2014

IFSS 1.1

USER GUIDE

DATA EXPERTS CONSULTING INTERNATIONAL

TABLE OF CONTENTS

IFSS ARCHITECTURE

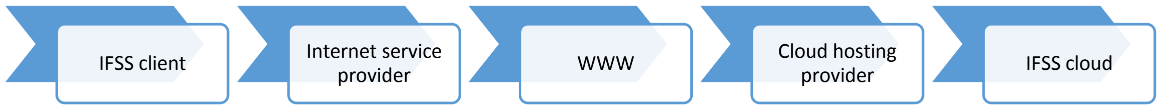
OVERVIEW

Internet File Streaming System (IFSS) is a cloud based electronic files delivery Web service. The primary objective of the service is to deliver files from one user to another in exceptionally fast and secure way.

Users invoke the functionality of the IFSS cloud through specially designed client applications known as *IFSS clients*, which are available for Microsoft Windows PC (XP/Vista/7/8) and Windows Mobile operating systems.

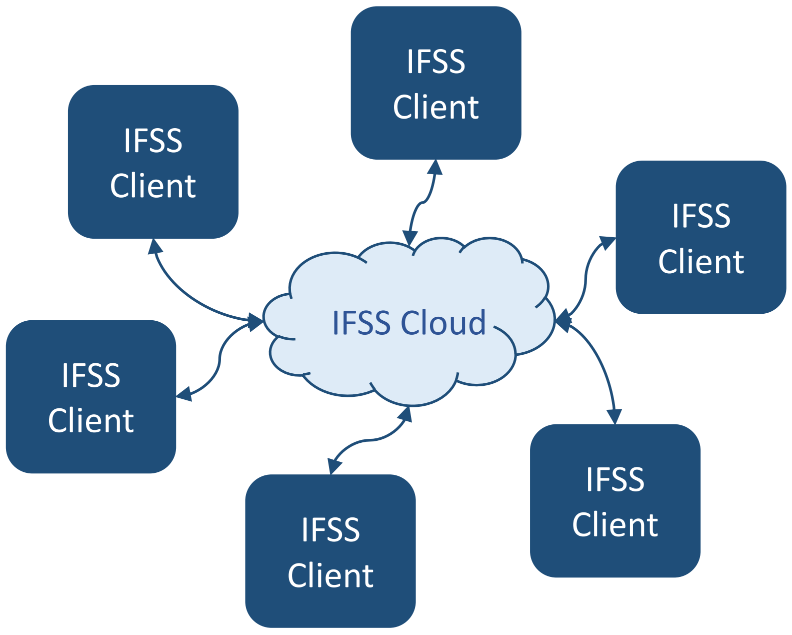
The IFSS cloud is hosted as an ASP.NET Web service on a dedicated Windows IIS Web server. All transactions management is done through an SQL database server. The general scenario of a client/server connection is illustrated in **Figure 1.1** the client application automatically detects Internet connection on a user's device and attempts to log into the cloud with a unique user ID, virtual project ID and password.

Figure 1.1. Connecting client to the IFSS cloud



Any number of registered users can be connected to the cloud at any point in time as shown in **Figure 1.2**. Once connected, users of the same virtual project are able to exchange data as electronic files using one of the client applications.

Figure 1.2. Connecting multiple clients

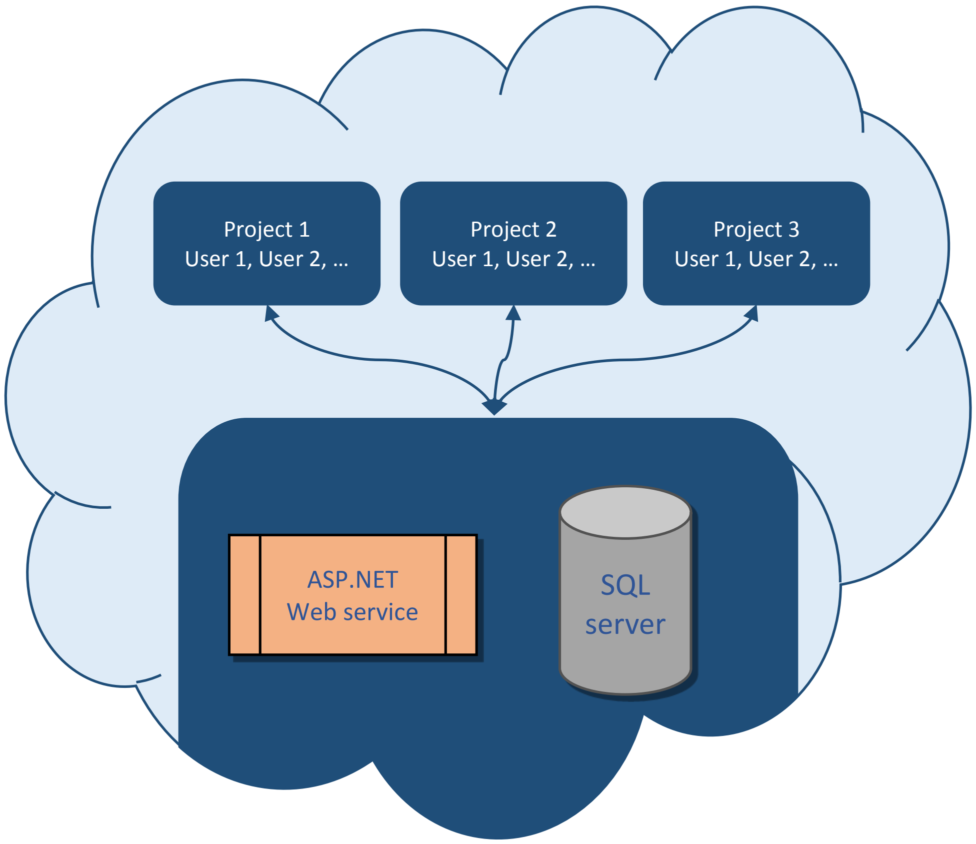


Users do not need to be connected at the same time to send and receive files. The system works similar to e-mail. Every package sent via IFSS has a sender and recipient (or multiple recipients). The server temporarily stores the transmitted files until the recipient logs in and the files are downloaded by their clients. It is important to note that all files sent via IFSS are compressed and encrypted with 128 bit encryption to ensure data security.

The sender has autonomy to select specific files and send them to appropriate recipients via IFSS. The transaction can only be initiated by the said sender. A recipient is unable to access data on other users' devices. Only the sender decides which files should be transferred to which recipients. The process of packing files for delivery is managed by client applications that provide either manual or automated way of doing that. .

The IFSS users are partitioned in virtual projects groups (see **Figure 1.3**). The absolute IFSS restriction is that no data transfer can be done between users of different virtual projects. Virtual projects are discussed in more detail on pp. 8-11.

Figure 1.3. Separation of virtual projects in IFSS structure



IFSS WEB SERVICE

As mentioned above, IFSS Web service is based on ASP.NET Web platform. This Web service processes all incoming requests from the client applications, serving as a façade for the SQL server database and for the package manager object. Therefore, the Web service is the only application the clients communicate with. They do not have direct access to any other part of the system.

Since the underlying protocol is XML-based Simple Object Access Protocol (SOAP), this makes the Web service easily extensible and platform independent. It also helps the system scalability as the client applications can be profiled to connect to any Web server hosted the IFSS service application.

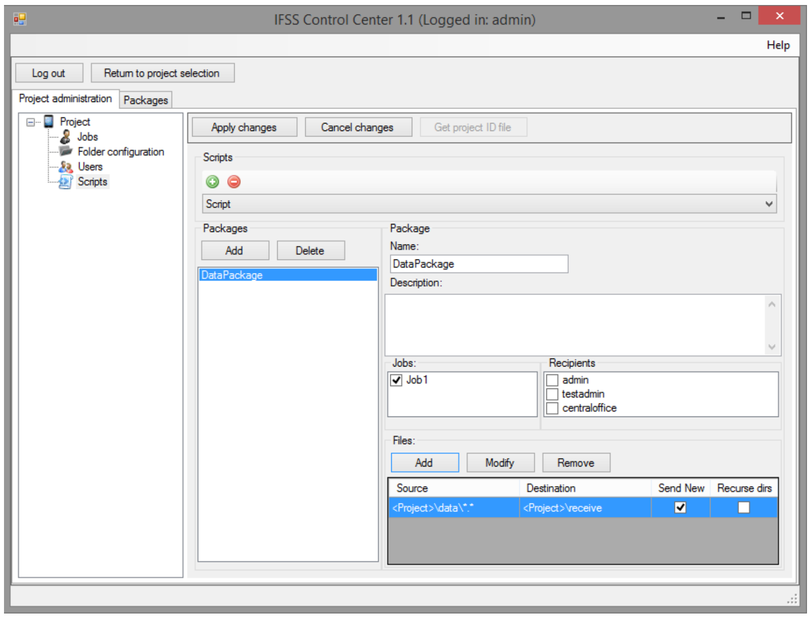
IFSS CLIENT APPLICATIONS

IFSS 1.1 includes the following two types of client applications:

1. Control Center Client available for Microsoft Windows operating system
2. Automated Client available for Microsoft Windows and Windows Mobile operating systems

The Control Center Client application is used for system management. It allows system administrators creating and deleting virtual projects as well as project administrator accounts, manage user account information, and profile connection to IFSS Web services. It also encapsulates full front end control over virtual project metadata and project users as well as provides user interface (UI) for sending, receiving, and tracking data packages.

Figure 1.4. Control Center Client UI: XML scripts management dialog window



Name: DataPackage

Description:

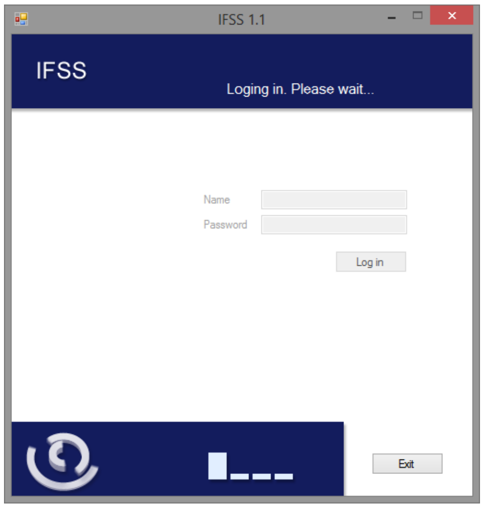
Jobs:	Recipients
☑ Job1	☐ admin
	☐ testadmin
	☐ centraloffice

Files:

Source	Destination	Send New	Recurse dirs
<Project>\data\*.*	<Project>\receive	☑	☐

Automated Client, as the name suggests, is used for sending and receiving data over the IFSS network automatically. This client is installed on the devices intended for sending data via IFSS with limited user interaction. Essentially, that the only action that is required from the user is to log into the client, and the client takes care of the rest automatically.

Figure 1.5. IFSS Automated Client: Login screen



IFSS USER PRIVILEGES (SYSTEM VS PROJECT ADMINSTRATORS)

There are three levels of user privileges in the IFSS system:

- System administrator
- Project administrator
- Project user

System and project administrator accounts can be attached to multiple projects, while the project user accounts can only be created under one given virtual project. **Table 2.1** lists different tasks that can be performed by the IFSS system and shows user privileges for each task.

Table 2.1. User privileges

Tasks	Project user	Project administrator	System administrator
Account management	✓	✓	✓
File transfer	✓	✓	✓
Receive files	✓	✓	✓
Create packages	✓	✓	✓
Install packages	✓	✓	✓
Add/delete user accounts		✓	✓
Manage folder config.		✓	✓
Manage project roles		✓	✓
Assign project encryption keys		✓	✓
Create automation scripts for users		✓	✓
Create schedulers		✓	✓
Invite other administrators to projects		✓	✓
Generate project ID files for users		✓	✓
Create/delete projects			✓
Create/delete project administrator accounts			✓
Set project start and end dates			✓
Activate/deactivate existing projects			✓

Normally, the role of a user is to transfer and receive data. The manner in which these data are transferred is controlled by predefined scripts created by project administrators. However, users can also manually create packages and send them via IFSS. But they cannot control any project related metadata, such as creating or deleting project user accounts, defining project roles and folder configuration profiles, setting encryption keys, and creating scheduler files.

Project administrators have access to all features that project users have. Additionally, project administrators can change any project related metadata except for project start and end dates. They can create or delete project users associated with the project, set encryption keys, project roles, etc. They can also invite other administrators to the project by creating "**admin invitation codes**" (see pp. 30-31 for more detail).

System administrators have access to all features that project users and project administrators have. Additionally, system administrators can change project start and end dates and activate or deactivate projects. They can also create new projects and delete existing projects from the server. On top of it, they have the power to create or delete administrator accounts on both system and project level.

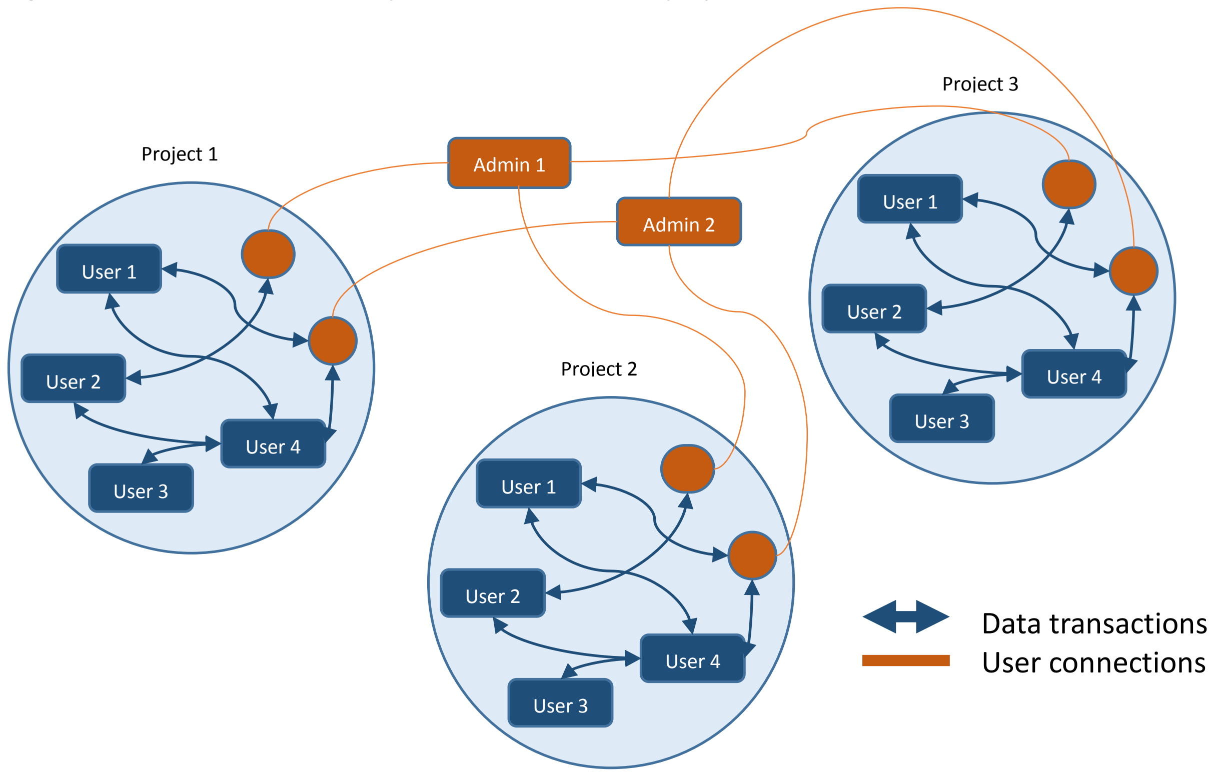
Below is a list of steps for setting up a project by the project administrator:

1. Project administrator requests system administrator to create a new project on server
2. System administrator creates project and shares invitation code with project administrator
3. Project administrator uses invitation code to attach project to their account
4. Project administrator profiles project metadata and creates user accounts for project
5. Project administrator creates automation scripts (if necessary) and provides "Project ID" files to users

IFSS VIRTUAL PROJECTS

The main units of data encapsulation in IFSS are virtual projects. Virtual projects are bubbles on the cloud that define the boundaries of all data transaction between system users. The system is designed in the manner that no data can breach this boundary. Any piece of data transferred via IFSS is always between users within a virtual project. System and project administrators can participate in multiple projects, but they can only transfer data within a project, never between different projects. Project user accounts are created for a specific project and cannot be used in another project. **Figure 3.1** describes the relationships between users and projects.

Figure 3.1. Data transaction encapsulation within virtual projects



PROJECT GLOBAL PROPERTIES

Virtual project can be created in or deleted from the system only by system administrators. Virtual projects have the following global properties (or metadata):

- **Project name** is assigned at the time of project creation by system administrators. It must be unique on the server.
- **Project country** is assigned by system administrators.
- **Project start date** is assigned by system administrators. When a new project is created, the current date is set as the start date by default. The default date can be updated by system administrators.
- **Project end date** is assigned by a system administrator. When a new project is created, the current date plus six following months is set as the end date by default. The default date can be updated by system administrators.

- **Project active flag** is assigned by system administrators and defines project status (active or not active).
- **Encryption key** is randomly generated at the time of project creation. It can be reset by the system or project administrators.
- **Jobs** define grouping roles for users and are assigned by project administrators (see details below).
- **Dynamic folder configuration profiles** are assigned by project administrators. They define dynamic masks for parts of physical file paths and are used in scripts to dynamically adjust paths to source files based on the user's file system (see details below).
- **Scripts** are created by project administrators. There are two types of scripts: (1) Flex logic-based scripts and (2) XML markup-based scripts. When they are attached to a user profile, they are executed by the Automated Client application after login (see details below).

Once a virtual project is established, users of that project are able to exchange files using either Control Center Client or Automated Client application.

JOBS

Jobs are used for grouping users into their respective roles. Each user on the project needs to be assigned to a job. When a package is created, it has an originating user and a list of destination users. Another piece of data in the package header is a list of job indices. The list of recipient determines which users will receive the package from the server, when they log into the client application. However, once the package is received in order for that package to be installed, the user's designated job needs to be present in the list of jobs in the package header. In other words, recipient users will receive the package, but installation of that package is determined by the user job.

This feature enables recipient users to serve as proxies for package transactions. For instance, there is a team of users, which are divided into two jobs: supervisor and interviewers. But only a supervisor user has access to the Internet. A system update is required to be installed on interviewers' devices, but not on supervisors'. In this scenario a package will be created, where a recipient is going to be a supervisor, but the supervisor job will not be included into the list of jobs in the package header. Thus the supervisor, having the only Internet access in the team, will be able to receive the package, but since the supervisor job is not included in the package header the package will not be installed on the supervisor's device. The supervisor then can transfer the package over to interviewers using short-range wireless network or even a flash drive, and the package will get installed on the interviewers' devices.

DYNAMIC FOLDER PROFILES

In order to facilitate a transaction between users using different file path configurations, IFSS utilizes dynamic folder configuration profiles. These profiles can contain masks for files path portions that are different between different devices. Consider a situation where a package is sent to two users: one is running IFSS on a Windows Mobile device and another on a Windows desktop computer. The root project directory on the Windows Mobile device is ".\My Documents\Project," while on the Windows desktop computer it is "C:\\Project." The file that needs to be transferred to these devices should be installed on the Windows Mobile device in the ".\My Documents\Project\Data" folder, while on the Windows desktop computer the same file should be installed in the "C:\Project\Data" folder. One solution is to create two packages and send them to each user separately. However, this is inconvenient and time-consuming. It is more efficient to create two folder profiles with mask "<Project>", one profile named "PC" pointing to the "C:\Project" folder, and second profile named "PDA" pointing to the ".\My

Documents\Project” folder. Then the “PC” profile can be associated with the desktop user and the “PDA” profile with the Windows Mobile user. Then only one package needs to be created with destination directory set as “<Project>\data”. The system will extract the relevant folder profile based on the logged in user and replace “<Project>” path portion with the appropriate value based on the profile associated with the user.

PROJECT USERS

Users of all three levels can be included into a virtual project. The difference is that system administrators and administrators are “attached” to the virtual project, while project users are created inside the project itself. Project user credentials are, therefore, not valid in any project other than the one they have been created under. Project user accounts can be created by the project or system administrators who are already attached to this project. Since project user credentials are not valid outside of the virtual project they have been created in, the Control Center Client application requires project identifier file to be supplied with user credentials. It is not required for system administrators, as Control Center Client allows administrators to select projects that they are attached to. Automated Client always requires the project identifier file, including for administrators, as it maintains data transactions within a particular virtual project space.

FILE TRANSFER FUNCTIONALITY

The main function of IFSS is to transfer files between project users. Files are transferred in packages. Each package can contain one or more files. Files in packages are compressed and encrypted with an encryption key defined by the project administrator or randomly generated by the system. Each package also has a header, which contains information about destination directories for each file, a list of recipients in the project, a list of jobs eligible for this installation, a package name and description.

Packages can be created either through UI of the Control Center Client application, or automatically by the Automated Client application running automation scripts. Manual package creation is useful in case a file or files need to be delivered to users on demand. Automatic package creation is useful when files need to be delivered on a regular basis with no user interaction.

When package is created, the first thing that is required from users is to supply a package name. Package description is optional but may be useful in many scenarios, where upon downloading a package users can evaluate its content based on the description. Then users need to specify package recipients within the project and jobs this package will install on. Finally, one or more files need to be added to the package. The added files need to have a source path and destination directory. Both source path and destination directories can contain masks from dynamic folder configuration profiles. This is especially useful in scripts, where the system is running the same scripts on devices with different file systems. In XML markup-based scripts files can be added in batch buy using wildcard characters. There is also an option to recursively include subdirectories in the source path.

Once a package has been created, it is added to the IFSS local cache on the user's device as an outgoing package. After the handshake procedure between the client application and the IFSS Web service, the file is transferred in pieces to the server. The server registers the package and when the destination user is logged in, the package is pushed over to that user. Once the package receipt transaction has been completed, the package is installed automatically on the destination user's device and also is saved to the cache as a processed package. Packages are processed in the same order as they are received by the server.

FILE TRANSFER AUTOMATION

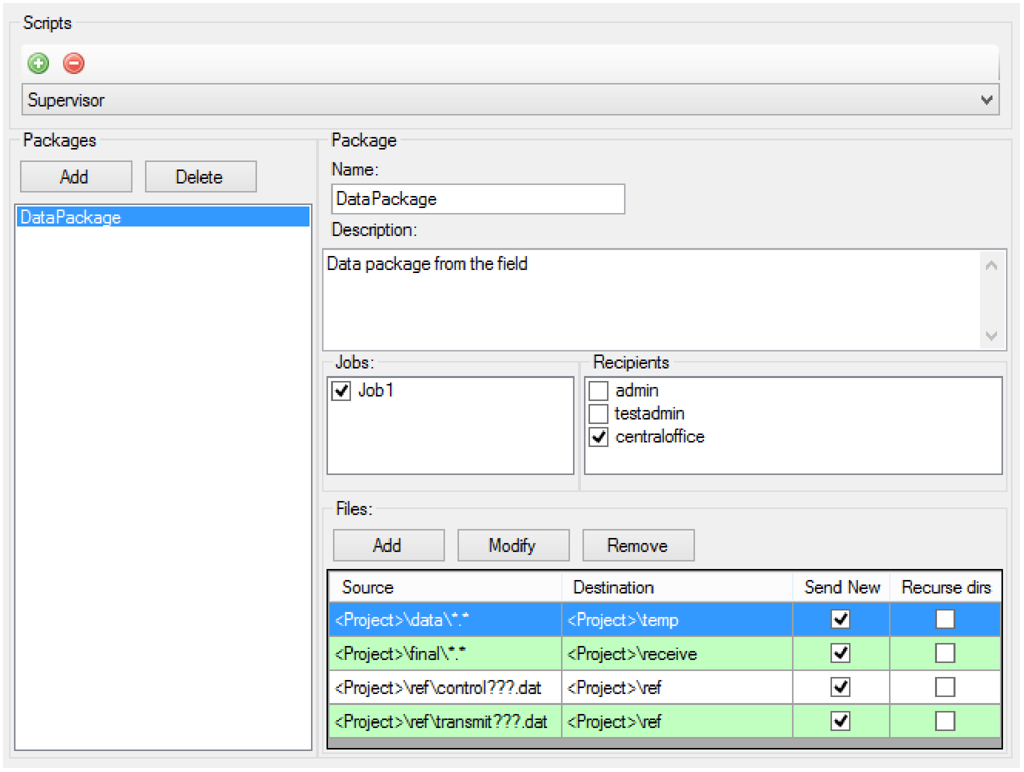
As mentioned above, IFSS has the capability to automate file transfer through scripts, which are executed by the Automated Client application. This feature allows project administrators to design the project so that the flow of data is controlled by the system, instead of a user. Once the script is set up and attached to a project user, this user only needs to log into Automated Client and the system will take care of the rest. In fact, after the user's first log in Automated Client deposits a cookie on the user's device which allows the client application to log in without user providing name and password. From the user's perspective the data transfer becomes a task that can be accomplished by one click.

There are two types of scripts that can be defined in the system: (1) XML markup-based scripts and (2) Flex logic-based scripts. After the script is created, it can be associated with a project user on the "Users" panel in the Control Center Client application (see **Figure 6.15** for more detail).

XML MARKUP-BASED SCRIPTS

XML markup-based scripts are a simple, flexible and straightforward approach to IFSS scripting. It is purely declarative with no logic. The idea is to declare instructions with parameters for the IFSS client to execute. Although under the hood the script is constructed as an XML document, the Control Center Client application provides a convenient UI to construct the script without the need to write any code. **Figure 5.1** shows an example of the UI involved in declaring an XML markup-based script.

Figure 5.1. IFSS XML markup-based script design UI



Scripts: Supervisor

Packages: DataPackage

Name: DataPackage

Description: Data package from the field

Jobs: [✓] Job1

Recipients: [ ] admin, [ ] testadmin, [✓] centraloffice

Source	Destination	Send New	Recurse dirs
<Project>\data\*.*	<Project>\temp	✓	
<Project>\final\*.*	<Project>\receive	✓	
<Project>\ref\control???.dat	<Project>\ref	✓	
<Project>\ref\transmit???.dat	<Project>\ref	✓	

The main instruction that can be declared is to create a data package. A script can contain multiple package creation instructions. After package creation instruction is added to the script, the UI provides the project administrator with a form to profile the following package parameters:

- Package name (required parameter)
- Package description
- Jobs
- Recipients
- Files

The **Files** records contain four parameters as follows:

1. Files **source** path – may include dynamic folder configuration masks and wildcards
2. Files **destination** folder – may include dynamic folder configuration masks
3. Flag for including only **new** or **modified** files in the package
4. Flag for including files **recursively** from subdirectories of the source path

FLEX LOGIC-BASED SCRIPTS

Flex is a custom script language definition. Flex engine library provides a Flex script parser and a socket to plug in custom functions. Flex script was adopted by the IFSS package management infrastructure. It provides a more powerful and flexible way to manipulate package creation procedure by the Automated Client application. The only downside of it is that as it is a real programming language, it is logic-based, and therefore requires writing code, which is more time-consuming and error prone when compared to XML declarative scripts constructed with the help of the UI.

Full Flex script specification lies outside of the scope of this user guide. The following section covers the basics of Flex architecture in the context of IFSS scripting.

Flex logic-based script consists of lines of text enclosed between [Script] and [/Script] tags. The script consists of procedures declarations in the following format:

```
[Script]
main(1)
{
     foo(); #function call without parameters
     function1(arg1, "arg2"); #function call with parameters
}
[/Script]
```

The IFSS Automated Client application loads script procedures and executes a procedure called **"main"** by default. Other procedures may be called from within the **"main"** procedure.

The procedure syntax consists of a procedure name followed by an integer constant enclosed in parenthesis, which indicates how many threads this procedure can be executed on concurrently. In the example above the procedure is single threaded. The procedure commands are then included after the name and thread parameter enclosed in curly brackets.

Commands are represented by function calls. Function call syntax consists of a function name followed by function parameters in parenthesis. Commands are separated by semicolons.

There are only three data types in Flex logic-based scripts:

1. Integer – a 32 bit signed integer value
2. Float – a 32 bit signed floating point value
3. String

Flex interpreter does not enforce any sort of type safety. Therefore, expressions can mix arguments of any of the three types. The type of the result of the operation depends on the two arguments used. For instance, if we have two variables ‘a:integer’ and ‘b:string’, operation (a + b) will return a string value, where ‘a’ is converted to string and concatenated with ‘b’. Integer and Float literals are written as numbers and String literals are enclosed in “double quotes”.

Appendix A includes a reference for the most commonly used Flex built in functions as well as a list of IFSS custom functions (pp. 35-36).

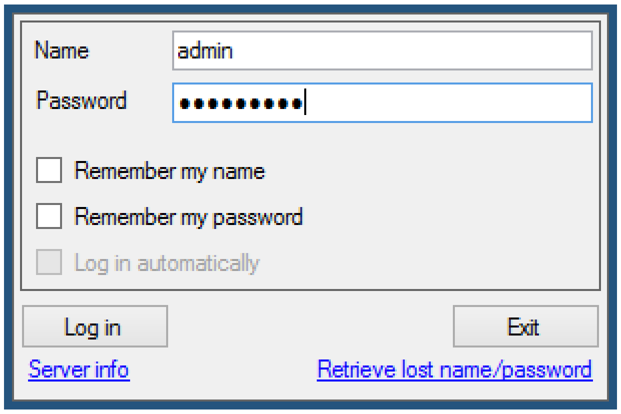
CONTROL CENTER CLIENT APPLICATION

The Control Center Client application serves as a main front end access interface of IFSS. It allows system and project administrators to control all features of the system. It also includes functionality to construct, send, and receive IFSS data packages as well as track package delivery to recipients.

LOGIN SCREEN

The first screen of the IFSS Control Center Client application is the login screen. Users need to provide a valid user name and password in order to be able to log into the system.

Figure 6.1. IFSS login dialog window


Name: admin
Password: •••••••••

☐ Remember my name
☐ Remember my password
☐ Log in automatically

In the login screen users can also instruct the client to deposit a cookie, which will enable the client to remember the username and password for future login and/or to log in automatically with these saved credentials, bypassing login screen entirely.

Additional actions that can be performed by users on the login screen are setup/change server connection configuration and retrieve lost user credentials as discussed below.

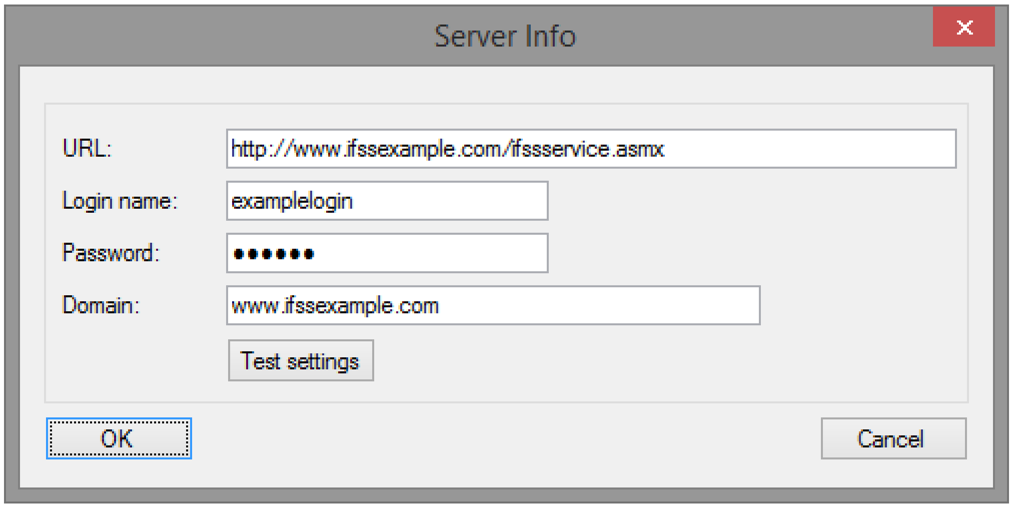
SERVER INFO

The **Server info** link on the login screen enables the user to set up IFSS server connection properties. As multiple IFSS servers are available on the World Wide Web, Control Center Client needs to know which of these servers to connect to.

As shown in **Figure 6.2**, these credentials include the following four fields:

- **URL** to the IFSS Web service
- Server user **login name**
- Server user **password**
- IFSS **domain**

Figure 6.2. IFSS server connection properties



URL: http://www.ifssexample.com/ifssservice.asmx
Login name: examplelogin
Password: ●●●●●●
Domain: www.ifssexample.com

If unknown, server connection information can be provided by system administrators. Once it has been set up, the client application saves this information and retains it for future use. Only system and project administrators should have access to these properties. When project users log into any of the IFSS clients, this information is retrieved from the encrypted "project ID file" supplied to users by project administrators.

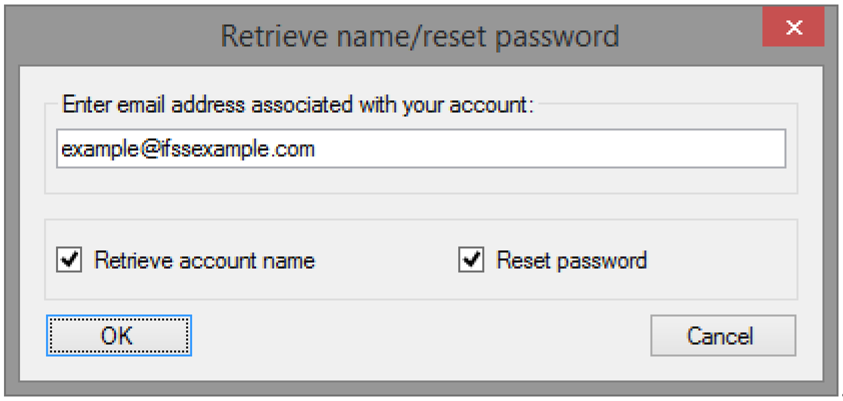
It is important to note that the server login name and password are different from the IFSS user account name and password. These are credentials to access the IFSS Web service, while IFSS user account credentials are for logging into the service after the connection has already been established and verified.

The **Test settings** button can be used to test server connection. Once clicked on, it will attempt to establish connection to the server with supplied credentials and will produce a message based on the result of the made attempt.

RETRIEVING LOST ADMIN ACCOUNT INFORMATION

If IFSS login credentials of system or project administrators are lost, they can be retrieved by clicking on the **Retrieve lost name/password** link on the login screen of the IFSS Control Center Client application (**Figure 6.1**). Then the user credentials retrieval dialog window appears as shown in the **Figure 6.3**.

Figure 6.3. IFSS user credentials retrieval dialog window

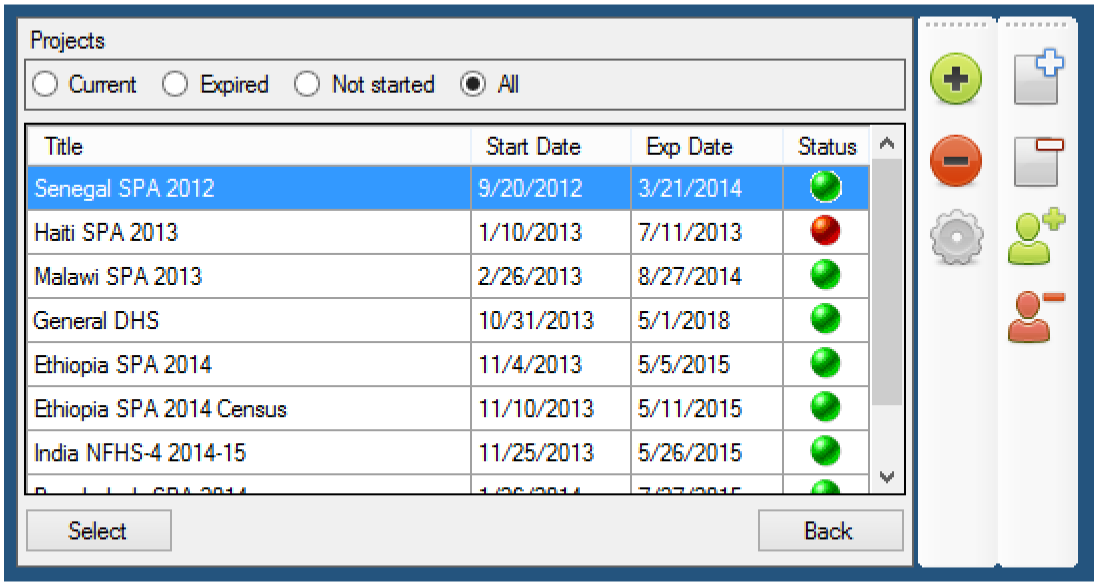

Retrieve name/reset password
Enter email address associated with your account:
example@ifssexample.com
[✓] Retrieve account name
[✓] Reset password
OK
Cancel

To be able to retrieve lost account information, the user needs to provide their **e-mail address** that was used when account was first created. After entering the e-mail address the user can retrieve the **account name** and/or **reset password**. Once the **OK** button is clicked on, the system will send account user name and temporary password on the specified e-mail address. It is important to note that for security reasons the account name and temporary password are sent to the same user in separate e-mail messages. As soon as the user receives the temporary password, they should change it in the system upon next login.

PROJECT SELECTION SCREEN

Since system and project administrators can participate in multiple projects, the second screen after the login screen lands them to the project selection screen. Project users bypass this screen and directly log into the project management console. **Figure 6.4** demonstrates an example of the project selection screen for a system administrator.

Figure 6.4. Project selection screen



Title	Start Date	Exp Date	Status
Senegal SPA 2012	9/20/2012	3/21/2014	🟢
Haiti SPA 2013	1/10/2013	7/11/2013	🔴
Malawi SPA 2013	2/26/2013	8/27/2014	🟢
General DHS	10/31/2013	5/1/2018	🟢
Ethiopia SPA 2014	11/4/2013	5/5/2015	🟢
Ethiopia SPA 2014 Census	11/10/2013	5/11/2015	🟢
India NFHS-4 2014-15	11/25/2013	5/26/2015	🟢

The project selection dialog window contains a grid with projects that the user has attached to their account and two strips with control buttons on the right-hand side of the screen. Project administrators only have access to first controls strip, while system administrators have access to additional controls as described below.

The project administrator strip includes the following buttons:

- Attach project ( )
- Disassociate from project ( )
- Edit profile ( )

In addition to the project administrator buttons, system administrators also get access to:

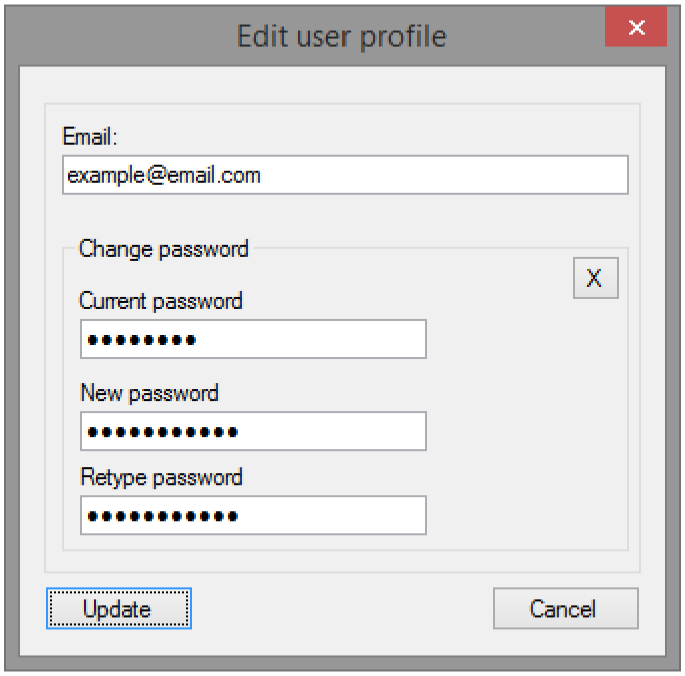
- Create new project ( )
- Delete project ( )
- Add new administrator account ( )
- Delete existing administrator account ( )

The project selection grid contains basic information about projects: **title**, **start date**, **end date**, and **status** indicator. Projects can be filtered by their status.

EDITING USER PROFILE

The edit profile button ( ) on the project selection screen allows project administrators to change their e-mail address and password. When clicked on, the following dialog window appears as shown in **Figure 6.5**.

Figure 6.5. User profile editing dialog window


Edit user profile

Email:
example@email.com

Change password

Current password
••••••••

New password
•••••••••••

Retype password
•••••••••••

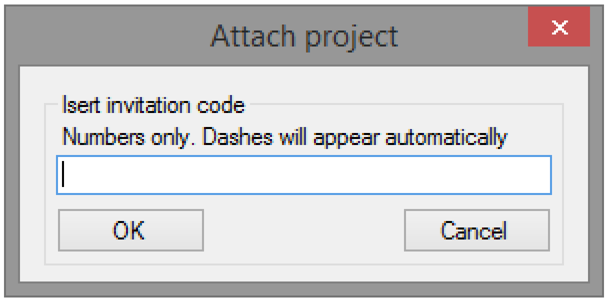
ATTACHING AND DISASSOCIATING PROJECTS

The attach and disassociate from project buttons (➕ and ➖) on the project selection screen for project administrators are used to attach to and disassociate projects from the administrator account. It is important to note that attaching to and disassociating from a project is not the same as creating and deleting a project. Projects can be created or deleted only by system administrators. By attaching a project system or project administrators simply attach their user profile to a virtual project space. This can only be done with projects that already exist. Disassociating means removing users profile from the project space. The project itself is not deleted from the server.

Attaching project administrators to existing projects is done via invitation codes. Only existing project administrators on the project can create invitation codes for other project administrators. The code is a 16 digit number generated by administrator's request and remains active for 24 hours since its creation. Once the code has been created by a system or project administrator attached to this project, it can be shared with other system or project administrators who have yet to be attached to the project.

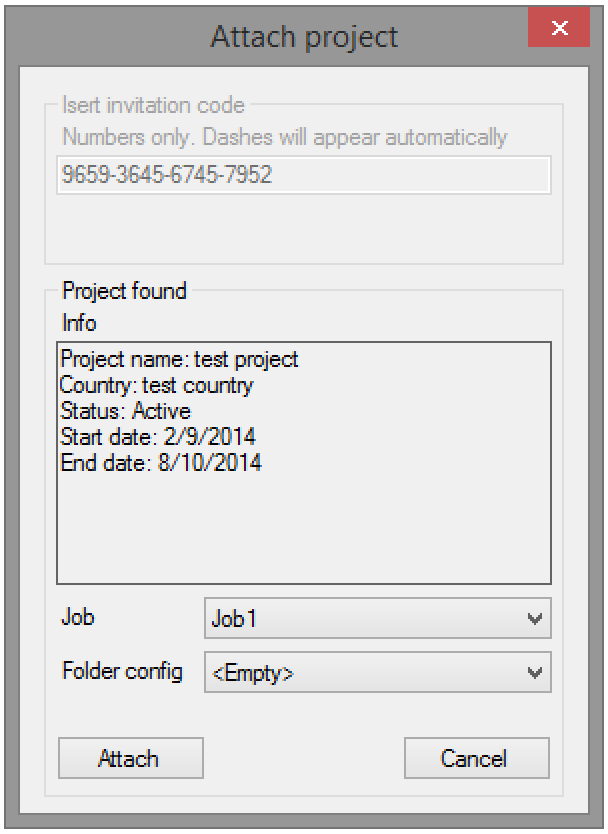
When the attach project button (➕) is clicked on, the following dialog window will request the invitation code from the user (**Figure 6.6**):

Figure 6.6. Attach project dialog window



Entering the code provided by another project administrator and clicking the **OK** button, if entered correctly, will expand the dialog window further (**Figure 6.7**):

Figure 6.7. Attach project dialog window after entering the valid admin invitation code



9659-3645-6745-7952

Project name: test project
Country: test country
Status: Active
Start date: 2/9/2014
End date: 8/10/2014

Job: Job1
Folder config: <Empty>

The expanded dialog window now displays the following project basic information: **project name**, **country**, **status**, **start** and **end dates**. It also allows users to select relevant **job** and **folder configuration** profile. After clicking on the **Attach** button the project will be attached to the project administrator account and will appear in their project selection screen.

CREATING AND DELETING PROJECTS (SYSTEM ADMINISTRATORS)

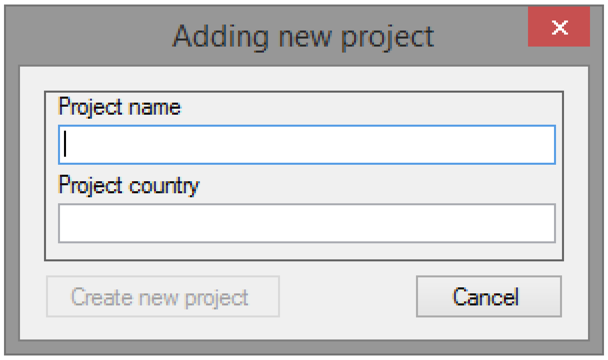
System administrators have privileges to add new projects to the cloud and remove them. The corresponding buttons on the project selection screen are: for creating new project and for removing projects. Removing a project from the server is different from disassociating from this project. Disassociation only implies removing the link between the user account and the project, while removing the project deletes all project related metadata from the server permanently. These metadata include:

- Links to other administrator accounts
- All project user accounts
- All project properties
- All packages (delivered and in transit)
- A project record on the server

Once the project has been removed, no data can be transferred by any user in that project. Therefore, system administrators need to exercise caution while removing projects from the server.

Clicking the create project button ( ) produces the following dialog window (**Figure 6.8**):

Figure 6.8. Project creation dialog window



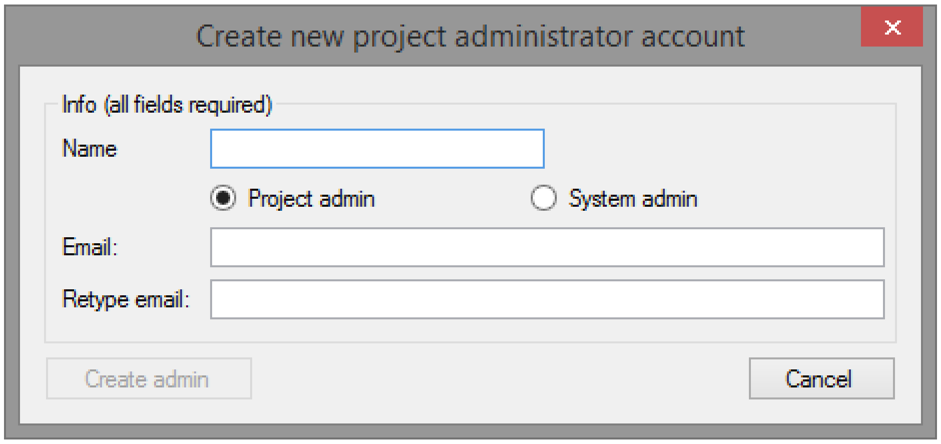
In order to create a project system administrators need to supply a project name and project country. A project name must be unique. The system will not allow two projects with the same name on the server. However, projects with identical names can be created on different servers.

Once the project has been created, the creator's administrator account is automatically attached to the project, and the record is also added to the project selection roster.

CREATING AND DELETING ADMINISTRATOR ACCOUNTS

Apart from creating and deleting projects, system administrators can also create and delete other administrator accounts. The button to create a new administrator account is marked with the icon. When clicked on, the following dialog window appears (**Figure 6.9**):

Figure 6.9. Create new administrator account dialog window



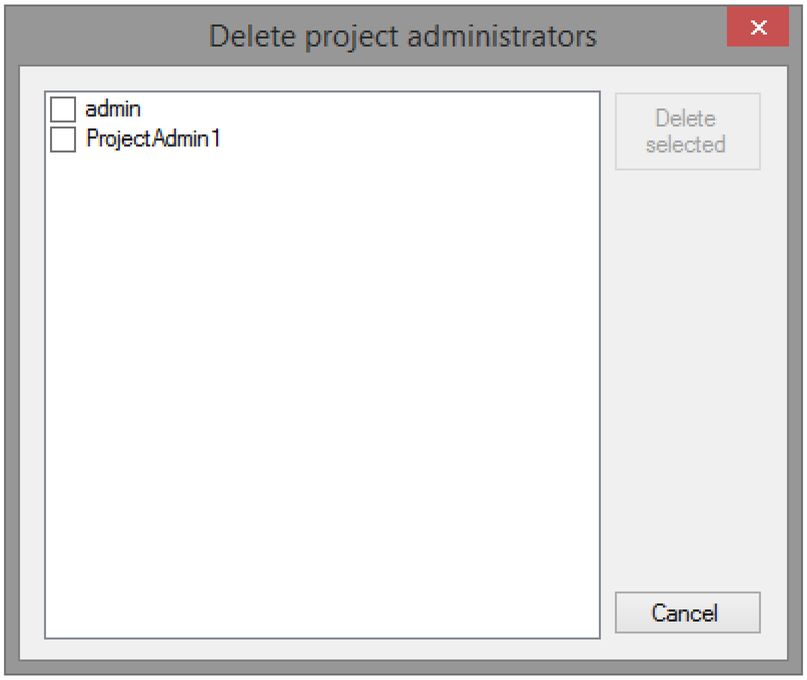
System administrators need to supply an account name for a new administrator account. This name must be unique in the system. Project user names don't need to be unique in the system, only inside their designated projects. Administrator accounts penetrate project boundaries and can have more than one project attached to them. Therefore, their name, being the main account identifier, needs to be unique in the system.

When creating new administrator account it is important to specify what level of privileges this account will have. There are **two types of administrators** in IFSS: **system administrators** and **project administrators**. Most of the time project administrator account is sufficient for the majority of the tasks. System administrator accounts only need to be created for users with complete access to the server. Normally, no more than one system administrator account is present per server.

Finally, a valid **e-mail address** must be specified when creating a new administrator account. Supplying the correct e-mail address is important, because the system will create a new account, randomly generate a password for this account, and send this password to the specified e-mail address. The recipient of this e-mail will have access to the system with the newly created profile. This way the system administrator who created the account will not be able to have access to it.

The button that allows system administrators to delete existing administrator accounts is marked by the icon. When clicked on, the following dialog window appears (**Figure 6.10**):

Figure 6.10. Delete administrator accounts dialog window

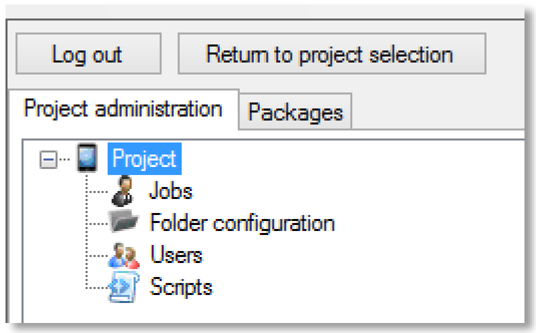


This dialog window contains a list of all administrator accounts created on the server. By marking them and clicking on **the Delete selected** button, system administrators can permanently remove these accounts from the server. After that is done no login is possible with the deleted credentials.

PROJECT ADMINISTRATION USER INTERFACE

After the **Select** button is clicked on the project selection screen (**Figure 6.4**), the client application will request the server for the selected project, download all metadata relevant to that project, and will switch to the project administration screen. The project metadata structure is represented by a tree of the project administration panels (**Figure 6.11**)

Figure 6.11. Project metadata tree.



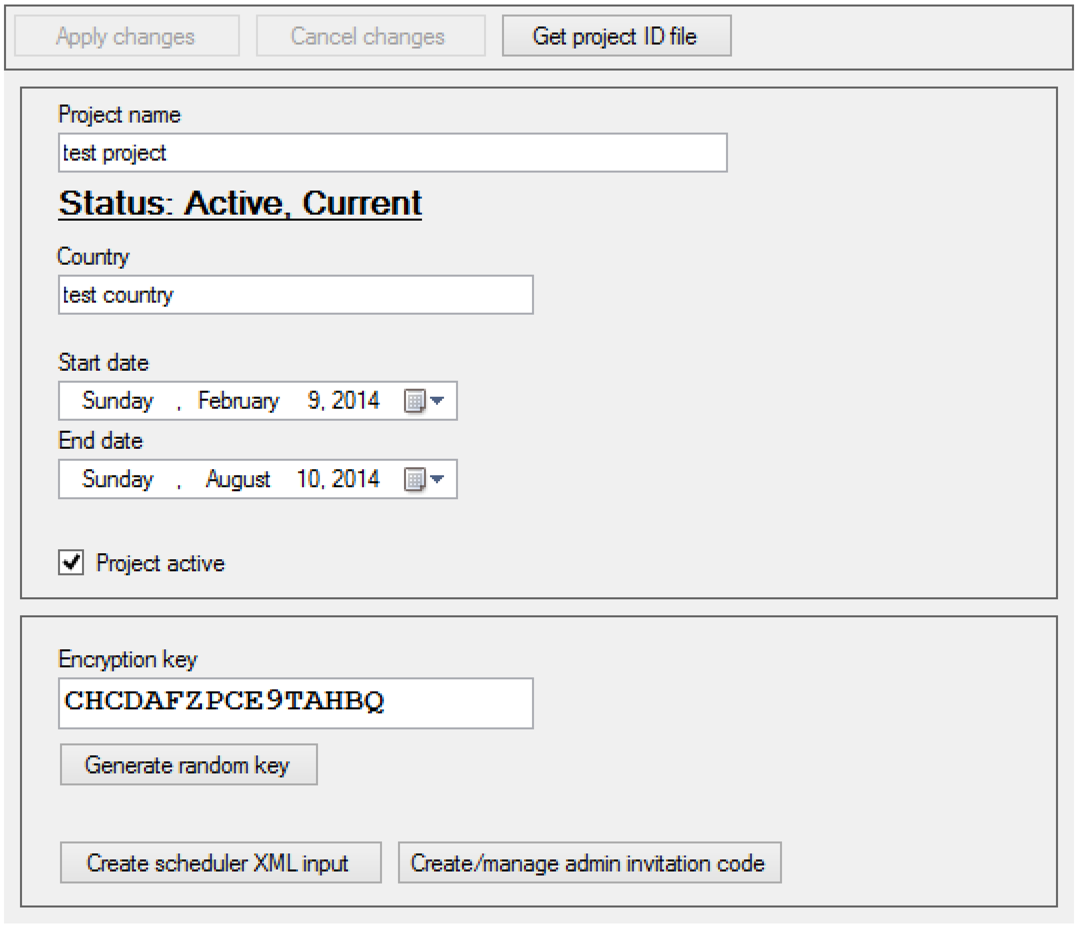
The tree consists of a root **Project** node and the following four child nodes, each representing a corresponding project administration panel:

- **Jobs**
- **Folder configuration**
- **Users**
- **Scripts**

The **Return to project selection** and **Log out** buttons allow the user to return to the project selection screen and to the login screen, respectively. Only the **Log out** button will be visible for non-administrators. Project users do not have access to the project selection screen because the project user account is only attached to one project.

The root project panel contains a form that allows project and system administrators to change global project properties as shown in **Figure 6.12**.

Figure 6.12. Project global properties



Project name: test project

Status: Active, Current

Country: test country

Start date: Sunday , February 9, 2014

End date: Sunday , August 10, 2014

☑ Project active

Encryption key: CHCDAFZPCE9TAHBQ

Generate random key
Create scheduler XML input
Create/manage admin invitation code

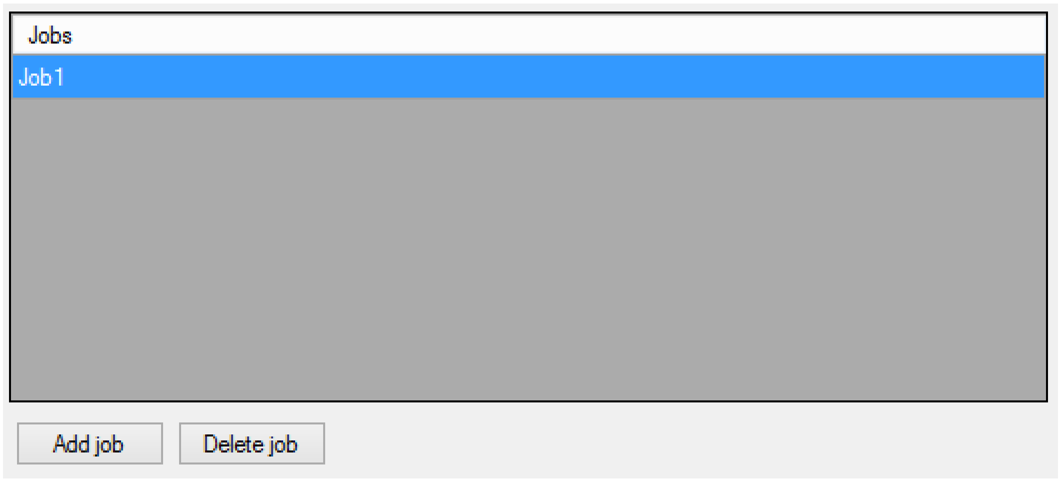
On this screen system administrators can update the **project name** and **country** as well as set the **start** and **end dates** for the project. The top panel is only accessible to system administrators. The lower panel is accessible to both system and project administrators that allows the administrators to change the project **encryption key**, to **create scheduler XML input** and to **create/manage administrator invitation codes**.

In the event of any project property change, the **Apply changes** and **Cancel changes** buttons are activated. Clicking on the **Apply changes** button will update the project metadata on the server, while **Cancel changes** will reset the UI of the project to the previously saved state. The **Get project ID file** button generates project identifier file that is used by the project users to connect to the project.

The project encryption key is a 128 bit string value. It is recommended to automatically generate encryption keys instead of manually assigning them. A random encryption key is generated for the project when this project is first created by system administrators. This key can be regenerated by clicking on the **Generate random key** button.

The **Jobs** panel allows administrators to define jobs in the project (**Figure 6.13**).

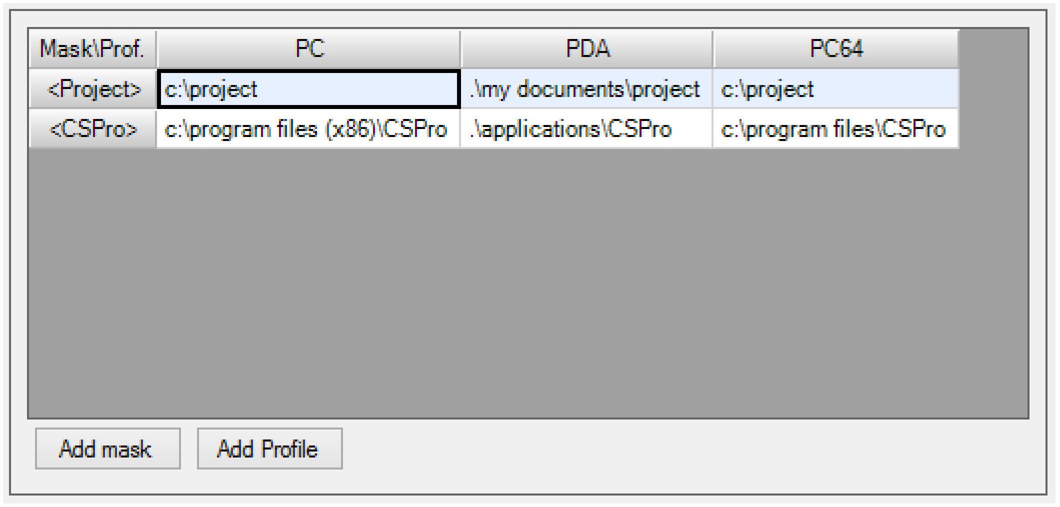
Figure 6.13. Jobs panel



The **job** titles cannot be duplicated within the same project. Clicking on the **Add job** button allows project administrators to create new jobs. The **Delete job** button removes the selected job from the project.

The folder configuration panel is used for defining dynamic folder profiles in the project. It contains a table, where column headers represent user folder profiles, row headers are masks for dynamic folders and the body of the table are values for every mask and profile. **Figure 6.14** shows an example of a folder configuration panel with two folder masks and three folder profiles defined.

Figure 6.14. Folder configuration panel

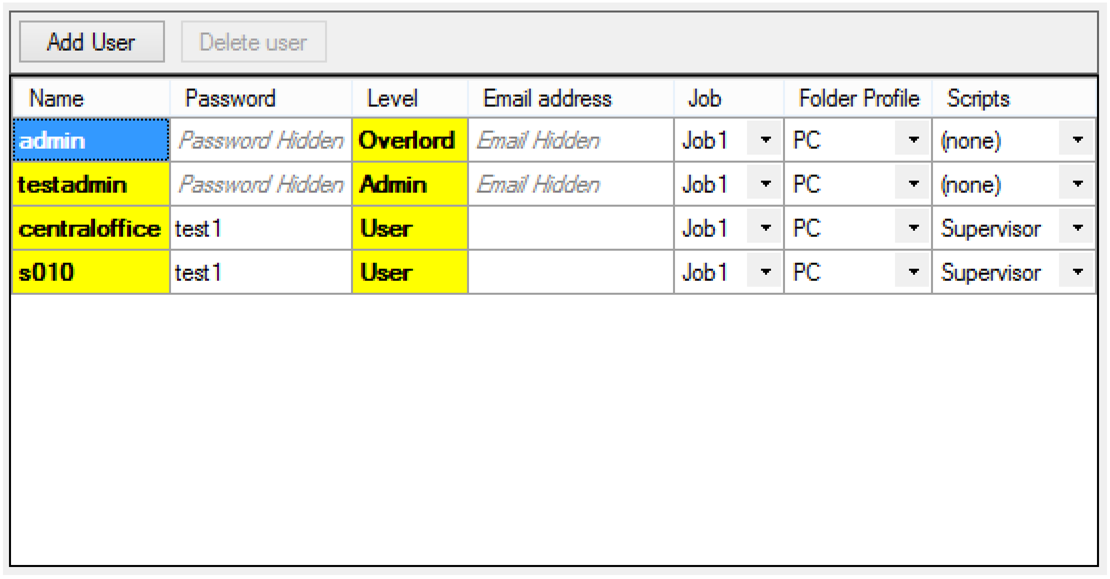


Mask\Prof.	PC	PDA	PC64
<Project>	c:\project	.\my documents\project	c:\project
<CSPro>	c:\program files (x86)\CSPro	.\applications\CSPro	c:\program files\CSPro

In the example above the project has three folder profiles defined: **PC**, **PDA,** and **PC64**. These may correspond to a Windows 32 bit laptop, Windows CE mobile device and Windows 64 bit desktop. There are also two folder masks set up: **<Project>** and **<CSPro>**. These may be used to point to the main project directory and CSPro installation directory. For each mask there is a value in the table corresponding to the actual physical location of the dynamic folder for each profile. For instance, a **<Project>** directory for users running Windows desktops the value will be "C:\project," while for users running Windows Mobile devices the same dynamic folder **<Project>** will be translated into physical path ".\my documents\project". Each user has a folder profile associated with their account. This association will determine which of the values for physical path need to be used when the client application detects a dynamic folder mask such as "**<Project>"** in the portion of the file path. For example, if a project administrator defines a script that is supposed to package files in the path "<Project>\data\*.*", when this script is executed by the Automated Client application by the user running a Windows desktop, the path will be interpreted as "c:\Project\data\*.*", while the same path on a Windows Mobile device will be interpreted by the client as ".\My Documents\data\*.*." It is possible to combine dynamic folder masks in one path, i.e. the following expression would be valid if **<Project>** and **<Work>** masks are set up in the project: "<Project>\<Work>*.*."

The user panel allows administrators to control user accounts in the project (**Figure 6.15**).

Figure 6.15. User panel

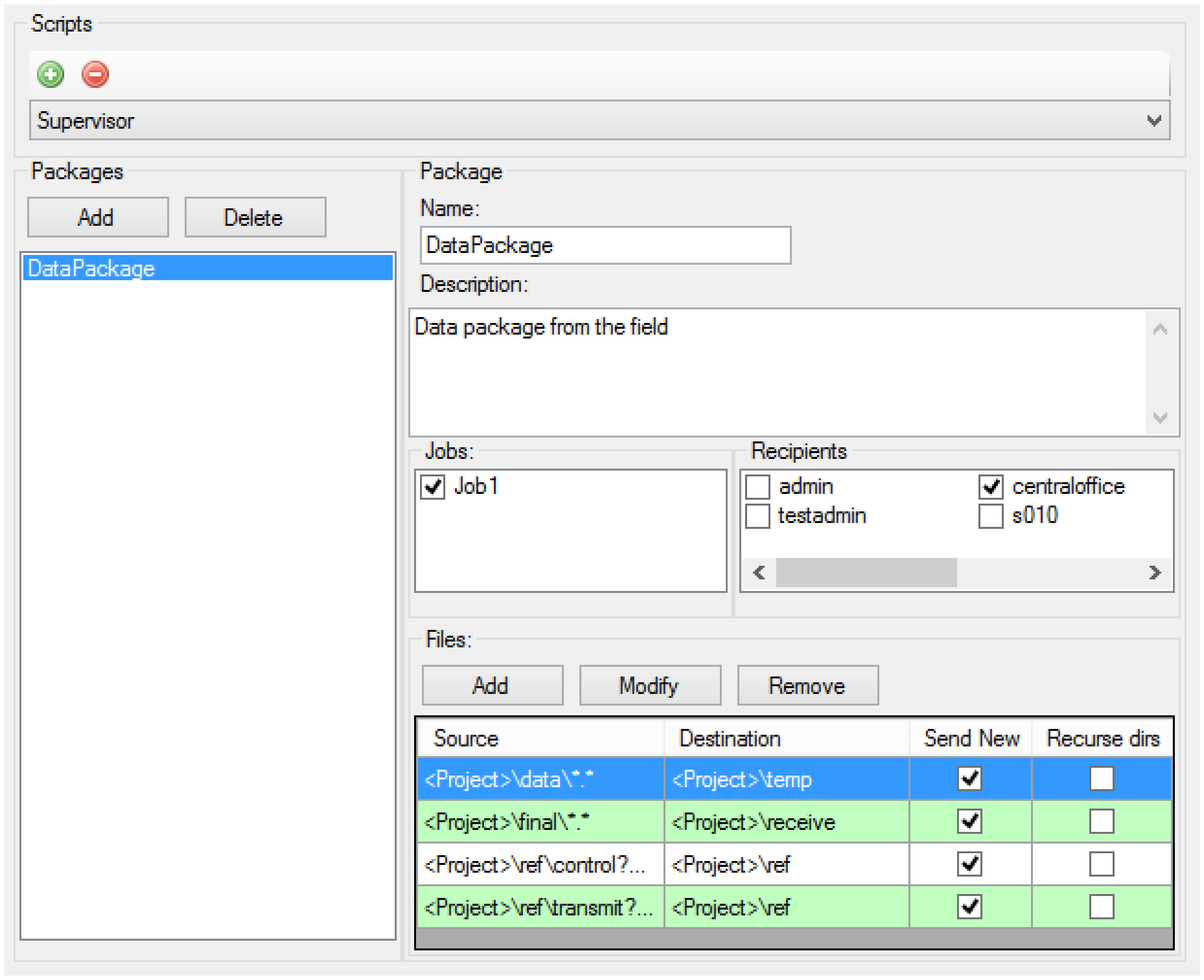


Name	Password	Level	Email address	Job	Folder Profile	Scripts
admin	Password Hidden	Overlord	Email Hidden	Job1	PC	(none)
testadmin	Password Hidden	Admin	Email Hidden	Job1	PC	(none)
centraloffice	test1	User		Job1	PC	Supervisor
s010	test1	User		Job1	PC	Supervisor

In the example above, the project has four users defined: one system administrator (overlord), one project administrator (admin) and two project users (users). On this panel new users can be added and existing users can be deleted from the project. It is important to note that project administrators cannot remove other administrators from the project. As there is no such concept as "project owning user." Administrators can only choose to disassociate themselves from the project independently from other administrators. Here administrators can set the **e-mail addresses**, **jobs**, dynamic **folder profiles** and **scripts** for each user.

The last panel in the project administration tree is **Scripts**. This panel is for defining XML and Flex scripts attached to users in the project. Figure 6.16 shows an XML script definition UI.

Figure 6.16. XML script definition interface



Scripts

Supervisor

Packages

DataPackage

Package

Name: DataPackage

Description: Data package from the field

Jobs: [✓ Job1]

Recipients: admin, [✓ centraloffice], testadmin, s010

Files:

Source	Destination	Send New	Recurse dirs
<Project>\data\*.*	<Project>\temp	✓	
<Project>\final\*.*	<Project>\receive	✓	
<Project>\ref\control?...	<Project>\ref	✓	
<Project>\ref\transmit?...	<Project>\ref	✓	

At the top of the panel there are the buttons for **adding** and **removing** scripts (⊕ and ⊖) and a dropdown list containing all currently defined scripts in the project. In the example abovewe have a script named "**Supervisor**". When executed by Automated Client of the user which has this script attached to their account, it will create a data package with files specified in the **Files** table and **destination** user "centraoffice." it is important to note that it is possible to add multiple data packages into one script.

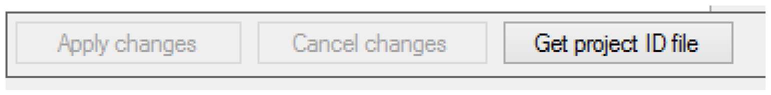
Flex logic-based scripts allow defining scripts using Flex programming language, instead of through UI as XML markup-based scripts.

PROJECT ID FILE

The IFSS cloud can contain multiple virtual projects as well as be hosted on different physical Web servers. For that reason, apart from supplying a user name and password, client applications need to be able to identify what server and what project the users are trying to log into. For that purpose, a specialized encrypted file called "Project ID file" needs to be created by a system or project administrator and distributed to all project users, who intend to have access to data transfer capabilities within the virtual project. The project ID file contains information needed by the client applications to connect to the correct server and project within the IFSS cloud.

In order to create a project ID file, users have access to the **Get project ID file** button in the main project administration strip in the Control Center Client application (**Figure 6.17**).

Figure 6.17. Get project ID file button



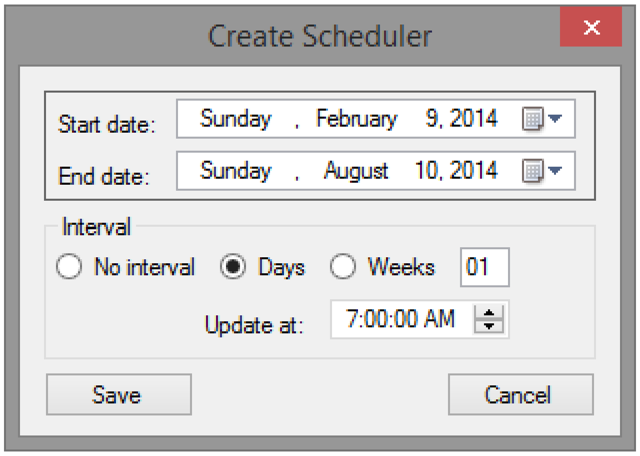
When created, the default file name of the project ID file is “Project.ifss.” When present in the same directory as the client application (both Control Center Client and Automated Client), it is automatically detected. However, if the file is present in a folder different from the client application executable or has another name than “Project.ifss,” it can be supplied to the client application as a command line parameter. For Control Center Client, the parameter is just a path to the project ID file, while for Automated Client the parameter syntax is “projectid=*filename*”, where “*filename*” is a path to the project ID file.

CREATING SEND SCHEDULER XML FILE

When the Automated Client application is set up to run transparently to a user, sometimes there is a need to schedule these runs at certain intervals of time. For instance, the user might log in several times a day, however, a scheduled data transfer is only needed once per day. For that purpose the IFSS system has a mechanism to schedule automated client runs. A specialized scheduler file is supplied to Automated Client, which tracks the time interval since previous successful login, and if the set interval is not reached, the client will terminate before even attempting to log in.

Scheduler files are created by system or project administrators in Control Center Client on the global project properties panel. Clicking on **Create scheduler XML input** button (**Figure 6.12**) produces the dialog window shown in **Figure 6.18**):

Figure 6.18. Scheduler file creation dialog window



Start date: Sunday , February 9, 2014
End date: Sunday , August 10, 2014
Interval: Days, 01
Update at: 7:00:00 AM

Start and **end dates** specify the active period. Usually this corresponds to a project start and end date. But it is possible to narrow the active period down. The **interval** panel allows the user to set up the scheduled intervals of time, during which Automated Client will not attempt to log into the system after a first successful login within the same period. In the example above the interval is set to one day, and the update time is at 7 a.m. In this scenario Automated Client detected a successful login during the day, even if that the day had already started, it will not attempt to log in again until the following morning at 7 a.m.

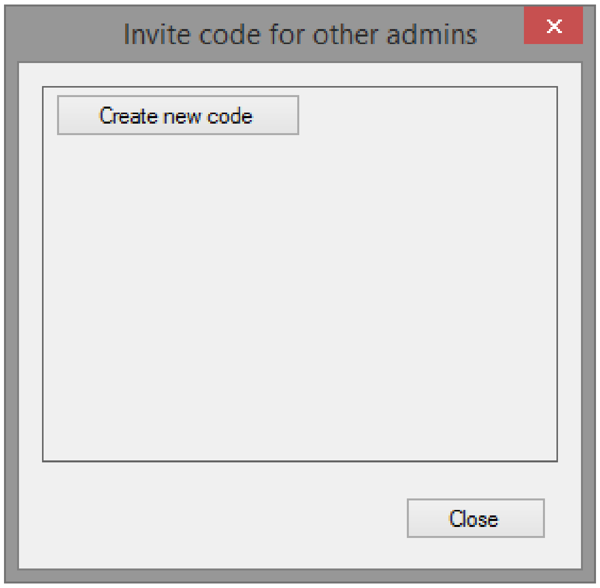
Once the **Save** button had been clicked on and the destination folder has been specified, scheduler file is saved as "scheduler.ifss." Similar to the project ID file, Automated Client detects the scheduler automatically whether it is present in the same folder as the client application executable. If the scheduler is in a different folder or with a different name, it can be supplied to the client via the command line parameter "scheduler=*filename,*" where "*filename*" is a path to the scheduler file. Unlike the project ID file, the scheduler file is optional. If it is not present, the client application will run every time.

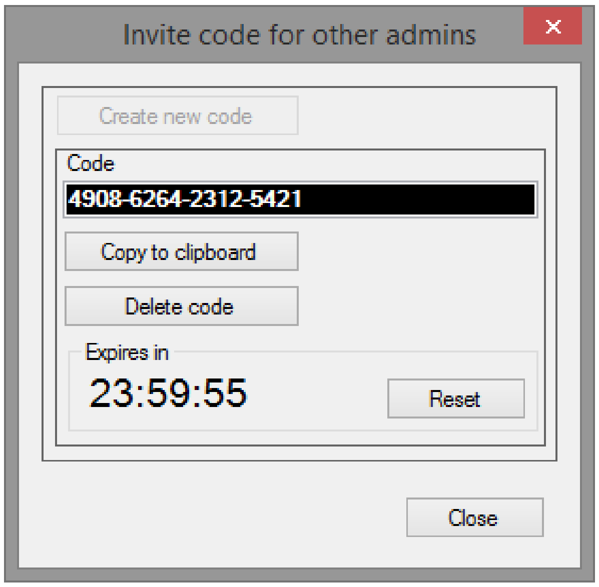
CREATING ADMIN INVITATION CODE

As it was mentioned earlier, administrator accounts are not created under any particular projects, and can, therefore, participate in multiple projects unlike project user accounts. Existing administrators are "invited" to the project by administrators who already participate in that project. To invite another non-participating administrator a special code can be generated and supplied to that administrator through external means such as e-mail or even verbally.

To create an administrator invitation code, currently participating administrator has access to the **Create/manage admin invitation code** button in the Control Center Client application on the global project properties panel (**Figure 6.12**). When clicked on, the system checks for active invitation codes created by this administrator in the last 24 hours and if such code is present, it displays the code and timer to this code expiration. If such code is not present, the dialog window allows the administrator to create a new one. The code is represented by a randomly generated 16 digits numeric value. **Figure 6.19** shows the administrator invitation dialog window before and after the invitation code has been created.

Figure 6.19. Administrator invitation code dialog windows (before and after code creation)





4908-6264-2312-5421

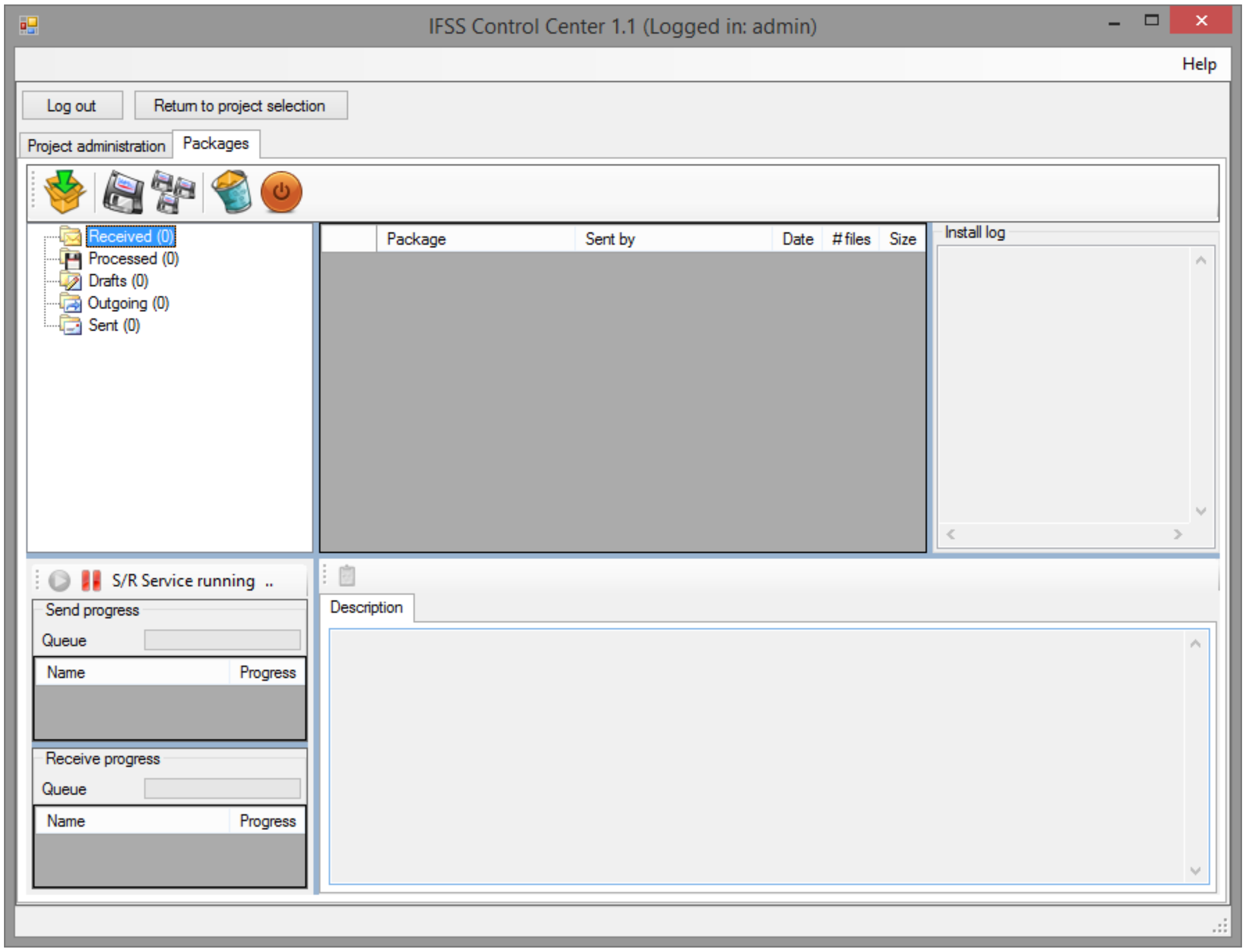
23:59:55

After the code is created, it remains valid for 24 hours. Only one invitation code can be active at a time per participating administrator account. Once created, the code can be shared with other non-participating administrators so they can use it to attach this project to their accounts. If the code is expired, the administrator can create a new one. It is also possible to delete the active invitation code or reset the expiration timer to 24 hours again.

SENDING AND RECEIVING PACKAGES IN CONTROL CENTER CLIENT APPLICATION

In the Control Center Client application, apart from the project administration screen, users have access to the packages send/receive screen (**Figure 6.20**):

Figure 6.20. Package send and receive screen

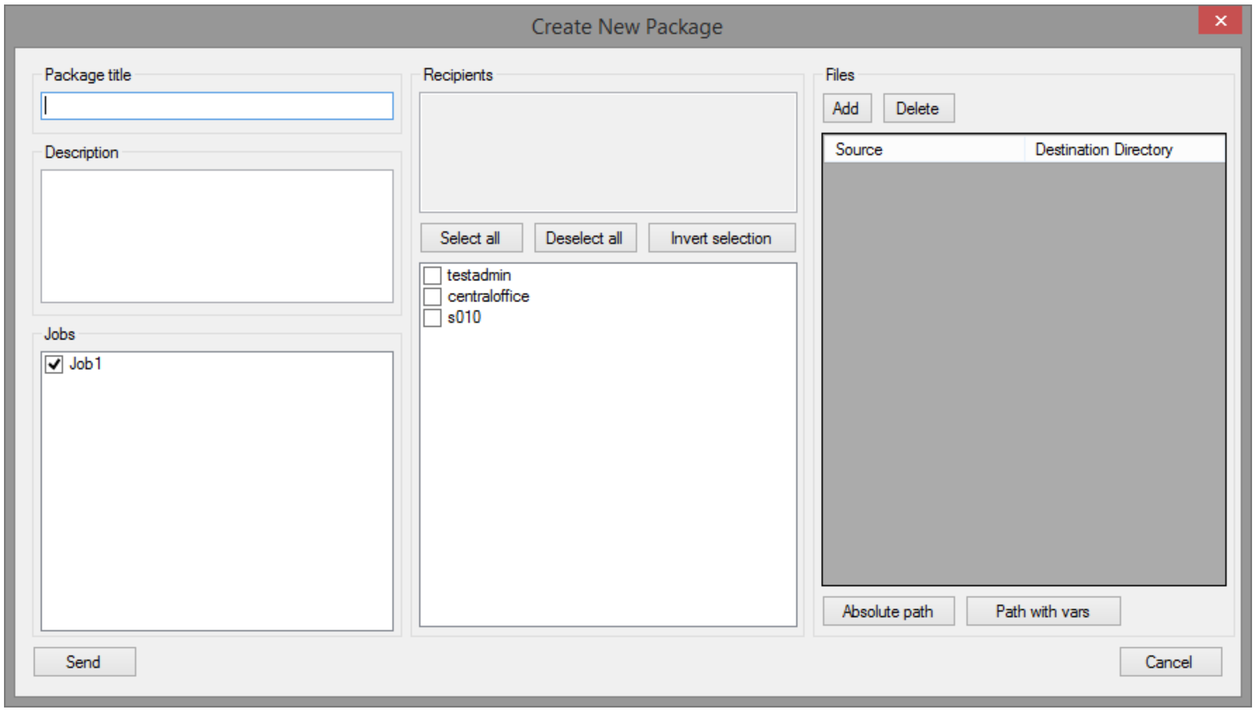


On this screen can create new packages and send them to other users in the project as well as receive and install packages sent by other users. The UI works similarly to e-mail, where on the left-hand side of the screen there is a list of folders with cached packages, a list of packages, and a package header information in the center. When the package is first received, it appears in the **Received** folder. Administrators have to clear the package for installation by clicking on the **Install package** or **Install all pending packages** buttons. The client automatically installs the packages for project users. Once installed, they are moved in the cache from the **Received** to **Processed** folder.

It is important to note that the order of package installation is tightly controlled by the system. Even administrators cannot choose to install packages out of order. Instead they can either choose to install the next pending package or discard it.

To create a new package, the user must click on the **New package** button on the buttons strip at the top of the screen. Once clicked on, it produces the following dialog window (**Figure 6.21**):

Figure 6.21. Create new package dialog window



On this screen users must supply a package name and optional package description. Then they need to specify jobs for installation and package recipients in the project, followed by adding files to the package. The source path is always absolute and must have no wildcard characters, while the destination directory can contain dynamic folder masks. This is different from the XML markup-based scripts, where both the source and destination can contain dynamic folder masks, and the source can also have wildcard characters. The difference is that the script is created by administrators to run on someone else's device, while this package is originating from the same device where it is being created. In the first case you are instructing some other device to package the files in the future through a script, while in this case you are actually creating the package manually at the current moment in time.

Once the form is filled in and the **Send** button is clicked on, the package is created and deposited into the **Drafts** folder (**Figure 6.20**). The client then tries to connect to the server and to start sending the package. While the package is being sent, it is moved to the Outgoing folder. And after the sending is completed, the package is transferred to the **Sent** folder, where it will remain until the project is either reset or deleted.

It is important to note that if the Internet connection is lost while the package is being sent or received, the system will not discard the package, but rather cache a portion of it until the Internet connection is restored. Once the Internet connection is active again, the system will restore the transaction from the point where it was interrupted. This is true for both client applications. This feature is helpful in areas with unstable internet connections.

The **View package log** button shows a dialog window with the server log for the currently selected package (**Figure 6.20**). It shows the date/time information on when the package was received by the server and all recipients. Any package in the cache can be tracked this way.

AUTOMATED CLIENT APPLICATION

As mentioned above, Control Center Client is mostly reserved for project and system administrators. And while it is possible for project users to use it as well, it is highly recommended to deploy the Automated Client application for them instead. As the name suggests, this client, unlike Control Center Client can be configured to perform all necessary data transactions automatically through scripts. This greatly simplifies the process, and, therefore, reduces the risk of errors for the users. Essentially, all that users need to do in order to use the automated client is to log in with their user name and password once. After that, the client remembers their credentials and data transaction becomes a one click affair.

There are two sides to configuring Automated Client for project users: server side and client side. The server side is related to creating scripts and associating them with users. The client side involves creating input files such as the project ID files and optional scheduler files, and providing instructions to Automated Client through command line parameters.

INPUT FILES

There are two types of input files for the automated client application: (1) project ID file and (2) scheduler file. The project ID file is required as the Automated Client application is only used to log into a specific project unlike Control Center Client, which can be used for management of multiple projects by the same administrator. Therefore, Automated Client needs to have access to information about the server and the project on that server. This information is contained within the project ID file. By default the project ID file has the name “Project.ifss” and, if present, in the same folder as Automated Client executable, it is detected by the client automatically. If the file is not in the same folder or named differently, it can be supplied to Automated Client through a command line parameter.

Scheduler files are used to schedule Automated Client execution at predefined time intervals. This file is optional. The default name for the file is “Scheduler.ifss.” Similar to the project ID file, if present in the same directory as the client executable, it is picked up by the client automatically. It can also be supplied through a command line parameter.

COMMAND LINE PARAMETERS

There are several command line parameters that can be supplied to the automated client application to modify its default behavior as summarized in **Table 6.1**.

Table 6.1. Commands line parameters for the Automated Client application

Parameter	*Description*
projectid=filename	Supplies the project ID file to the client application
scheduler=filename	Supplies the scheduler file to the client application
Resetsentfiles	Resets the information in the cache about the files that have been previously sent by the client in the project
Resetcookie	Resets the login cookie before logging in
resetlastmarker	Resets the time marker for the last successful login, so the scheduler would not detect the actual time interval since the last login. This is useful for circumventing the scheduler without deleting the scheduler file.
quickexit	If present in the command line, opts out the client to display the "Work completed" message at the end of the transaction
block=blockname	Specifies the block to run in the Flex logic-based script. Default block is "main()."

FOUR STEPS OF SEND/RECEIVE DATA PROCESS

Automated Client goes through the following four steps when run:

1. Log in
2. Run script
3. Send/Receive package
4. Install received package

First step is logging in. The client connects to the IFSS cloud and verifies that the supplied project ID information and login credentials are valid.

After a successful login, if a script is associated with the user account, the client attempts to execute it. If the script contains instructions to create new packages, they are created automatically and put into the outgoing folder in the cache.

Third step is the send/receive procedure. All the outgoing packages that are queued in the cache are transmitted to the server, and all the packages on the server, where the specified recipient is a logged in user, are downloaded.

Once there are no more pending packages for send or receive, the client proceeds to the final step, which is installation of received packages. Installation involves extracting files from the package and copying them in their destination directories. If the destination directory does not exist, it is created automatically. If the file with the same file name exists in the destination directory, it is overwritten by the received file.

APPENDIX A: FLEX LOGIC-BASED SCRIPT FUNCTIONS

Flex logic-based script common functions:

Function	Description
SETVAR("name", value);	Sets a global script variable
SETTHREADVAR("name", value)	Sets a variable only accessible in the current thread
GETPARAM(paramIndex, "varName");	Sets a value of the block parameter with index "paramIndex" into the variable "varName"
GETPARAMCOUNT("varName");	Gets the number of parameters in the block and sets the value of "varName" to this number
RUNBLOCK("blockName");	Starts a new thread and executes block "blockName" on that thread
GETBLOCKTREADCOUNT("blockName", "varName");	Gets the number of thread that are currently executing block "blockName" and set the value of "varName" to this number
STOPBLOCK("blockName");	Stops the last thread in the queue executing block "blockName"
STOPBLOCKALL("blockName");	Stops all threads executing block "blockName"
WAITFORBLOCK("blockName");	Suspends execution on the current thread, until all threads that are running bock "blockName" are completed
IF(condition);	Conditional function. If "condition" is true, executes all function following IF() function until ELSEIF(), ELSE() or ENDIF() is hit. Otherwise skips to next ELSEIF() or ELSE().
ELSEIF(condition);	Conditional function. See IF() function.
ELSE();	Conditional function. Alternative to IF() and ELSEIF() functions.
ENDIF();	Conditional terminator function
WHILE(condition);	While loop function
ENDWHILE();	While loop terminator function
SLEEP(milliseconds);	Suspends current thread for "milliseconds" time

IFSS custom Flex functions:

Function	**Description**
SETPROGRESSBARMIN(value);	Sets the minimum value of the progress bar
SETPROGRESSBARMAX(value);	Set the maximum value of the progress bar
SETPROGRESSBARVAL(value);	Sets the progress bar value
CLEARTEXT();	Clears all text from the main text output panel
WRITELINE(text);	Writes a line of text on the main output panel
GETFILESNUM("var", dir, mask);	Gets the number of files in directory "dir" with mask "mask" and assigns this number to the value of "var"
CREATEPACKAGE("name", ["descr", "job"*]);	Creates a package header with name "name" and descriptions "descr," which will install for jobs "job"
ADDRECIPIENT("name");	Adds recipient with name "name" to the currently open package
ADDFILE(index, dir, mask, "var");	Adds a file to the currently open package with index "index" from the list of files in directory "dir" with mask "mask". "Var" variable receives the value of the filename that's been added.
SAVEPACKAGE();	Saves the currently open package to the outgoing folder in the cache
READFILEMETABASE();	Reads the database of the previously sent files from the cache
SAVEFILESMETABASE();	Updates the database of the previously sent files with the files that's been added to the package.