
DrawerBackup Documentation

Release 0.0.1

mInternauta

September 21, 2016

1	What is DrawerBackup?	1
2	Who needs DrawerBackup?	3
3	Documentation	5
3.1	Architecture & Services	5
3.2	Terminology	6
3.3	Configuration	9
3.4	Features List	9
3.5	Server/Storage Installation	10
3.6	Client Installation	11
3.7	WebUI Installation	13
3.8	Quick Start	14
3.9	Jobs	21
3.10	Schedules	22
3.11	Operations	22
3.12	Clients	23
3.13	Storages	23
3.14	Repositories	24
3.15	Customizable database support	24
3.16	Built-in Jobs for the Client	24
3.17	Advanced Filters for the Indexer	25
3.18	Auto-Discovery	26
3.19	Server Auto-Discover Script	26
3.20	Purging a Operation for the Server/Storage	27
3.21	Auto-cleaning up the Old Operations (Backups)	27
3.22	Scripts	27
3.23	Modules	28
3.24	Configurations Files	29
3.25	Configuring the MySQL	34
3.26	Logs and Output	34
3.27	Client And Server Configuration Via CLI	35
3.28	Entity Editor in the CLI	35
3.29	Tray Monitor for the Client	36
3.30	Performance Variables	36
3.31	WebS API	36
3.32	Developer Environment	37

What is DrawerBackup?

It is a set of computer services and tools to backup and recovery important data. Allowing a quick and easy storage of important information and rapid restoration of them. It's a computer network-based architecture allowing backup/restore begin done via any type of network, even over the internet. It is also modular and adapts to any environment quickly and easily. Allowing perform backups of small networks and large networks.

**FAST**

With fast indexing algorithm and fast transfer of backup entries. The client can perform quickly back up data.

**Backup of files while in use**

It allows you to perform backups of files that are in use.

**OVER INTERNET**

It allows clients perform backups and restore via internet from any location.

**MULTIPLE STORAGE**

Backups can be stored in different locations, on different servers over other networks or via internet and on different media.

**WEB INTERFACE**

GUI via web that can be accessed from anywhere. You can manage multiple servers and multiple storage arrays at the same time, and doesn't need to be installed along with the DrawerBackup server.

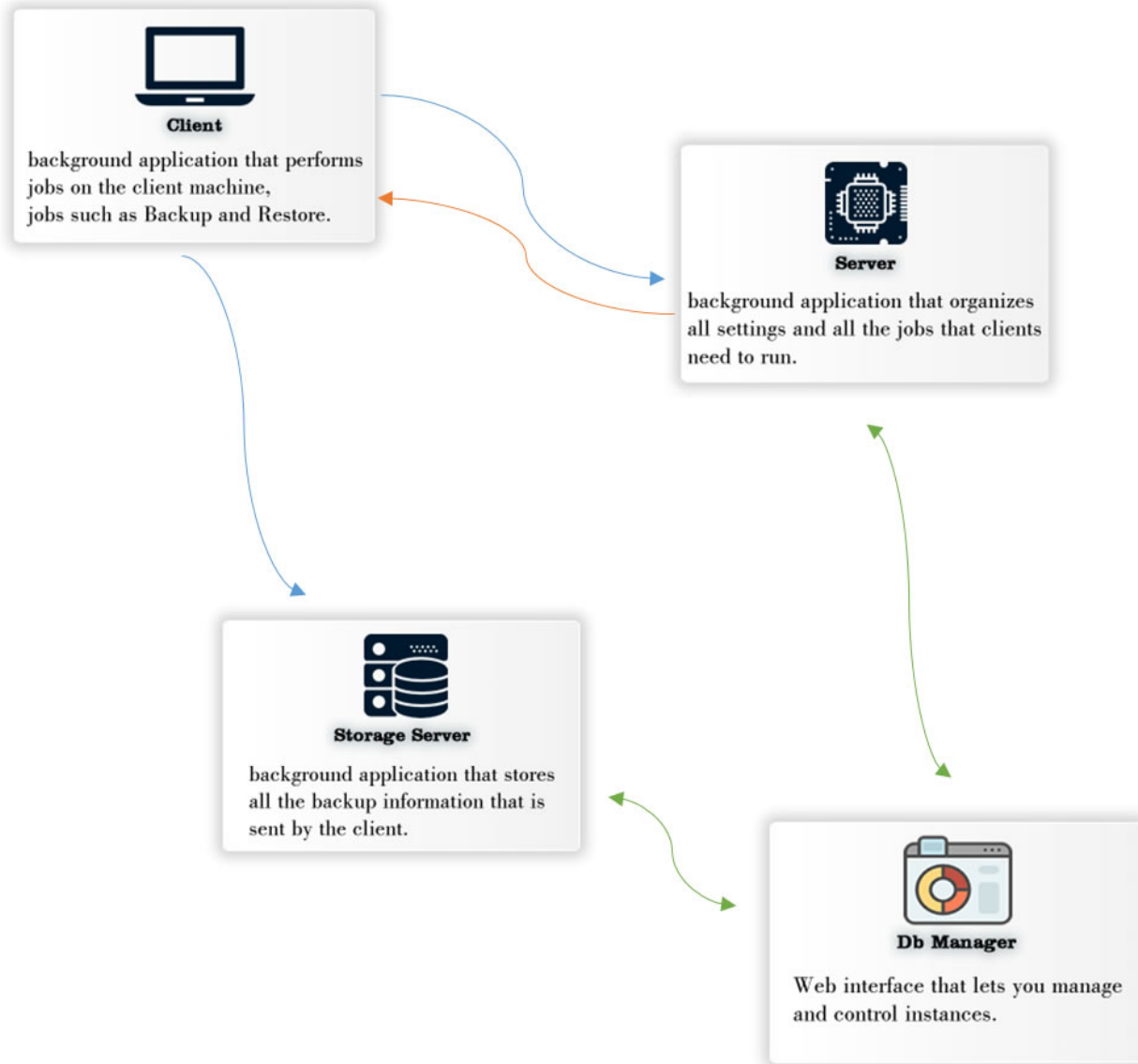
Who needs DrawerBackup?

Anyone who needs a safe copy of your data stored in a organized and indexed form. DrawerBackup stores backups in accordance with the rules that you configure and even via internet outside the network of your servers.

Documentation

3.1 Architecture & Services

The DrawerBackup has a services-based architecture, they are: Storage , Server and Client. As we can see below:



3.1.1 Services

The DrawerBackup is divided into three main services, all services runs as a daemon (or service):

- Server**: The server manages the configuration of the Jobs and the settings of the clients, also keeps stored information about the activities that clients perform.
- Storage**: The storage is a server that stores all backup data generated by the clients.
- Client**: Performs all pre-configured jobs from the server

3.2 Terminology

These are the terms defined by the project:

Storage: It is the server / location where the client stores the backup.

Client: It is the client for the backup server, which suffers backups and perform all works.

Server: Its where is centered all the settings and the server that organized all events.

User Account: In both the server and the storage exists user accounts, these accounts are used by the client to access the resources of the WebS API from the server or storage.

3.2.1 Storage Components

Repository: The repository is a container of images that store information related to backups, each repository have a name, a location on the storage physical disk and a maximum size. A job is always set to a specific repository, when this repository fills to the maximum size. The job will report that storage does not have enough space to store the backup. Unless the job is set to auto-discover a repository, so that will try to use the first free repository configured in storage.

Repository Image: Each repository can have one or more images. Each image stores the backup entries in FLAT format (no rating folders). Each image can have a maximum SIZE. When the image reaches the size limit, the storage creates a new image.

Repository Entry: Each image can contain one or more entries, each entry is pre-allocated before it is stored within the image and represents a client file.

3.2.2 Server Components

Client: It is the client for the backup server, which suffers backups and perform all works. Each client has an ID that used by the Client Daemon to connect in the server.

Client Config: Are the settings for each client, when the client initializes these settings are synchronized.

Client Group: One or more clients may be within a group of clients, groups can be associated with individual jobs.

Job: These works, they can be executed in the client, in the server or in the storage. Each job has a name, a schedule and the particular settings.

Schedule: It is possible to set up multiple schedules and applies each schedule to a job individually.

Operation: When a job executes, it generates an operation, each operation can contain events, entries, progress of the entries, the operation also has flags and a state.

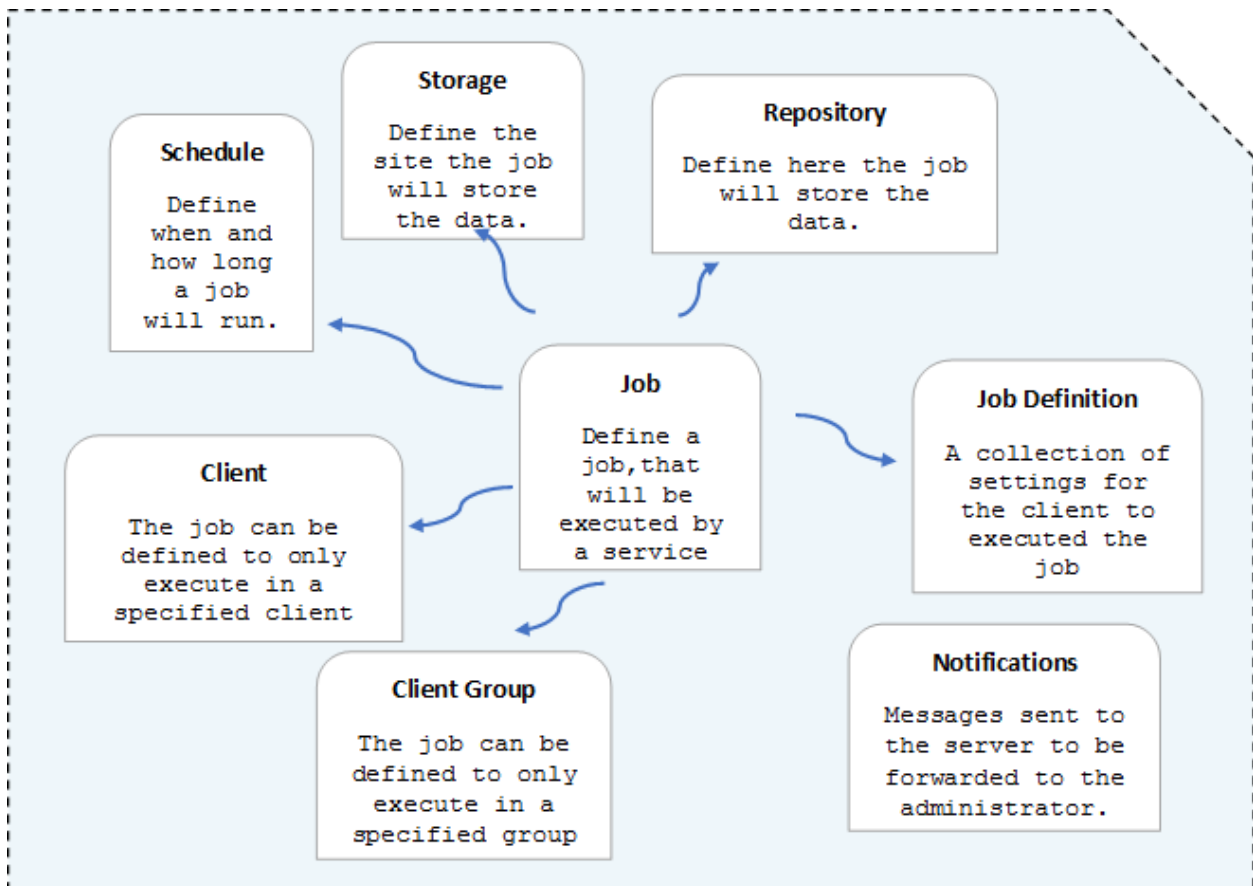
Entry: Each operation may have one or more entries. The entries are the files that begin copied by the operation.

Event: Each operation can generate one or more events, recording the state of the operation and the state of an entry.

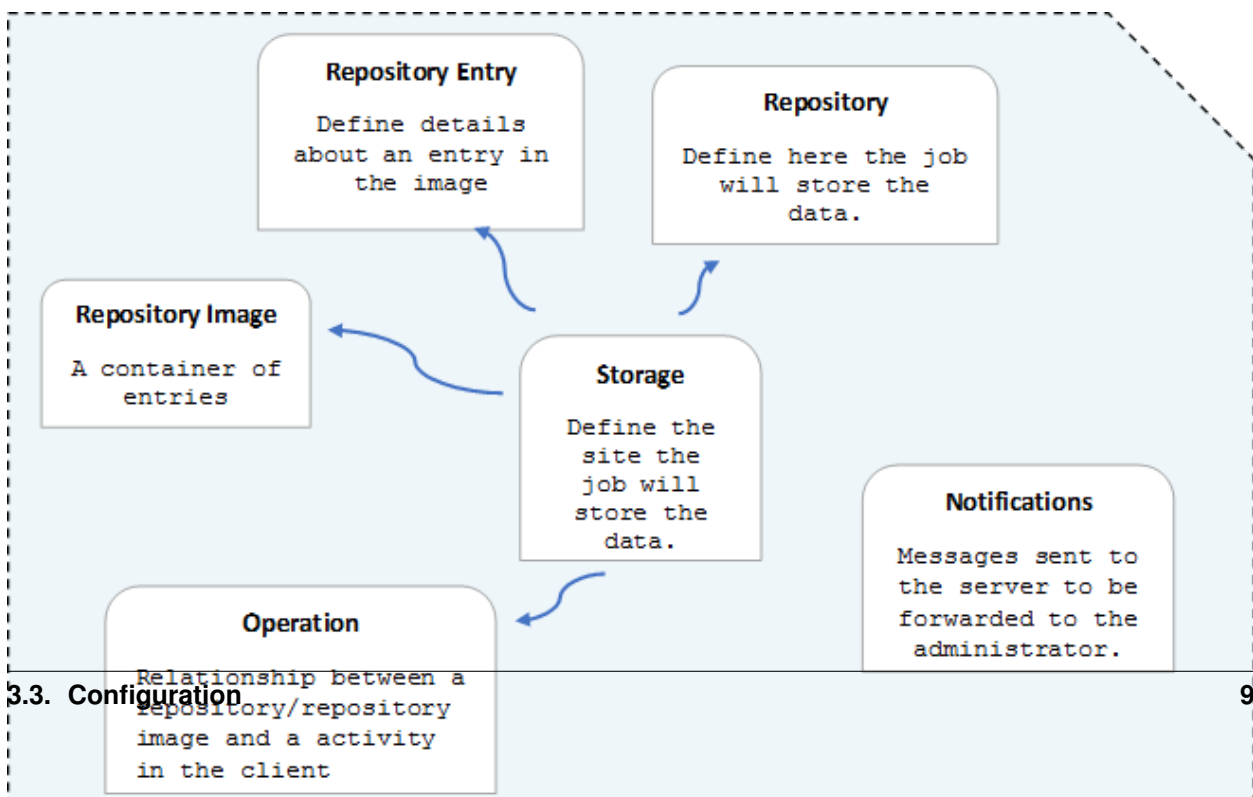
Operation Flag: A flag indicates what the operation is, the flags can be: None, Full, Incremental, Delta, Echo, Restore, Purge

3.3 Configuration

3.3.1 Server Objects



3.3.2 Storage Objects



MULTIPLE JOBS: Support to multiple jobs and parallel job execution.

MULTIPLE STORAGE SITES: Support to multiple storage servers in different locations and via internet.

FAST FILE STORAGE: Stores the data from backup in a fast and indexed way.

BACKUP/RESTORE VIA INTERNET: Can backup and restore jobs from any location and from a client in the internet.

MODULAR AND ADAPTABLE: With the correct configuration and hardware recommended DrawerBackup work in any scenario.

CUSTOMIZABLE: It allows the customization and extension of basic functionality.

CUSTOM JOBS *: Support to customizable jobs.

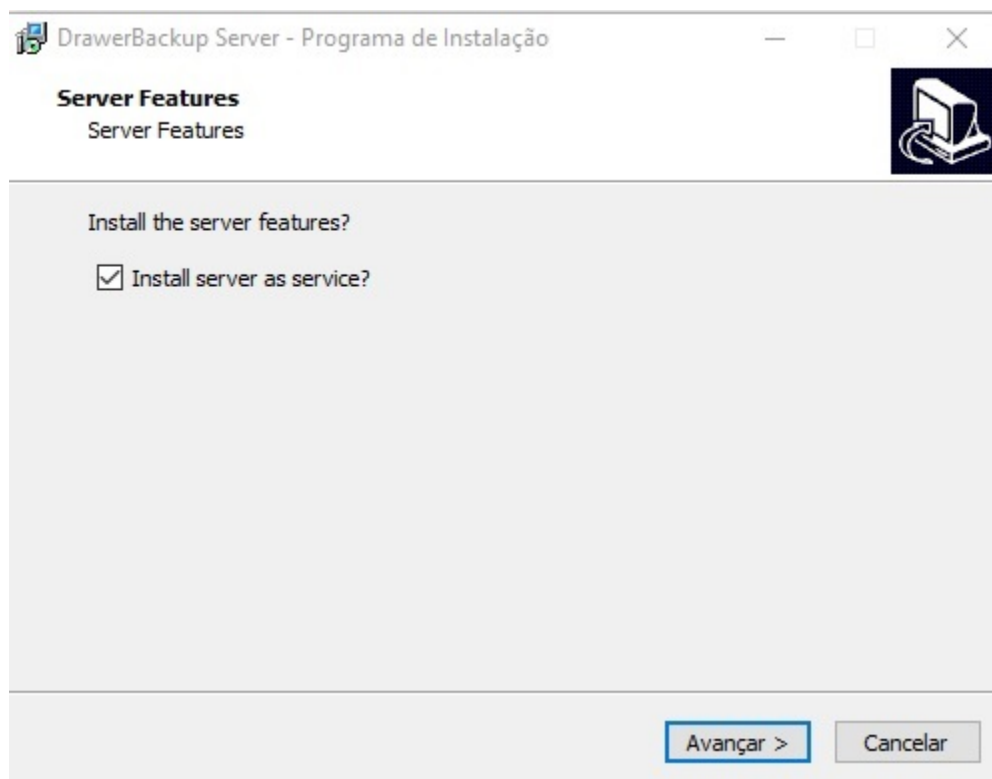
WEB INTERFACE: DrawerBackup Manager is an interface to manage and monitor the servers and storages. Can be accessed from anywhere and can be installed in any compatible web server (i.e Apache or IIS).

3.5 Server/Storage Installation

3.5.1 Server

The Server Installation is simple, just follow the installer steps.

Server Features

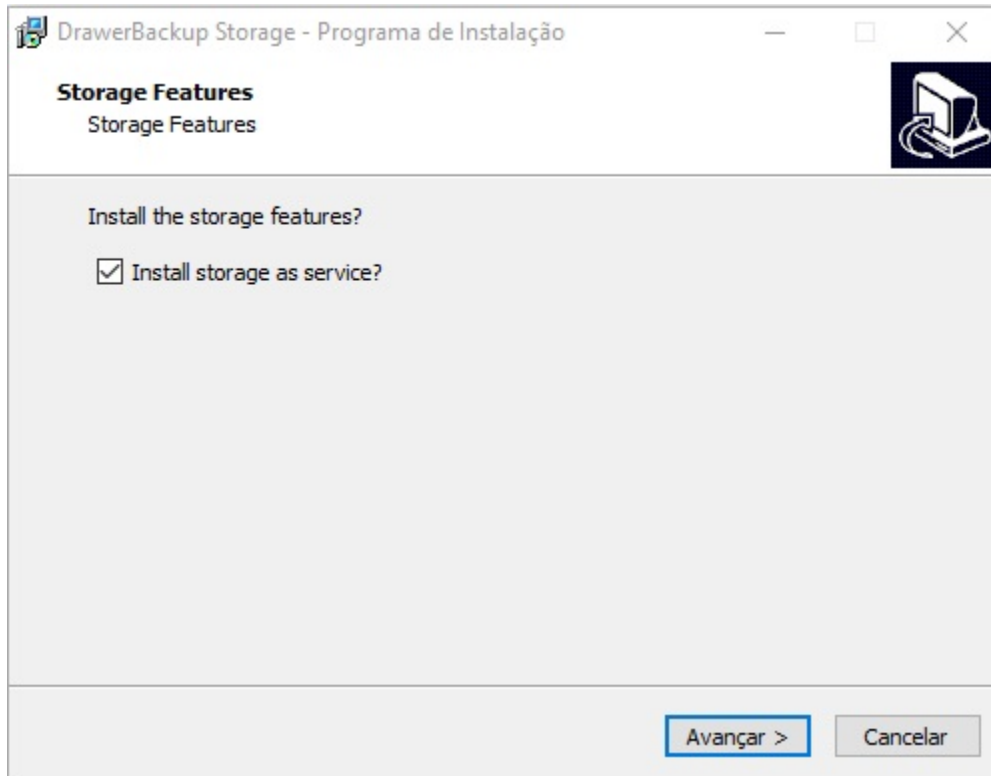


Install Server As Service: If checked the Windows Service of the Server will be installed.

3.5.2 Storage

The Storage Installation is simple, just follow the installer steps.

Storage Features



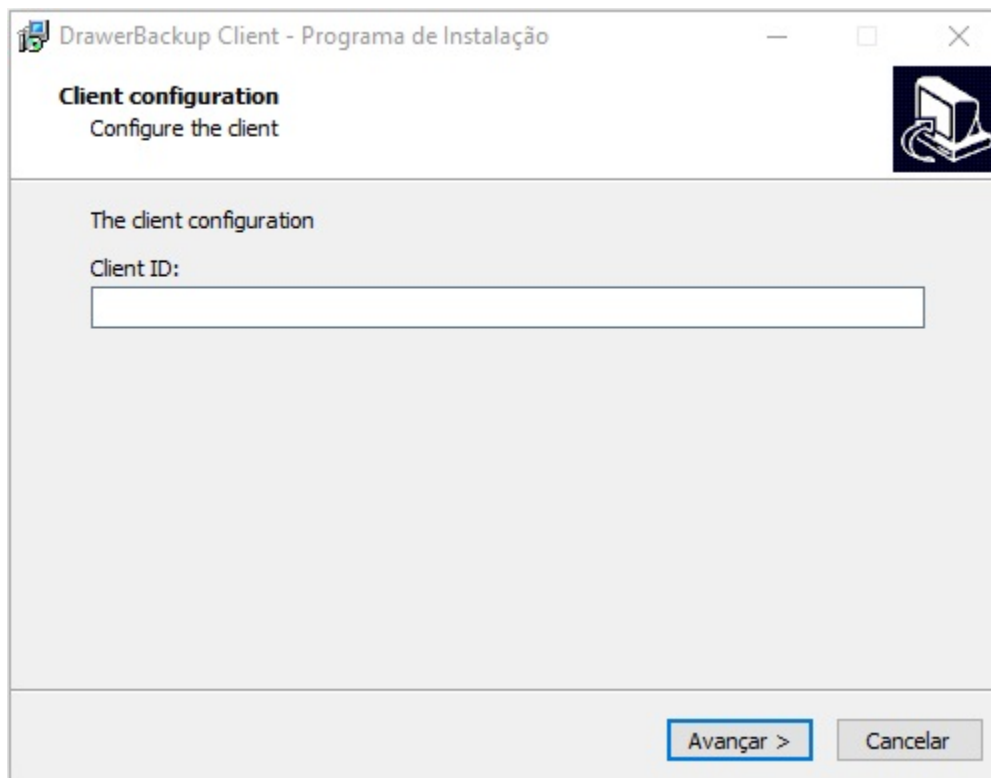
Install Storage As Service: If checked the Windows Service of the Storage will be installed.

3.6 Client Installation

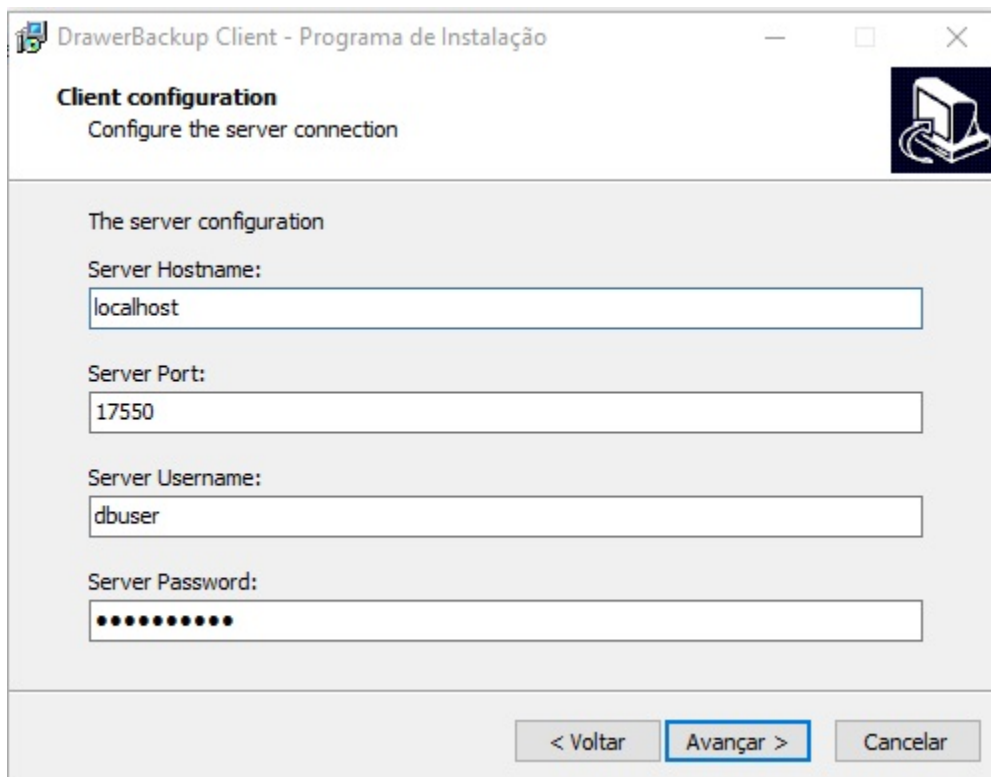
The client installation is a bit more delicate, since there is a little more attention, but just follow the wizard steps.

3.6.1 Client ID

This is the ID of the registered Client on the server, if it's left in blank, the client will request the server a auto-discover to return a valid ID for the current Client.



3.6.2 Server Configuration



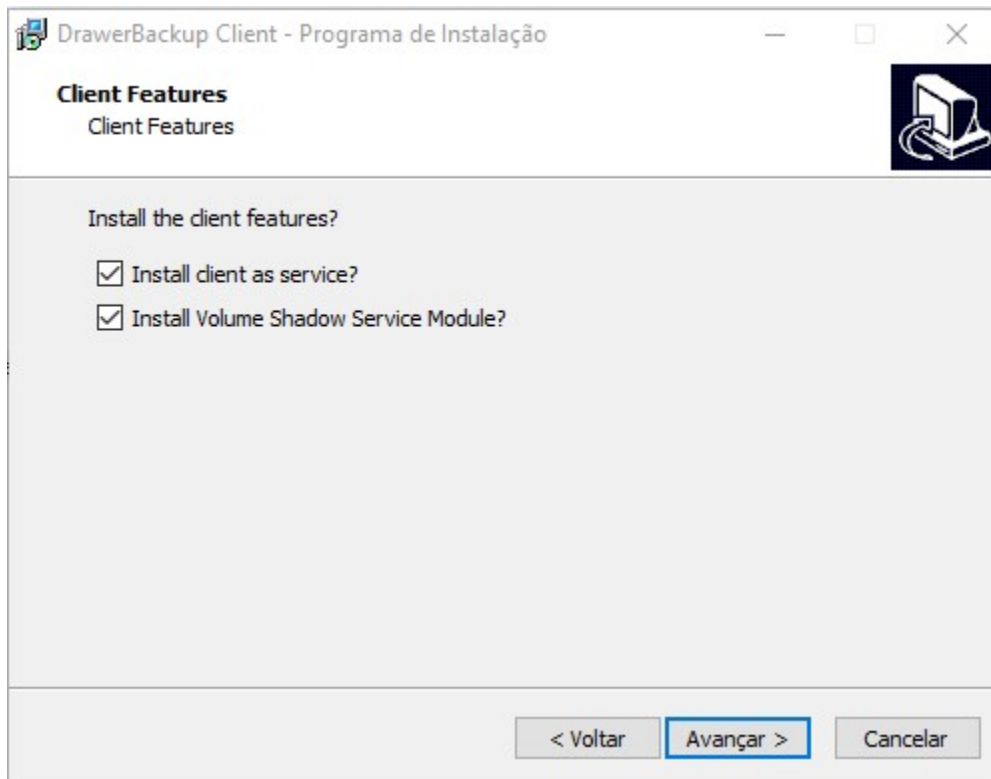
Server Hostname: Hostname or IP Address to the DrawerBackup Server

Server Port: Port of the DrawerBackup Server (Default : 17550)

Server Username: Username of the Account to authenticante in the Server

Server Password: Password of the Account to authenticante in the Server

3.6.3 Client Features



Install Client As Service: If checked the Windows Service of the Client will be installed.

Install Volume Shadow Service Module: If checked the VSS Module for Windows will be installed, *highly recommended if you need to back up files in use.*

3.7 WebUI Installation

The DrawerBackup Manager is a PHP/JS application that can run on any PHP server, such as Apache, IIS , XAMPP. Currently it has been tested on Apache, so is recommended to use a Apache Web Server. it is necessary that you have prior knowledge of the Web Server that will host the application.

Apache Documentation for Windows: <https://httpd.apache.org/docs/current/platform/windows.html> Apache WebSite: <http://httpd.apache.org/docs/>

3.7.1 How to install?

The application installation is simple, just configure the desired web server and copy the Release files (which can be downloaded in Releases of DrawerBackup) to the folder configured on the Web server . And voila! It doesn't require

any specific configuration.

3.8 Quick Start

Before heading to follow this document, see first the documentation of installation and configuration of the services DrawerBackup:

- *Getting Started*

Note that it might be better to follow the steps for configuring the Server before installing the DrawerBackup Client and Storage.

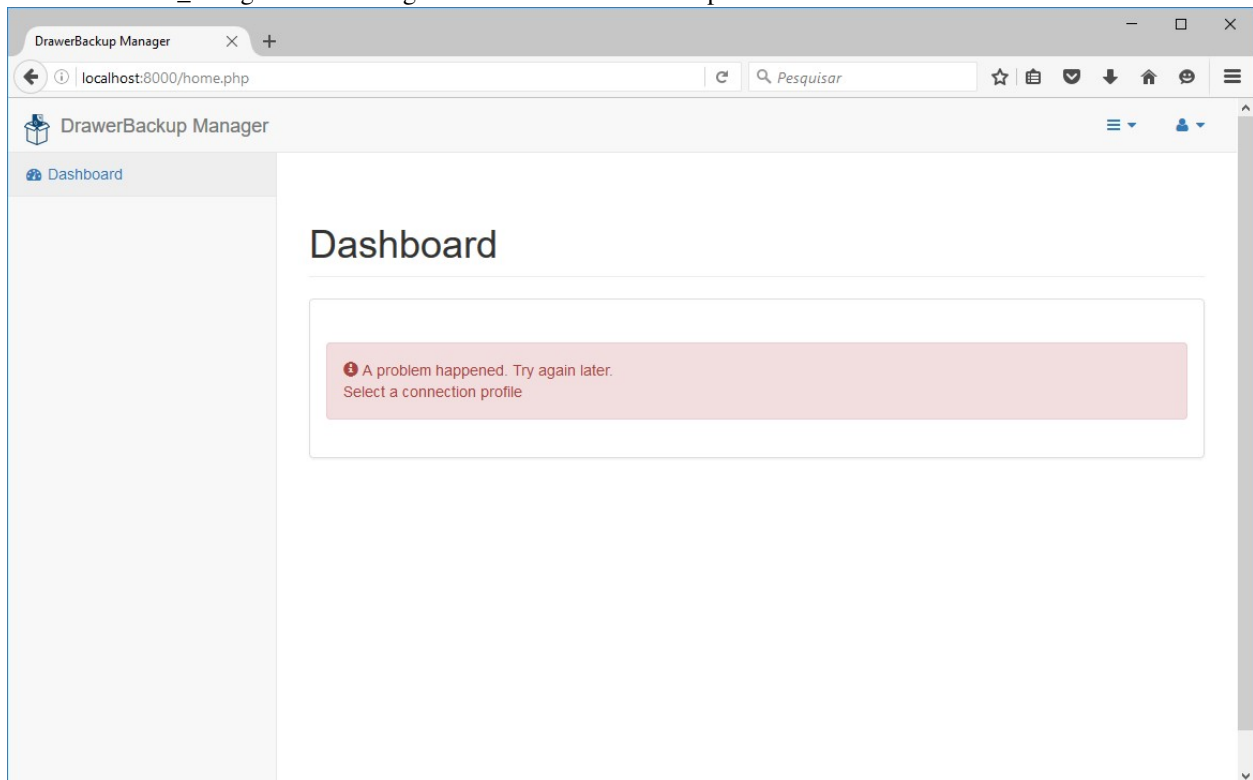
3.8.1 [Step 1] Accessing and Configuring the DrawerBackup Manager

After performing the DrawerBackup Manager installation, we will access it and configure connection profiles for Server and Storage.

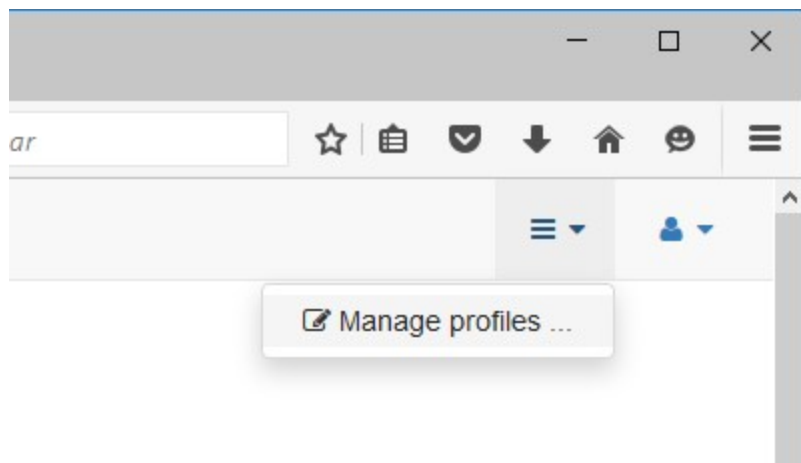
Default Username: Admin

Default Password: dbmanager

See :doc:webUI_changeAuth to change the default username and password.



Select the profile manager:



Default credentials for the Server and Storage:

Default Username: dbuser

Default Password: dbpassword

Create a new connection profile:

New profile

Id
57d2b5e5a6a4d

Name


Kind

Address

Port

Username

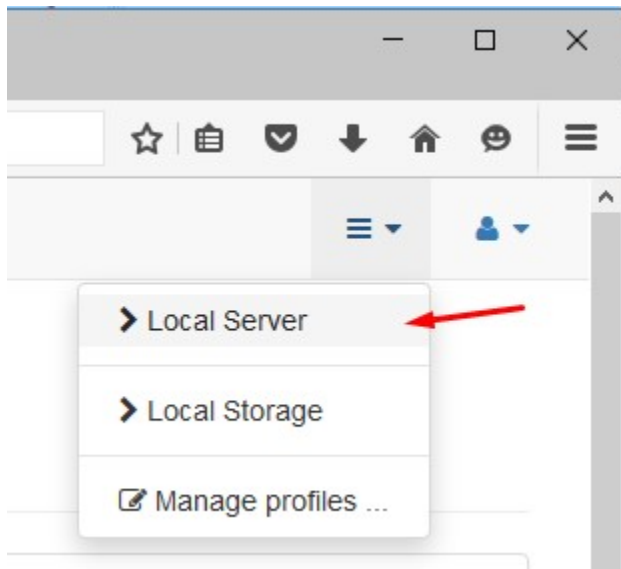
Password

 Save

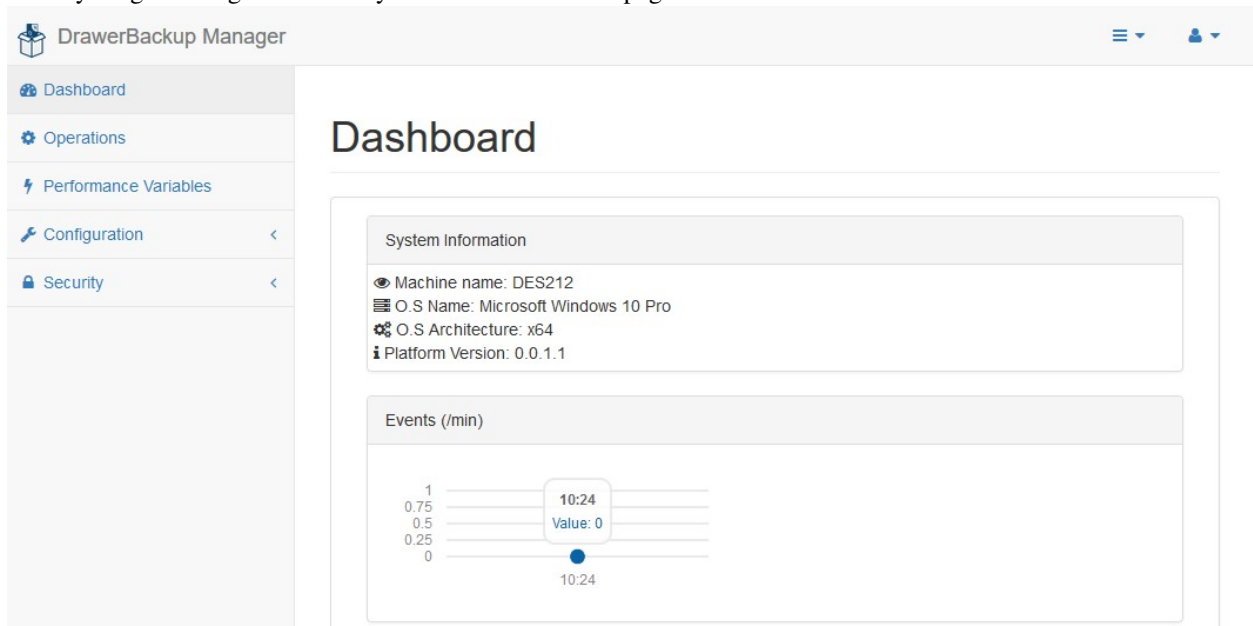
Save and do the same for the Storage.

3.8.2 [Step 2] Configuring the Basics in Server

Select the Server Profile in the List:




If everything is configured correctly it was redirected to a page like this:



Configuring a New Backup Client

The first thing to do is set up an Client on the backup server.

Client

 Save

Id

Name


Group

If you want to place the Client in a Group then configure a client group first. After you save, edit again and copy the Client's ID, this code will be useful in the client installation.

Configuring a New Job Schedule

The Next step is to set up a new schedule to Job. the Schedule uses the pattern Cron , so it is recommended to use a graphical tool for editing: <http://crontab.com/>

Schedule

 Save

Id
e83c84ffa46748fbb61a9852b90b4f28

Name

Until

Schedule

The Schedule follows the CRON format, access this website for a visual editor: CronTab

In this case the schedule was set to 00:30 AM until 01/08/2100.

Configuring a New Storage The next step is to set up the storage in the Server catalog, just like creating a connection

profile for the Storage:

The screenshot shows the DrawerBackup Manager web interface. The browser address bar displays `localhost:8000/App/Server/Storages/edit.php?id=78b70dab28634afca2cbe39338078308`. The left sidebar contains a navigation menu with the following items: Dashboard, Operations, Performance Variables, Configuration (expanded), Clients, Groups of Clients, Jobs, Schedules, Storages (selected), and Security. The main content area is titled "Storage" and contains a form with the following fields: a "Save" button, an "Id" field with the value `78b70dab28634afca2cbe39338078308`, a "Name" field with the value "My Storage", an "Address" field with the value "localhost", a "Port" field with the value "17560", a "Username" field with the value "dbtest", and an empty "Password" field. The footer of the page shows the URL `localhost:8000/App/Server/Storages/` and the copyright notice "DrawerBackup Manager (c) - mInternauta Copyright 2016".


3.8.3 [Step 3] Configuring a Repository in the Storage

Select the storage profile in the list, in the menu go to Configuration > Repositories > New, and create a new repository:

3.8.4 [Step 4] Configuring a Backup for the Client

Select the Server Profile in the List, in the menu go to Configuration > Jobs > New:

Create new job, if you want in the *Lock the job* to select a client, this Job will be performed only by the selected Client:

 Save

Id

Schedule

Group

Is Active ☒ ON

The job repeats
☒ ON

Lock the job to

Name

Kind

Job

Configuration

Name	Value	Description
Storage.ID	<input type="text" value="My Storage"/>	The id of the storage
Repository.ID	<input type="text" value="Test Repository"/>	The id of the repository
Filter	<input type="text" value="ENTITY.Name != null"/>	The filter to the operation indexer
Source	<input type="text" value="C:\FolderToBackup\"/>	The source to the operation indexer
Include.SubFolders	<input checked="" type="checkbox"/> ON	The indexer will include subfolders?

3.8.5 [Step 5] Configuring the Client

Now just follow the client installation ([Client Installation](#)) and inform the previously copied ID when Setup ask. You can run the Console executable (client.exe) and watch him register the configured Job.

3.9 Jobs

Jobs are scheduled actions that DrawerBackup services will perform, every job has some properties:

Name: The name of the job, just a shor description to easy identify the job

Kind: The kind of the job define where that action will be executed. The kind can be: `Server`, `Client` or `Storage`

The job repeats: It indicates whether the job will run only time or several times.

Schedule: The pattern that indicates when the action is executed.

3.9.1 Configuraton or Job Settings

Each Job has its individual settings, see the documentation of the Job to see the documentation of individual settings.

3.9.2 Locking the Job to a Group or Client

For Job that will be performed by the Client Service, these can be locked to run on a specific Client or a specific group of clients.

3.9.3 Job Class Name

The whole logic of the Job is stored in a .NET class that implements the AbstractJob abstract class. The JobClassName property defines the path to this class, it follows the pattern:

[TYPE FULLNAME], [[ASSEMBLY.|SCRIPT.]ASSEMBLY/SCRIPT NAME]

Backup Job Classname:

Can be defined by:

`DrawerBackup.Client.Jobs.BackupJob, DrawerBackup.Client`

Or:

`DrawerBackup.Client.Jobs.BackupJob, ASSEMBLY.DrawerBackup.Client`

Classname referencing a script:

If the job is a script, it can be referenced by:

`DrawerBackup.Client.Jobs.BackupJob, SCRIPT.DrawerBackup.Client`

Note the word SCRIPT., This makes the Job Loader understands that the Job comes from a SCRIPT.

3.10 Schedules

Sets the period/pattern when an action is taken. A schedule is defined using the CRON pattern: <https://en.wikipedia.org/wiki/Cron>

Until: Sets up until when the schedule will be executed , if the date is equal or less to “01/01/1753” then the action will run for an indefinite time.

3.11 Operations

When a job is executed, it generates an operation, each operation can contain events, entries, progress of the entries, the operation also has flags and a state. Some operations may not generate entries, and some operations generate entries when needed.

Example:

Restore Job: Dont generate entries **Incremental Job:** Only generate entries of files that have not been backed up.

Operation properties:

ID: The operation id

Started at: When the operation started

Ended at: When the operation finished

Current State: Current Operation state

Flags: The operation flags

3.11.1 Operation Entries

Each operation may have one or more entries, the entries are the files that were copied by the operation.

Name: The name of entry

Path: The full path of the entry in the client disk

Checksum: A signature for the integrity check.

Length: The length of the entry

3.11.2 Operation Events

Each operation can generate one or more events, recording the state of the operation being processed.

3.12 Clients

Its the client for the backup server , which suffers backups and perform all jobs. Each client has an ID that used by the Client Service to connect to the server.

Properties:

Name: The name of the client

Group: The group of the client

Module of the File System: Check Modules in the wiki

3.12.1 Group of Clients

One or more clients may be within a group of clients, groups can be associated with individual jobs.

3.13 Storages

Its the server / location where the backup is stored by the client. Most jobs use the storage to store the backup information.

Properties:

Name: The name of the storage

Server hostname: The storage hostname or ip address

Server Port: The storage port (Default: 17550)

Username: The username to authenticate in the storage

Password: The password to authenticate in the storage

3.14 Repositories

The repository is a container of images that store information related to backups, each repository have a name, a location on the storage physical disk and a maximum size. A job is always set to a specific repository, when this repository fills to the maximum size the job will report that storage doesn't have enough space to store the backup.

Properties:

Name: The name of the repository

Path: The path of the repository in the disk

Prefix: The prefix of the images in the disk

Max Archive Size: The max size of a image

Max Archives: The max images in the repository (Unlimited: 0)

Max Size: The max size of the repository

For Max Archive Size and Max Size, Use the format: [SIZE][SIZE DEPTH]

Example: 1TB, 1GB, 1MB or 1KB

3.15 Customizable database support

The DataBase Manager is a module that handles the connection of DrawerBackup entities with the engine desired database. The SQL Server CE database module is installed and configured as the default.

Configuration: *DrawerBackup.Native.DataEngines.SQLServerCe.SQLCEDataManager*, *DrawerBackup.Native*

The database is created and automatically updated according to the DrawerBackup entities.

3.16 Built-in Jobs for the Client

3.16.1 Backup & Incremental Backup

These are the two main Jobs of the Client. Both backups perform a copy of indexed files to the storage. The incremental backup will only perform the copy of files that have been modified or have not been copied in previous copies, This job needs at least one full backup performed before.

File to Entry

Each file is converted into an Entry of the current operation, the Entry holds various information about the file as the relative path , the file original size, checksum, modification date and creation date.

The files Indexing

It's the previous list of all files that the Job will have to copy to the storage. The client makes the list of all files and their details at this time is also applied the customizable filter to remove files that user does not want to backup. The indexing can take a long time depending on the amount of files in the destination folder and the performance of the file system and disk where these files are stored.

The checksum

Each Entry receives a SHA key that is generated based on the file contents, that key allows the Client to identify if the file has been transferred correctly to the storage and allows The Incremental Backup to compare the local File to the Storage entry.

Backup Job Settings

These are the individual settings for the Backup job:

Storage.ID: The id of the storage where the backup will be stored.

Repository.ID: The id of the repository in the storage where the backup will be stored.

Source: The location containing the files to be copied, can contain multiple locations separated by semicolons.

Include.SubFolders: true/false , Indicates whether the indexer will include subfolders related to source paths.

Filter: Indexer Filter, see [Advanced Filters for the Indexer](#) for more information.

Incremental Backup Job Settings

These are the individual settings for the Backup job:

Storage.ID: The id of the storage where the backup will be stored.

Repository.ID: The id of the repository in the storage where the backup will be stored.

Source: The location containing the files to be copied, can contain multiple locations separated by semicolons.

Include.SubFolders: true/false , Indicates whether the indexer will include subfolders related to source paths.

Filter: Indexer Filter, see [Advanced Filters for the Indexer](#) for more information.

3.16.2 Restore Job

This Job performs the restoration of an operation (Backup), so its copies and replace the files of an operation that is stored in the storage right back to the client.

Restore Job Settings

Storage.ID: The id of the storage where the backup will be stored.

Repository.ID: The id of the repository in the storage where the backup will be stored.

Destination: The destination where the files are restored, it is recommended leave this field empty, if is empty the files will be restored to the default location, but if you want to restore an operation in another location just type here.

Include.SubFolders: true/false , Indicates whether the indexer will include subfolders related to source paths.

3.17 Advanced Filters for the Indexer

The filter for the index is a C# code for creating filter conditions for the files that will be copied by the backup job.

The filter is based on the .NET FileInfo class: [https://msdn.microsoft.com/pt-br/library/system.io.fileinfo\(v=vs.110\).aspx](https://msdn.microsoft.com/pt-br/library/system.io.fileinfo(v=vs.110).aspx), which is referenced by the variable SUBJECT.

3.17.1 Empty Filter

This is a blank filter that will include all files that have content in the Backup:

```
SUBJECT.Length > 0
```

3.17.2 Filtering a Extension

Natively the indexer implement some extensions in the FileInfo class, in this case we used the method “x” to exclude any file with the extension “.log” or “.tmp”:

```
SUBJECT.WithoutExtension(".log") || SUBJECT.WithoutExtension(".tmp")
```

3.18 Auto-Discovery

The client implements a feature called auto-discovery, this feature is executed when the client doesn't have a configured ID. The client request the auto-discover in the server, the server runs a script that examines whether the Client is already registered, if the client doesn't have a registration the server registers the client in the database. Finally, the server returns the Client ID, so the client can communicate with the server.

3.19 Server Auto-Discover Script

The auto-discovery script allows the server to evaluate whether a Client can be registered by the server.

Default Script:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DrawerBackup.Server.AutoDiscoverable;
using DrawerBackup.Server.Data.Entities;

namespace DrawerBackup.Server.Scripts
{
    public class AutoDiscover : IAutoDiscover
    {
        public bool Accept(AutoDiscoverRequest request)
        {
            return true;
        }

        public Client Create(AutoDiscoverRequest request)
        {
            return new Client( )
            {
                LastSeen = DateTime.Now,
                Name = request.Name
            };
        }
    }
}
```

3.19.1 Accept Method

Should return true/false indicating whether the Client will be registred by the server.

3.19.2 Create Method

Should return the Client Object based on the AutoDiscover Request

3.19.3 AutoDiscoverRequest

The request object has two properties:

Name: The name of the client **Address:** The remote address of the client

3.20 Purging a Operation for the Server/Storage

The Purge Job allows purging and fully remove a operation from the server and storage where it is stored.

3.20.1 Purge Job Settings

Operation.ID: Id of the operation to purge off

3.21 Auto-cleaning up the Old Operations (Backups)

This job will clear all transactions are with the lifetime greater than specified. It can be used to create an old backups cleaning cycle.

3.21.1 Auto Cleanup Job Settings

Operation.Lifetime: The maximun lifetime of a operation, if the operation has a lifetime higher than that, then the operation will be purged off the server and storage. This configuration follows the TimeSpan pattern: [https://msdn.microsoft.com/pt-br/library/se73z7b9\(v=vs.110\).aspx](https://msdn.microsoft.com/pt-br/library/se73z7b9(v=vs.110).aspx)

3.22 Scripts

Scripts are customizable CSharp codes that run the platform for certain actions:

See a example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DrawerBackup.Server.AutoDiscoverable;
using DrawerBackup.Server.Data.Entities;
```

```
namespace DrawerBackup.Server.Scripts
{
    public class AutoDiscover : IAutoDiscover
    {
        public bool Accept(AutoDiscoverRequest request)
        {
            return true;
        }

        public Client Create(AutoDiscoverRequest request)
        {
            return new Client( )
            {
                LastSeen = DateTime.Now,
                Name = request.Name
            };
        }
    }
}
```

3.22.1 Script References

Some scripts need additional assemblies references .NET, this can be done by SCRIPTNAME-References XML, see an example:

```
<?xml version="1.0" encoding="utf-16"?>
<ArrayOfScriptReference xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <ScriptReference Assembly="System" Location="System.Core.dll" />
  <ScriptReference Assembly="System" Location="System.dll" />
  <ScriptReference Assembly="DrawerBackup.Server" Location="" />
  <ScriptReference Assembly="DrawerBackup.Server.Data.Entities" Location="" />
  <ScriptReference Assembly="DrawerBackup.Native.Data.Entities" Location="" />
</ArrayOfScriptReference>
```

3.23 Modules

Modules make some specific functions of Drawer Backup. Module allows replacing and easily customizing that functions. The module configuration follows the following format: [NAMESPACE + CLASS], [ASSEMBLY]

3.23.1 File System Module for the Client

The file system module is executed by the client and allows you to manage the current machine file system.

Built-in File System It uses the basic functions of the machine file system

Configuration: DrawerBackup.Client.Modules.Builtin.BuiltinFileSystem,
DrawerBackup.Client.Modules

Volume Shadow File System Uses the shadow copy service to access the file system. (Recommended)
Windows Only

Configuration: DrawerBackup.FileSystem.VssModule.VssFileSystem,
DrawerBackup.FileSystem.VssModule

3.23.2 File System Module for the Storage

The Storage uses a module that manages the storage of backup entries.

Flat File System The file system in FLAT style where the files are stored in the same folder. See:
https://en.wikipedia.org/wiki/File_system#Flat_file_systems

Configuration: `DrawerBackup.StorageFileSystem.FlatFileSystem.FlatFileSystemModule,`
`DrawerBackup.StorageFileSystem`

3.24 Configurations Files

In all services it is possible to find the advanced configuration files in `INSTALL_DIR/Storage/Settings/`

3.24.1 Authentications Settings

Services: Storage, Client, Server

File: Auth.jcfg

Configurations:

ExpirationTime:

- Description: Time until the authentication token expires
- Type: Number
- Format: seconds
- Default: 60

3.24.2 Data Settings

Configuration related to the catalog (Database)

Services: Storage, Client, Server

File: Data.jcfg

Configurations:

Engine:

- Description: The database Engine which connects to the database.
- Type: String
- Format: .NET Assembly FullName
- Default: `DrawerBackup.Native.Data.Engines.SQLServerCe.SQLCEEngine, DrawerBackup.Native`

3.24.3 Client Settings

Services: Client

File: Client.jcfg

Configurations:

FileSystemModule:

- Description: File System Module classname
- Type: String
- Format: .NET Assembly FullName

Client.ID:

- Description: Id of the client
- Type: String
- Format:

Remote.Username:

- Description: Username to access this client remotely
- Type: String
- Format:

Remote.Password:

- Description: Password to access this client remotely
- Type: SHA1 Hash
- Format: The SHA1 hash for the password

TransferPackage:

- Description: Enable/Disable the package of entries to transfer to the storage
- Type: Boolean
- Format: True/False
- Default: True

TransferPackage.Size:

- Description: The max-length of a transfer package
- Type: Bytes readable string
- Format: 1MB, 1TB ..
- Default: 64MB

TransferPackage.MaxEntries:

- Description: The max number of entries in a tranfer package
- Type: Number
- Format:
- Default: 256

3.24.4 Jobbing Settings

Settings of the Job Manager

Services: Client, Server, Storage

File: Jobbing.jcfg

Configurations:

MaxRunningJobs:

- Description: The number of max simultaneous jobs running.
- Type: Number
- Format:
- Default: 5

3.24.5 Client Monitor Settings

Settings of the Client Monitor, used by the Client Tray Monitor application.

Services: Client, Client Tray Monitor

File: Monitor.jcfg

Configurations:

Server.Hostname:

- Description: The hostname of ip address of the remote client
- Type: String
- Format:

Server.Port:

- Description: The port of the remote client
- Type: String
- Format:
- Default: 17545

Server.Username:

- Description: The username to connect in the remote client
- Type: String
- Format:
- Default: dbremoteclient

Server.Password:

- Description: The Password to connect in the remote client
- Type: String
- Format:
- Default: dbremoteclient

3.24.6 Client Server Connection Settings

Settings for the client connection in the DrawerBackup Server

Services: Client

File: ServerConn.jcfg

Configurations:

Server.Hostname:

- Description: The hostname of ip address of server
- Type: String
- Format:

Server.Port:

- Description: The port of the server
- Type: String
- Format:
- Default: 17550

Server.Username:

- Description: The username to connect in the server
- Type: String
- Format:
- Default:

Server.Password:

- Description: The Password to connect in the server
- Type: String
- Format:
- Default:

3.24.7 Streaming Tweaks and Settings

Settings for the streaming between the client and the storage

Services: Client, Storage

File: Streaming.jcfg

Configurations:

BufferSize:

- Description: The number of bytes to stored in buffer before transfer or read a data
- Type: Number
- Format: Bytes
- Default: 4194304

3.24.8 Debug and Trace Settings

Settings to advanced users, manage the application logging and tracing.

Services: Client, Storage, Server

File: Trace.jcfg

Configurations:

Level:

- Description: The level of tracing data to the logs and output
- Type: String
- Format: SourceLevels ([https://msdn.microsoft.com/pt-br/library/system.diagnostics.sourcelevels\(v=vs.110\).aspx](https://msdn.microsoft.com/pt-br/library/system.diagnostics.sourcelevels(v=vs.110).aspx))
- Default: Information

3.24.9 WebS API Tweaks and Settings

Settings WebServices API of the DrawerBackup Services (WebS)

Services: Client, Storage, Server

File: WebS.jcfg

Configurations:

Port:

- Description: The port where the WebS API will listen
- Type: Number
- Format: Port Range
- Default for Server: 17550
- Default for Storage: 17560
- Default for Client: 17545

Debug:

- Description: Enable/Disable the debug of the requests in the WebS API
- Type: Boolean
- Format: True/False
- Default: False

LocalOnly:

- Description: Enables to the Webs only accepts LAN connections.
- Type: Boolean
- Format: True/False
- Default: False

SwaggerUI :

- Description: Enables the User Interface for the Swagger
- Type: Boolean
- Format: True/False
- Default: False

3.24.10 Storage Image Settings

Settings for the Storage Image Manager

Services: Storage

File: Image.jcfig

Configurations:

Lifetime:

- Description: The maximum time that an image will remain open by the storage.
- Type: Number
- Format: seconds
- Default: 1800

3.25 Configuring the MySQL

It is recommended to use the MySQL only for Server and Storage. In addition to creating a separate database for each instance of the Server or Storage.

To list the available native engines:

```
configtool data -l
```

To change the setting to the MySQL engine:

```
configtool data -d MySQL
```

3.25.1 Configuring the options of the Engine

To configure the Username for the Connection:

```
configtool dbenginecfg -n Username -v [USERNAME]
```

To configure the Password for the Connection:

```
configtool dbenginecfg -n Password -v [PASSWORD]
```

To configure the Address of MySQL Server for the Connection:

```
configtool dbenginecfg -n Address -v [ADDRESS]
```

To configure the Port of MySQL Server for the Connection:

```
configtool dbenginecfg -n Port -v [PORT]
```

To configure the Database name for the Connection:

```
configtool dbenginecfg -n Database -v [DATABASE]
```

3.26 Logs and Output

The services create log files in the folder: /Storages/Logs/

Changing the Level in Trace settings can change the logs causing more information is written.

The Log is separated by Daemon he is being generated. The log structure is:

[SYSTEM MODULE]: [CODE] : [DESCRIPTION] DateTime=[LOG TIME]

3.27 Client And Server Configuration Via CLI

Client Config Tool: configtool.exe

The Config Tool is a common tool in DriverBackup services and helps as a CLI wizard to change the settings of the current service.

The “client” is only important for DrawerBackup Client service settings.

Server Utility: sutility.exe

It is a utility for DrawerBackup Server configuration, with it you can set up schedules, jobs, storages.

Server Utility: strutility.exe

It is a utility for DrawerBackup Storage configuration, with it you can set up repositories.

**** How to use? ****

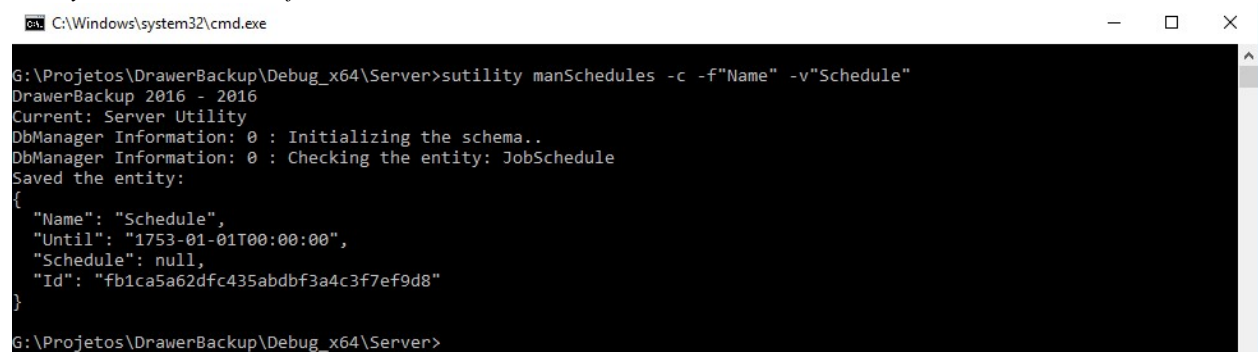
Just launch the executable from the command prompt it will show the available options. So see the help of a option, type *configtool.exe ACTION -help* or *sutility.exe ACTION -help*

3.28 Entity Editor in the CLI

Each DrawerBackup Server and Storage has CLI utilities with the entity editor to manage the catalog entities.

**** Creating a new Entity ****

sutility manSchedules -c -f="Name" -v="Schedule"

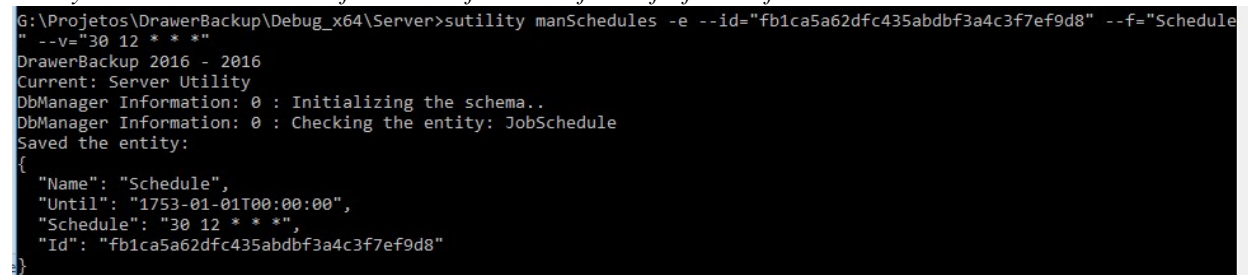


```

C:\Windows\system32\cmd.exe
G:\Projetos\DrawerBackup\Debug_x64\Server>sutility manSchedules -c -f"Name" -v"Schedule"
DrawerBackup 2016 - 2016
Current: Server Utility
DbManager Information: 0 : Initializing the schema..
DbManager Information: 0 : Checking the entity: JobSchedule
Saved the entity:
{
  "Name": "Schedule",
  "Until": "1753-01-01T00:00:00",
  "Schedule": null,
  "Id": "fb1ca5a62dfc435abdbf3a4c3f7ef9d8"
}
G:\Projetos\DrawerBackup\Debug_x64\Server>
  
```

**** Editing a Entity ****

*sutility manSchedules -e -id="fb1ca5a62dfc435abdbf3a4c3f7ef9d8" -f="Schedule" -v="30 12 * * *"*



```

G:\Projetos\DrawerBackup\Debug_x64\Server>sutility manSchedules -e --id="fb1ca5a62dfc435abdbf3a4c3f7ef9d8" --f="Schedule" --v="30 12 * * *"
DrawerBackup 2016 - 2016
Current: Server Utility
DbManager Information: 0 : Initializing the schema..
DbManager Information: 0 : Checking the entity: JobSchedule
Saved the entity:
{
  "Name": "Schedule",
  "Until": "1753-01-01T00:00:00",
  "Schedule": "30 12 * * *",
  "Id": "fb1ca5a62dfc435abdbf3a4c3f7ef9d8"
}
  
```

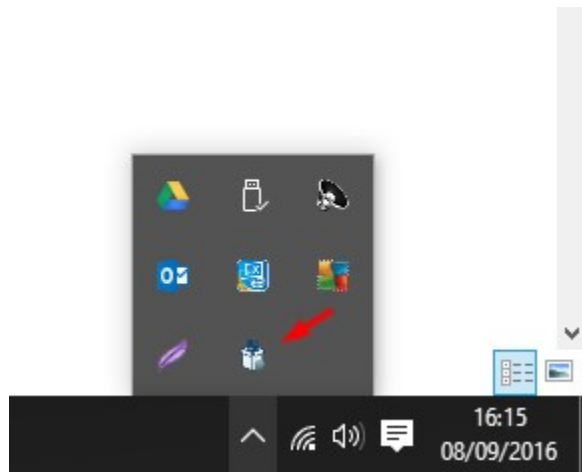
**** Listing all entities ****

sutility manSchedules -l

```
G:\Projetos\DrawerBackup\Debug_x64\Server>sutility manSchedules -l
DrawerBackup 2016 - 2016
Current: Server Utility
DbManager Information: 0 : Initializing the schema..
DbManager Information: 0 : Checking the entity: JobSchedule
Entities in the Database:
5fcd1070d30d4c49b752fb1da7e9d647 , Unit-Testing Schedule
```

3.29 Tray Monitor for the Client

This application runs in the user session and can be viewed in the Windows Tray:



It provides some tools to visualize in real time the current state of the client. See the Monitor settings in the documentation: [Configurations Files](#)

3.30 Performance Variables

All DrawerBackup services perform the collection of performance information and statistics of use, all that information is stored in the performance variables.

3.31 WebS API

The Webs is a set of APIs for communication with DrawerBackup services, the API implements Swagger for documentation.

The URL is:

http://ServerAddress:Serverport/docs/ServerVersion/swagger/

ServerAddress: Deamon address (Client, Server or Storage Address)

ServerPort: Deamon port (Client, Server or Storage Port)

ServerVersion: Platform Version

If case you want to view the graphical interface, simply enable in the Webs configuration:

The URL is:

<http://ServerAddress:Serverport/docs/swagger-ui/index>

3.32 Developer Environment

For you who want to develop the DrawerBackup, first you need to configure the development environment:

Requirements:

- Visual Studio 2015
- .NET Framework 4.5.2
- An administrator user to run the Visual Studio 2015
- NuGet for Visual Studio 2015
- Knowledge in .NET, C# and ASP.NET Web.API
- Sandcastle Help Builder (With the Visual Studio Extension)
- Inno Setup 5

3.32.1 Solutions and Projects

The DrawerBackup is composed of two solutions, which are in two different repositories:

DrawerBackup Solution: *<https://github.com/mInternauta/DrawerBackup>*

DrawerBackup Libraries Solution: *<https://github.com/mInternauta/DrawerBackup-lib>*

You must create a folder with two other subdirectories:

- Libraries
- Source

Where `Libraries` is the clone of the libraries repository (Drawer Backup-lib) and the `Source` the clone of source code repository.

Example:



3.32.2 NuGet Packages

The first thing to do when you open the ANY solution in the Visual Studio is to restore all NuGet packages.

3.32.3 Building

The compilation can be performed by the Visual Studio or by the build.bat script:

build.bat [BUILD MODE] [ARCHITECTURE]

Compiling Release for the X64 platform:

build.bat Release x64

Use the script for the project's release build will also create installers and packages for the API Docs and Drawer-Backup Manager.

Indices and tables

- `search`
- *Getting Started*
- *Platform in Deep*
- *Client*
- *Server*
- *Advanced configuration*
- *Tools and Utilities*
- *Technical Details and Specifications*