INNES

User manual

Briva LDAP Server

May 2016
Manual version 003A

Table of contents

1 Introduction

Briva LDAP G3 is a product developed by INNES.

It permits to offer LDAP service without installing Active Directory or any other integrated LDAP systems.

Version 3.10.12 includes 2 connectors, which represents the source file format:

- Til-technologies : csv and xls versions
- Belledonne : xls version

2 Architecture

Briva LDAP supports user&phone directory stored in

- CSV format file or
- XLS format file

Note: for other customers, building your own plugin (JS) and your own data base is proposed in appendix in case CSV format XLS format (rows and columns had to change). Or Innes can do an other plugin to support an other datasource format.

The server Briva LDAP can manage only one format at a time. So choose the format according to your needs.

2.1 Listener

The server includes a listener of the source file (CSV or XLS).

When the source file is updated (or replaced), a new repository will automatically be created (the new repository replaces the current repository).

Note: please do save the current source file before update with a new one in order to have the possibility to return back to the current repository.

3 Installation

3.1 Executable

In order to install “Briva LDAP server” run the executable:

briva_ldap_server-setup-3.10.12.exe

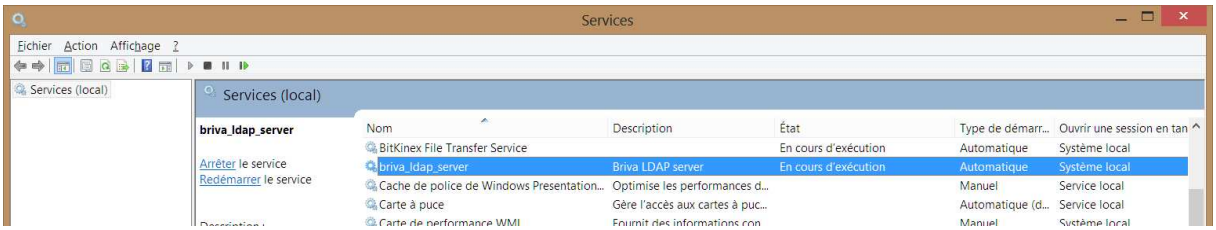
3.2 Default directory

The (default) installation directory is:

C:\Program Files (x86)\Innes briva ldap server

3.3 Windows services manager

The server starts immediately after installation (service manager => “briva_LDAP_server”).



Note: If briva_ldap_server is not present in services list, consult log files in the following directory INSTALL DIRECTORY\daemon.

Note: If briva_ldap_server is not running in services list, please right click on service briva_ldap_server and execute.

3.4 Uninstall

In order to uninstall, launch:

briva_ldap_server-uninstall.exe

4 Configuration

The Briva server can works with the configuration by default.

4.1 Configuration file

Two files are permitting to configure Briva LDAP server

- Main configuration file permitting to choose technology (CSV, XLS, …)
  - **[installation directory]/**configuration.json
- Secondary configuration file in each plugin config.json permitting to make data mapping and transformation from CSV or XLS file to server
  - **[installation directory]/plugins**/til-technologies-csv/config.json*
    An example is provided in appendix
  - **[installation directory]/plugins**/til-technologies-excel/config.json
  - **[installation directory]/plugins**/belledonne-excel/config.json

4.2 Configuration file "configuration.json"

The server parameters are contained in file "configuration.json". This file needs to be modified according to your needs.

Default configuration:

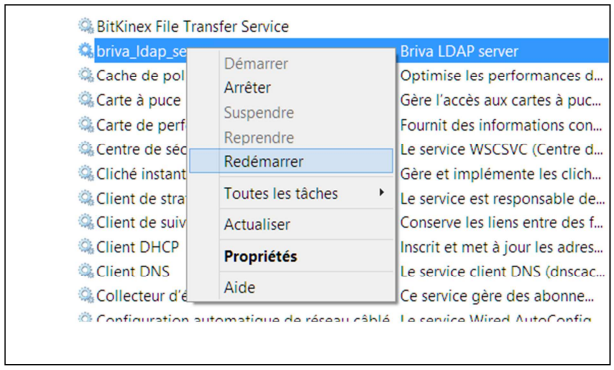
```
{
  "name": "briva_ldap_server",
  "version": "1.10.10",
  "description": "Briva LDAP server",
  "connector_type" : "ldap",
  "logger" : "none",
  "connector_plugin" : "til-technologies-csv",
  "directoryFile" : "C:/liste controle d acces.csv",
  "basedn" : "dc=innes, dc=pro",
  "ldap_port" : 389
}
```

This JSON file can be modified according to your needs.

Attention: JSON is a case sensitive. Take care to respect the format above.

Be careful to use / character instead of \ for the "directoryFile".

key	Description	Default value
name	Name which is present in LDAP request	briva_ldap_server
version	Version which is present in LDAP request	3.10.12
description	Label	Briva LDAP server
connector_type	Connector type	ldap
connector_plugin	Defines which plugin is used to create the repository: Possible values: • **"til-technologies-csv"** • **"til-technologies-excel"** • **"belledonne-excel"**	til-technologies-csv
Logger	• **"console"** output to Node js console, only applicable if the server is being run manually • **"event"** writes messages to the Windows event log under the application section • **"web"** stores all messages in order they can be consulted from a Web page (RFU) • **"file"** writes messages to a text file briva-server.log in the application directory Any other value will run the server in silent mode. By default this is set to **none** which mean no logging is done	none
directoryFile	Full path to the XLS (or CSV) local file Ex: C:\tmp_data_user_and_phone\ MS_BADGE-V2.XLS Ex: C:\tmp_data_user_and_phone\liste controle d acces.csv	D:\briva\briva-server\MS_BADGE-V2.XLS
basedn	Provides the base domain name for the repository	dc=innes, dc=pro
ldap_port	The port that will used to listen for LDAP calls	389



Note: In case modification of a configuration file, do restart service

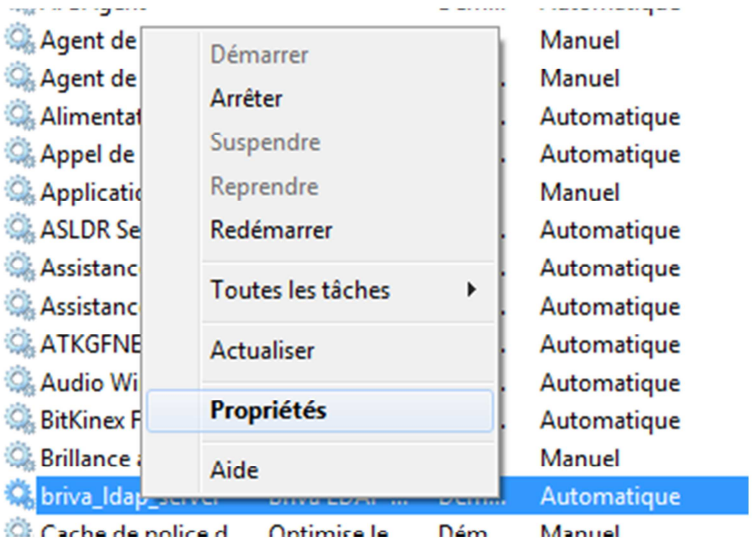
Important note if “directoryFile” is on a network drive:

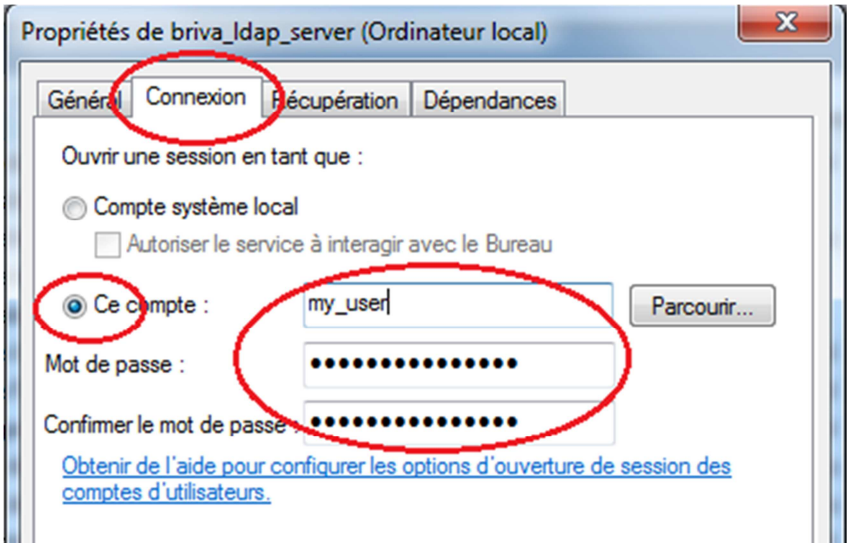
Example: We want to use the datasource on a network drive “Y:\temp\datasource.xls”

You have to replace the drive letter “Y:” by its real urn:

“\\innes-si01\temp\datasource.xls”

Because briva ldap is installed as a service, it doesn’t have the right to access such urn, so manually configure the service to gives it the right:





And then, restart the service.

4.3 Configuration file config.json

In a plugin directory, the **config.json*** defines:

- Some specific parameters,
- The data mapping and
- The transformation parameters

An example of config.json file is given in appendix

4.3.1 Data mapping

"DataMapping" defines how the source file will be loaded to the entity attributes.

4.3.1.1 Assumptions for Excel or CSV format

- In Til-technologies CSV or XLS source format, it is assumed that the first row has to contain the field name for each column (column title)s
- In CSV,
  - the value of the separator can be modified (by default this is a semi colon character ";")
  - By default, "Numéro de badge" of CSV file column tile is mapped to "badgeID" LDAP attribute

```
{"srcName" : "Numéro de badge",
 "dstName" : "badgeID",
```

- For XLS,
  - the sheet name (sheetName) that contains the repository source data need to be defined
  - By default, "BadgeID" of XLS file column tile is mapped to "badgeID" LDAP attribute

```
{"srcName" : "BadgeID",
 "dstName" : "badgeID",
```

- For belledonne XLS, in the dataMapping, you can use the format "extractNumber". This is used for a cell with type string, and the following action is done:
    - look for the first occurrence of a colon :
    - remove white space after until get to a numeric character and/or hexadecimal character
    - extract all numeric/hexadecimal characters (from current position) until get to a non-numeric/hexadecimal character or the end of the string.

  Example: “Numéro : 04A2E8FA8C3280, Numéro : 05 - Wiegand 37 bits TELEM” will extract “04A2E8FA8C3280”.

4.3.2 Data transformation

“DataTransformation” defines how additional attributes will be transformed from existing attributes.

Key	values
DataMapping	Defines how each column in the source file will be mapped to the attribute entities. For each column the following data must be defined **srcName** : name of the column in the source file **dstName** : attribute name **mandatory** : if the column is mandatory or not **format** : the format of the data (string, date, extractNumber) that is used to invoke a data conversion routine **default** : RFU
dataTransformation	**srcAttributes** : an array of source attributes **dstAttribute** : the name of the new attribute to be created **separator** : used to separate attributes if multiple source attributes are specified

5 Debug & console mode

In order to launch Briva LDAP in console mode,

- Stop service
- Set configuration.json in console mode

```
{
  "name": "briva_ldap_server",
  "version": "1.10.10",
  "description": "Briva LDAP server",
  "connector_type" : "ldap",
  "logger" : "console",
  "connector_plugin" : "til-technologies-csv",
  "directoryFile" : "C:/liste controle d acces.csv",
  "basedn" : "dc=innes, dc=pro",
  "ldap_port" : 389
}
```

- Launch Briva LDAP in manual mode with command prompt

```
C:\Programmes Files <x86>\Innes briva ldap server\node
briva_ldap_server.js
```

6 Appendix

6.1 Example of configuration file config.json (CSV)

```
{
  "separator":";",

 "dataMapping" :
  [
        {"srcName" : "Prénom",
     "dstName" : "givenName",
        "mandatory" : true,
        "format":"string"
        },

    {"srcName" : "Nom",
     "dstName" : "sn",
        "mandatory" : true,
        "format":"string"
        },
        {"srcName" : "Numéro de badge",
      "dstName" : "badgeID",
        "mandatory" : true,
        "format":"string"
        },
    {"srcName" : "Fonction",
     "dstName" : "organizationalRole",
        "mandatory" : false,
        "format":"string"
        },
    {"srcName" : "Service",
     "dstName" : "ou",
        "mandatory" : false,
        "format":"string"
        },
    {"srcName" : "Entreprise",
     "dstName" : "o",
        "mandatory" : false,
        "format":"string"
        },
    {"srcName" : "Téléphone",
     "dstName" : "telephoneNumber",
        "mandatory" : true,
        "format":"string"
        },
    {"srcName" : "Email",
     "dstName" : "mail",
        "mandatory" : true,
        "format":"string"
        },
     {"srcName" : "Début validité",
     "dstName" : "startValidDate",
        "mandatory" : false,
        "format":"date"
        },
    {"srcName" : "Fin validité",
     "dstName" : "endValidDate",
        "mandatory" : false,
        "format":"date"
        }
 ],

 "dataTransformation" :
   [
     {
       "srcAttributes" : ["givenName","sn"],
          "dstAttribute" :  "cn",
          "separator" : " "
       },
     {
       "srcAttributes" : ["cn"],
          "dstAttribute" :  "name",
          "separator" : " "
       }
   ]
 }
```

6.2 Example of plugin JS file (Til-technologies-csv.js)

```
var fs = require('fs');   // for watch file
var parse = require('csv-parse');

var dir = require('../../directory.js');
var logger = require('../../logger.js');
var config = require('../../configuration.json');

var editionConfig = require('./config.json');

// Attention Case Sensitive definition
var dataMapping = editionConfig.dataMapping;

// Order is important in this configuration
var dataTransformation =  editionConfig.dataTransformation;

var callBack;
 //=====================================================================
 function buildRepositoryFromCSV(err, data)
 {
 if (err !== null)
    {
     logger.log("File error state : " + err.toString() .bold.red);
        }
  else
   {
                var i,j;
                var curContact

                dir.userarray = [];

                if (data !== undefined)
                {

              // get array of lines
                        for(i=0;i<data.length;i++)
                               {
                                       curContact = {};
                        isComplete = true;

                                       for (j=0;j<dataMapping.length;j++)
                                       {
                                               curContact[dataMapping[j].dstName] ="";

                                               var  fieldData  =
data[i][dataMapping[j].srcName];

                                               if  ((fieldData == undefined) ||
(fieldData == ""))
                                                {
                                                  if (dataMapping[j].mandatory)
                                                        {
                                                         logger.log("Rejected entry :
row " + i  + " no data for "+ dataMapping[j].srcName);
                                                                isComplete = false;
                                                                break;}
                                                   }
                                               else
                                                   {
                                                        //logger.log(i + "  : " +
getCellData(cell,dataMapping[j].format));

curContact[dataMapping[j].dstName] = fieldData;
                                                  }
                                       }

                                               if (isComplete)
                                                   {
                                                                                       // Copy
attributes to other attributes
                                                                                       for
(j=0;j<dataTransformation.length;j++)
                                                                                         {
```

```
        curContact[dataTransformation[j].dstAttribute] = "";
                                                                                for
(k=0;k<dataTransformation[j].srcAttributes.length;k++)
                                                                                {

if (k > 0)   curContact[dataTransformation[j].dstAttribute] +=
dataTransformation[j].separator;

curContact[dataTransformation[j].dstAttribute] +=
curContact[dataTransformation[j].srcAttributes[k]]
                                                                                }
                                                                            }


        newEntity = {};


        newEntity.dn = "cn=" + curContact.cn + ", " + config.basedn;


        newEntity.attributes = {};

        newEntity.attributes.objectClass = [ "user" ];

                                                                                for
(var attribute in curContact)
                                                                                {

newEntity.attributes[attribute] = curContact[attribute];
                                                                                }


        dir.userarray.push(newEntity);

                                                        }

                                }
                }

                if (callBack)
                    {
                          callBack();
                             callBack = null;
                        }

                logger.log("Loaded new repository (entries = " +
dir.userarray.length.toString()  + ")");
    }
 }

 function readCSVFile()
{

        // deals with open/close automatically
        fs.readFile(config.directoryFile, "binary", function (err, data) {
                          if (err !== null)
                                {
                                 logger.log("File error state : " + err.toString());
                                 }
                          else
                           {
                                 var params = {};
                                 params.delimiter = editionConfig.separator;
                                 params.columns = true;

                                 parse(data, params ,buildRepositoryFromCSV);
                                }
        });
}



//=========================================================================
function initFileMonitor()
        {
```

```
                try{

                        // detects direct edit and copy paste
                        // Default : { persistent: true, interval: 5007 }
                        fs.watchFile(config.directoryFile, function (curr, prev) {
                             readCSVFile();
                              });
                }
                catch(ex)
                {logger.error("ERROR " + ex);}
        }


 function initialize(callback)
 {
    callBack = callback;

        initFileMonitor();

    readCSVFile();


 }


  module.exports = {

    initialize: initialize

};
```

6.2.1 Building a custom plugin "custom plugin"

In **[installation directory]/configuration.json**, define the new value of connector_plugin

Ex: "connector_plugin" : "custom_plugin"

(instead of "connector_plugin" : " "til-technologies-csv").

Under the plugin directory [installation directory]/plugins, create a new directory "custom_plugin"

[installation directory]/plugins/custom_plugin/.

6.2.1.1 plugin javascript pattern

In the directory custom_plugin create a javascript file named

custom_plugin.js

6.2.1.1.1 Export a function "initialize"

This function must do two actions:

- Fill the repository with the required data
- Start any procedures for updating the repository

6.2.1.1.2 Repository made up of entities

Each entity must contain a DN (distinguished name) and a list of attributes

For example

```
newEntity = {};
newEntity.dn = "cn=John Smith, dc=Innes, dc=Pro";
```

```
newEntity.attributes = {};
newEntity.attributes.objectClass = [ "user" ];
newEntity.attributes.ou = "sales and marketing";
```

The list of attributes is defined depending on

- the source data and
- the business requirements for the LDAP server.

Do follow existing attributes names

For example:

```
sn = Surname
```

Each entity then must be added to the core repository

```
repository.userarray.push(newEntity);
```

In order to access the repository, add this line

```
var repository = require('../../directory.js');
```

6.2.1.1.3 Logger activation

```
var logger = require('../../logger.js');
var config = require('../../configuration.json');
```