access to Blaise Data Files
  - .BDB, .BOI and ‘ASCII referenced by BOI’
  - View survey progress
  - View, extract and analyze data
- Administration (Admin mode)
  - Registration of Dictionaries
  - Simple Case Management
  - Distribution and deployment of cases
  - Synchronization of cases

Demo 3: Using Blaise Data Centre

Blaise 4.8.1 Storage

- BOI 4.8.1 Data is stored redundantly
  1. In the regular data tables
  2. As a stream per record in Form table
- Advantages
  - Optimal integration with Blaise system
  - Record streams can be delivered directly to the Blaise system without any translation

Blaise 4.8.1 Storage

- Disadvantages
  - Changes directly made to the data tables, outside Blaise, are not visible ...
  - As a result data in record stream and table data may become inconsistent
- Recommendation
  - Don’t apply changes, outside Blaise, to the data tables directly
  - But, for those who can’t resist the temptation...

Blaise 4.8.1 Storage

- ... we have invented the Fast-Reading BOI runtime setting
  - If True: record data is retrieved via record streams
  - If False: record data is retrieved the regular way; i.e. from the data in the tables
- Works only for record oriented clients like DEP, Manipula (and not for Data Viewer)

Demo 4: Using the Fast-Reading setting

Blaise 4.8.1 Storage

- How to make data consistent again?
  - Run a Manipula setup with a boi updatefile
    - Set Fast-Reading in the boi to true if you want to make the table data consistent with the streams
    - Set Fast-Reading in the boi to false if you want to make the streams consistent with the current table data
- 4.8.2 will have synchronize options

Blaise Data Server - Intro

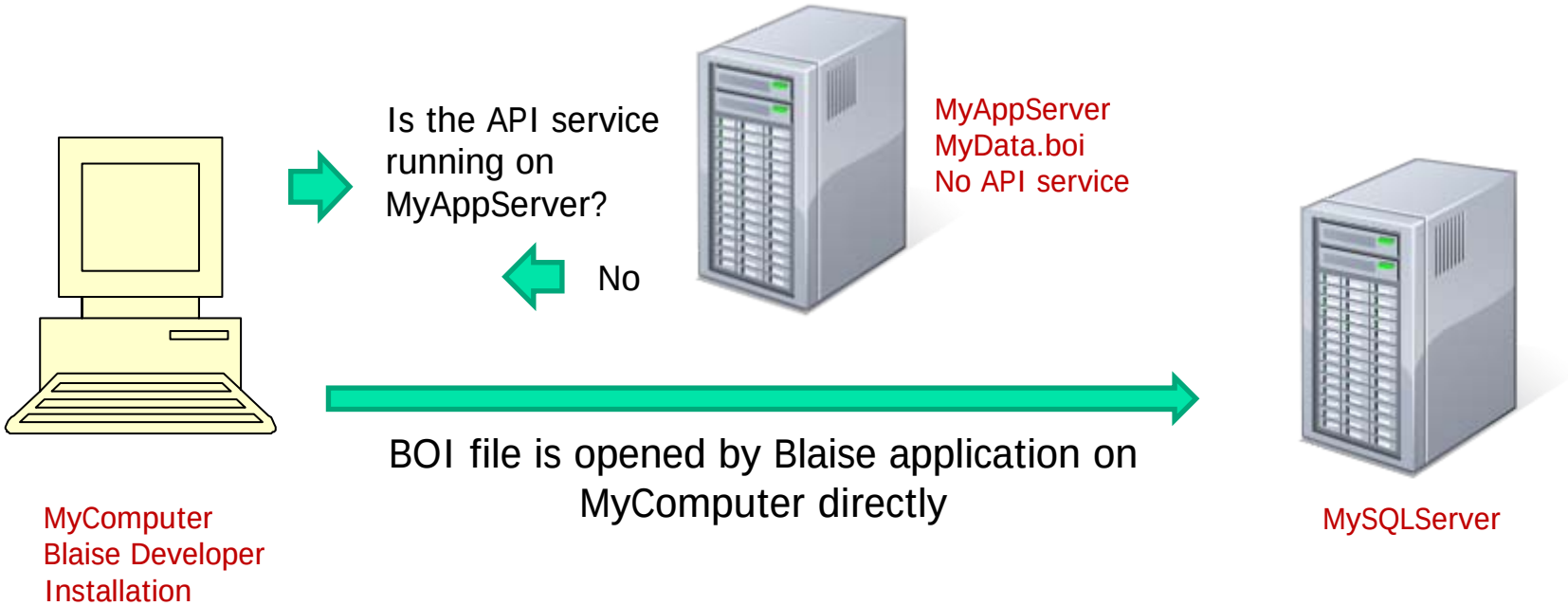
- Computer that runs the Blaise 4.8 API service
- Clients communicate with the API service through XML messages over TCP/IP
- Developed for Blaise Internet, but can be used without Blaise Internet also
  - API service has exclusive access to data files
  - Enables CATI with BOI files
  - Possibility to create a secure environment

Blaise Data Server – Connecting

- Location of BDB/BOI file determines who is going to connect to a database
  - What you probably expect:
    - Connection to the database is made by a Blaise application directly
  - What you might not expect:
    - If the BDB/BOI is on a computer which is running the Blaise API service, then the Blaise API service on that machine will try to connect to the database

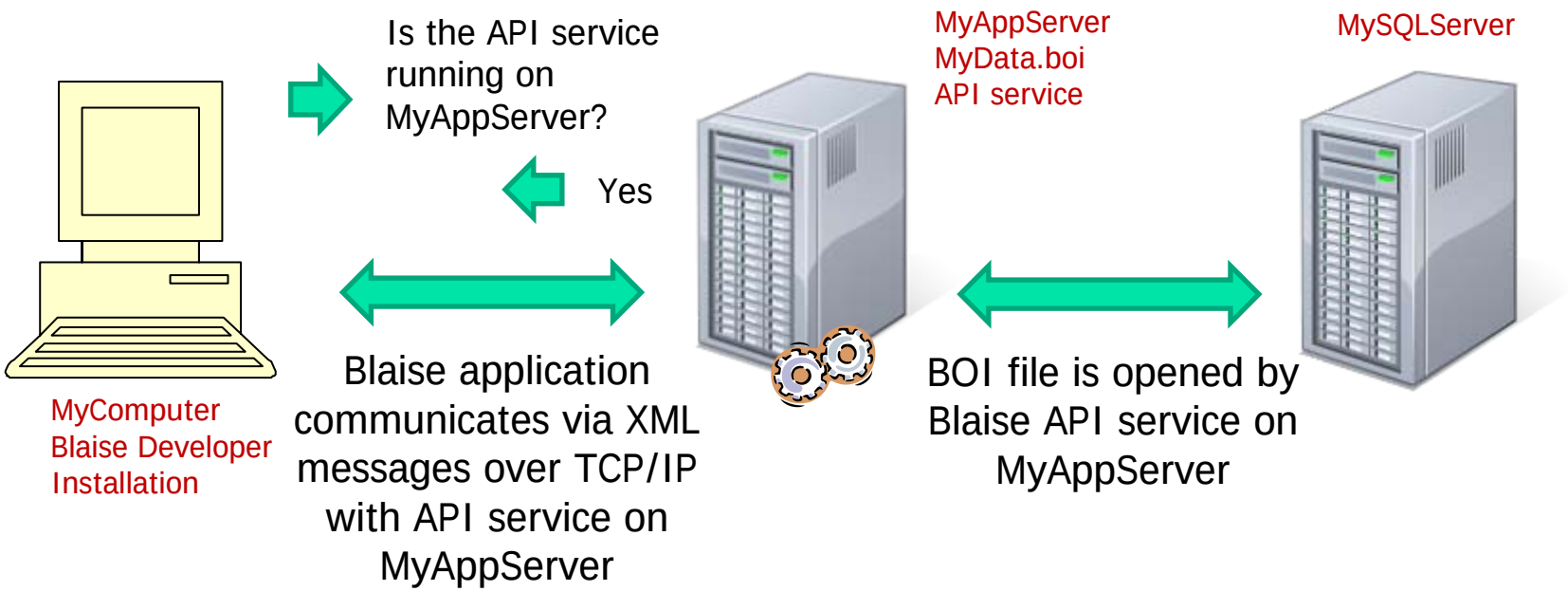
Blaise Data Server – Connecting without Service

- MyData.BOI is stored on MyAppServer
- BOI contains a connection to MySQLServer
- Blaise API service is not running on MyAppServer



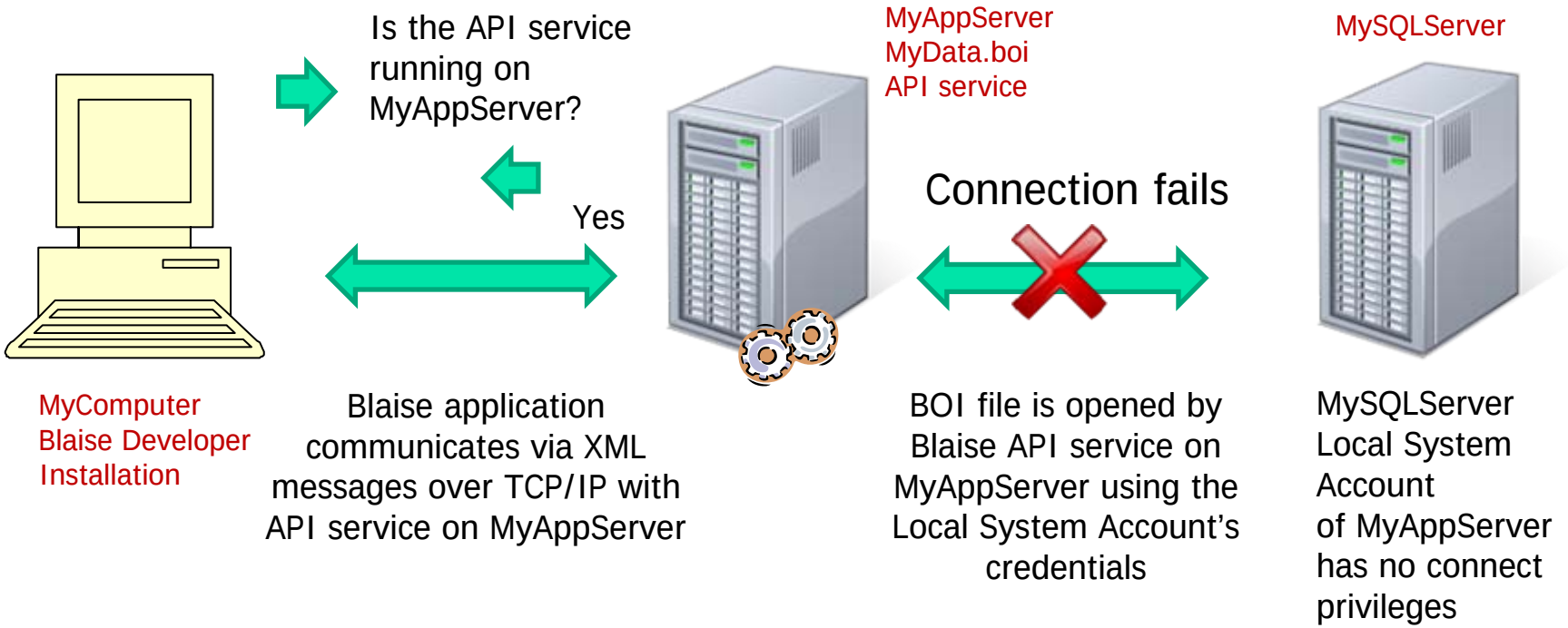
Blaise Data Server – Connecting through the Service

- MyData.BOI is stored on MyAppServer
- BOI contains a connection to MySQLServer
- Blaise API service is running on MyAppServer



Blaise Data Server –
Pitfall 1: Using Windows Authentication

- MyData.BOI is stored on MyAppServer
- BOI connects MySQLServer using Windows Authentication

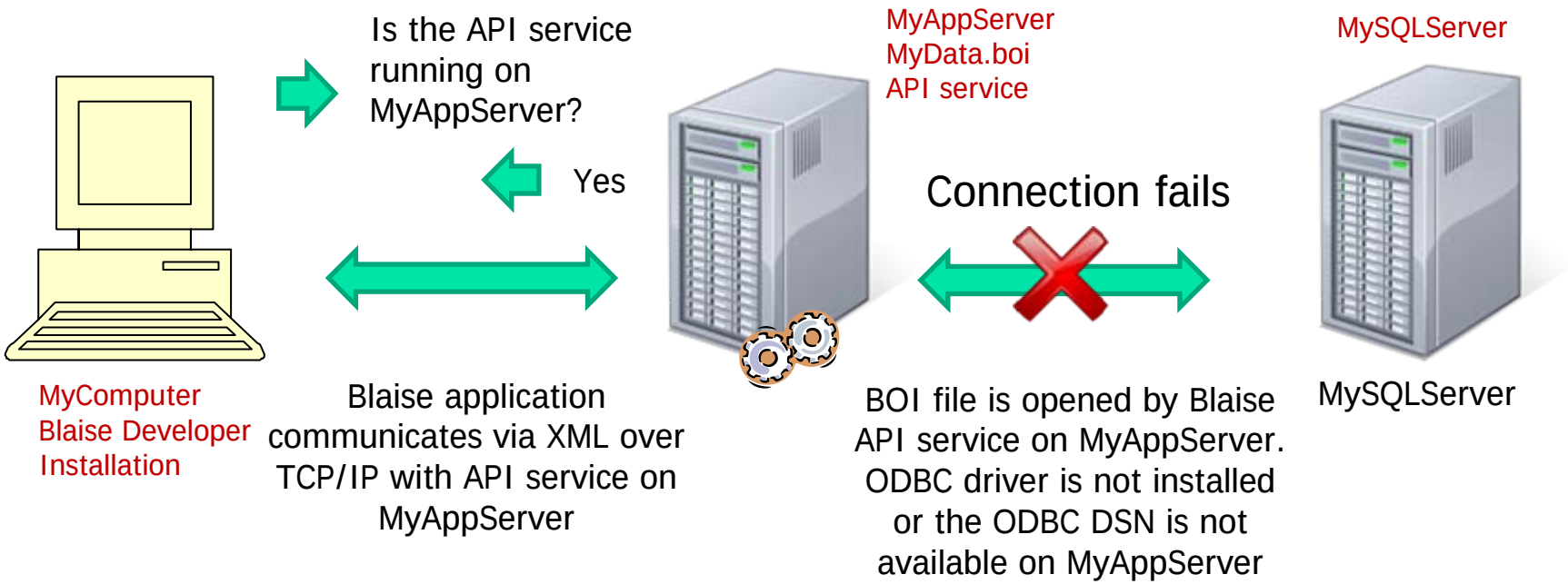


Solutions Pitfall 1

- Use database user account instead of Windows authentication in BOI connection string (preferable)
- Change the account of the API service to an account which has enough rights to access the database

Blaise Data Server –
Pitfall 2:Missing drivers / settings

- MyData.BOI is stored on MyAppServer
- BOI connects to MySQLServer by using an ODBC connection string



Solution Pitfall 2

- Install all necessary drivers / OLE DB providers / Database client software on the Blaise Data Server
- Be sure that all the needed settings, connection parameters, ODBC DSN can be resolved from the Data Server
- Connection string in BOI file has to be valid in the eyes of computer that runs the Blaise API service (and not from your computer)

Versioning and migration - Migration to a newer version of a data model

1. Always create a new BOI file based on the new data model.
   - There is no check during the boi generation process whether tables already exist
   - If the BOI contains already existing table names then rename those tables in order to avoid conflicts
2. If all conflicts have been solved, then create the tables and perform an old BOI to new BOI copy with help from Manipula

Versioning and migration - Meta versioning in Generic BOI files

- Datalink looks up the DMKey whenever a generic BOI file is opened
- If the data model checksum has been changed then a new DMKey is generated.
  - Data collected with the new model is stored with this new key.
  - Data collected with the previous version of the model remains untouched

Versioning and migration – Using generic BOI files

- No need to recreate any table in case of in depth and stream data partition types
- Data will automatically be stored under another DMKey
- Only flat data tables must be recreated. Rename flat table names in new boi file to non-existing table names.

Demo 5: Migrating to a newer version of a data model