
# ViewDS Identity Bridge: Getting Started

Published: September 2024
Version: 1.4

**ViewDS Identity Bridge: Getting Started**

For Identity Bridge Release 1.4

September 2024

**Document Lifecycle**

ViewDS may occasionally update documentation between software releases. Therefore, please visit www.viewds.com to ensure you have the PDF with most recent publication date. The site also hosts the most recent version of this document in HTML format.



ABN 19 092 422 47

# Contents

**Introduction** ..... **1**
What is Identity Bridge ..... 1
Identity Bridge projects ..... 2
Supported resources ..... 3
Summary of key concepts ..... 4
**Installing Identity Bridge** ..... **5**
System requirements ..... 5
Installing Identity Bridge ..... 6
Installing a license key ..... 7
Upgrading Identity Bridge ..... 8
**Investigating the application** ..... **9**
Application overview ..... 9
Workspace Explorer ..... 10
Main work area ..... 11
**Tutorial** ..... **15**
Select the Workspace ..... 15
Create the project ..... 16
Create the source resource ..... 16
Create the target resource ..... 17
Create the task ..... 18
Define the task ..... 18
Run the project ..... 20
Apply functions ..... 21
Distribute the project ..... 25
**Advanced options** ..... **27**
Project variables ..... 27
Variable sets and function libraries ..... 28

Delta updates ..... 32
Change logs ..... 40
JavaScript connector ..... 41

# Introduction

This guide provides the information you need to install, configure and use Identity Bridge.

Further instructions for specific tasks can be found in the Identity Bridge Studio help. Additionally, the release notes include information relating to this release.

This section includes:

- What is Identity Bridge
- Identity Bridge projects
- Supported resources
- Summary of key concepts

## What is Identity Bridge

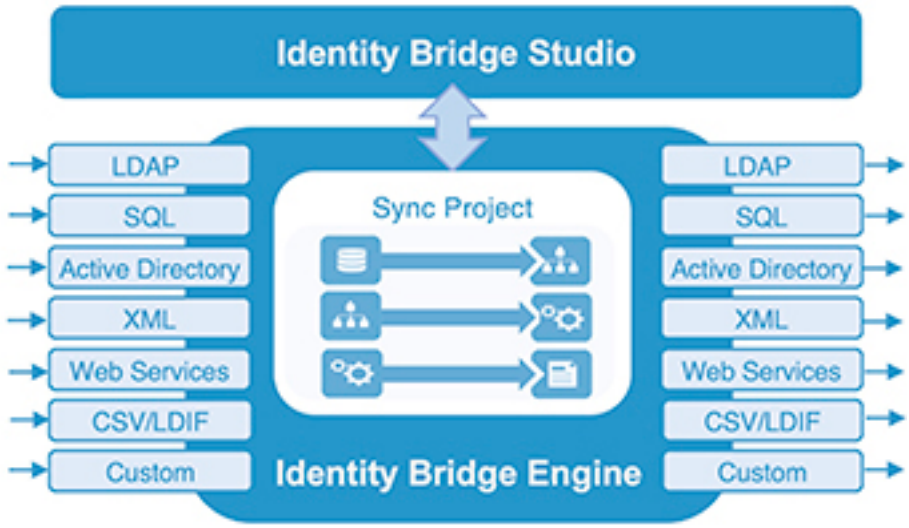
Identity Bridge is a highly flexible synchronization tool. It allows you to synchronize, migrate and provision identities across all your enterprise systems and cloud applications.

Identity Bridge has two platform-independent components:

- Identity Bridge Studio
- Identity Bridge Engine

Identity Bridge Studio is a user interface that allows you to define and maintain synchronization projects. These projects define the data sources and targets to be synchronized, plus the data processing tasks to occur during sychronization.

Identity Bridge Engine executes your synchronization projects.



# Identity Bridge projects

An Identity Bridge project is a collection of *resources* and *tasks*. A *resource* might be, for example, a database, directory or flat file. A *task* moves and transforms data between resources in order to fulfil a business need, such as provisioning new users.

Projects are defined and maintained through Identity Bridge Studio.

Each project has three main components:

- Source resource
- Target resource
- Task

The source and target can be any combination of the supported resource types.

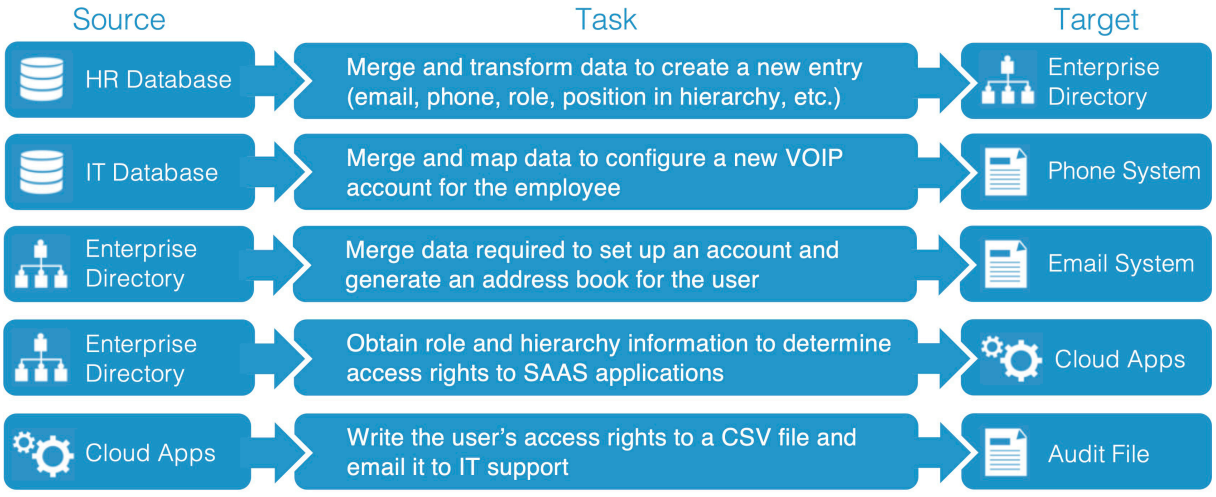
The task is the fundamental building block of a synchronization project. It allows you to apply JavaScript functions that, for example, impose business rules and transformations to the source data before it is written to the target. Identity Bridge includes a JavaScript library and editor so that you can write and maintain functions as required.

A project can include a sequence of tasks, each with a single source and target.

## Example: Employee provisioning

When a new employee joins an organization, there's an imperative to get them up and running as quickly as possible. This usually involves granting access to a range of resources and business systems.

The following illustration shows an example project comprising tasks that collectively provision a new employee.

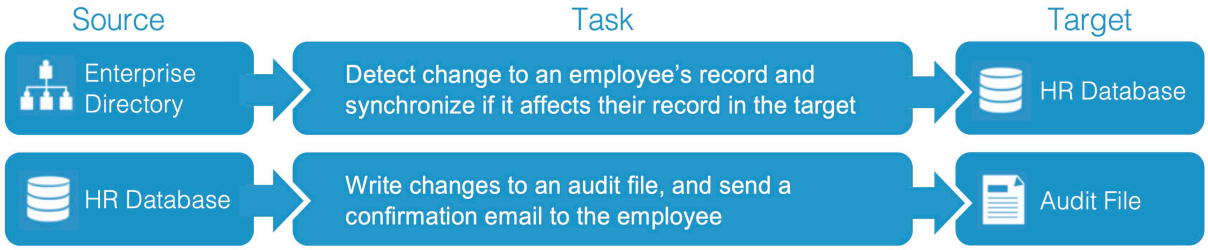



The workflow starts automatically when a new employee is added to the HR database. The consequent tasks extract, transform and load data into each business system required to provision the employee. Similarly, a workflow to deprovision could be triggered when someone is flagged as an 'ex-employee'.

As an extension of the above, consider the scenario where a provisioned employee wants to update their personal details in an Enterprise Directory, such as *ViewDS Directory* .

The user can make an update through the directory's self-service portal, which ensures that the task of maintaining personal data is in the hands of the person most qualified to do so. However, a major consideration is that these updates will usually need to be distributed to other business systems.

An Identity Bridge project to overcome this issue would automatically detect a change to an employee's record and then perform synchronization tasks with all relevant systems.




## Developing and executing projects

While Identity Bridge Studio allows you to define and maintain projects, the Identity Bridge Engine allows you to execute individual tasks and projects.

For testing purposes, you can run Identity Bridge Engine from within Identity Bridge Studio. However, in a production environment, project files are typically distributed and then executed by the engine as scheduled tasks (for example, using Linux *cron* or Windows Scheduler).

# Supported resources

Identity Bridge connects to a resource through a *connector plugin*.

The range of connector plugins supplied by Identity Bridge includes the following:

- Generic LDAP and Active Directory
- Flat File
- LDIF
- SQL
- WebAPI (WsSimple)
- JavaScript
- Delta Database

Implementing a *Delta Database* allows you to define tasks that only process new and modified source records before writing them to a target. This improves performance and also reduces the load on the target system. Tasks can also be defined to identify the records that have been deleted from a source, and then delete the corresponding records stored by the target.

Additionally, there are change-log *connector plugins* for generic LDAP and Active Directory, which provide the same functionality as a Delta Database.

Custom plugins can also be implemented.

# Summary of key concepts

The following terms have special meaning in Identity Bridge.

### Workspace

The file-system directory where projects are stored by Identity Bridge Studio.

### Project

An ordered collection of one or more tasks.

Identity Bridge Studio writes each project to a separate XML file, which can then be distributed and run by the Identity Bridge Engine.

Typically, the engine runs an entire project. Although it is also possible to instruct the engine to run individual tasks, or to disable specific tasks, within a project.

### Task

The basic building block of an Identity Bridge project.

Each task has a single source and target. The source and target can be of any combination of resource types supported by Identity Bridge.

### Resource

A data source or target.

Many resources such as LDAP directories and databases can be both the source and target, and appear in multiple projects. Identity Bridge Studio allows you to define resources that can be either specific to a project or as part of the workspace (from where they can be propagated into multiple projects).

### Delta Database

Allows a task to respond to the changes made to a source resource.

Therefore, tasks can be defined that only process new and modified source records before writing them to a target. This improves performance and also reduces the load on the target system. Tasks can also be defined to identify the records that have been deleted from a source, and then delete (or otherwise process) the corresponding records stored by the target.

# Installing Identity Bridge

This section includes the steps to install or upgrade Identity Bridge. It includes:

- System requirements
- Installing Identity Bridge
- Installing a license key
- Upgrading Identity Bridge

## System requirements

### Platform requirements

Identity Bridge is a platform-independent Java application.

A Java Runtime Environment (JRE) version 8 or 11 long-term support (LTS) release must be installed on the same machine as Identity Bridge.

The latest JRE for your platform is available from: www.java.com/getjava/

### Memory requirements

The memory requirements are:

- Identity Bridge Engine requires approximately 128MB of available memory; and
- Identity Bridge Studio requires around 512MB.

### Disk space requirements

Identity Bridge requires approximately 100MB of disk space.

Additional disk space is required for workspace and project files.

### Internal dependencies

The Identity Bridge Engine uses the following versions of third-party libraries:

- Mozilla Rhino 1.7.14 (provides JavaScripting functionality)
- JavaMail API 1.6.2 (provides sendMail functionality)
- JSON.simple 1.1.1 (to encode and decode JSON text)

# Installing Identity Bridge

Identity Bridge's distribution media includes two platform-specific installers, plus a platform-independent file for installing the application on an unsupported platform.

The supported installers are:

- **Linux**: `identitybridge-1.4.0.tar.gz`
- **Windows**: `identitybridge-1.4.0-windows.exe`

The unsupported **platform-independent** file is:

- `identitybridge-1.4.0.zip`

All of the above include both components of the application: Identity Bridge Studio and Identity Bridge Engine. The components cannot be installed individually.

Although ViewDS does not formally support platforms other than Windows and Linux, Identity Bridge can run on other platforms provided the system requirements are satisfied.

## Platform specific

To install Identity Bridge on Windows or Linux:

1. Download the Identity Bridge installer for your platform.
2. Double-click the installer and follow the prompts. The application is installed.
3. Identify a *workspace* directory and create it if it doesn't exist. This is the file-system location where project files will be stored by Identity Bridge Studio. It should usually be a location that is accessible to everyone who will use Identity Bridge.
4. If Identity Bridge Studio is not already running, start the application. You are prompted to select the workspace.
5. Use the **Browse** button to select the workspace directory, or enter its full path in the box.
6. Click **OK**. The workspace is displayed at the top of the Workspace Explorer (the application's left pane).

   Identity Bridge also creates a directory structure below the workspace directory. These directories will contain files relating to Identity Bridge projects.

   ```
   [workspace directory]
     ib_data
       keystore
       lib
       logs
       plugin
       projects
       store
   ```

7. If you plan to use JDBC Resources, copy the JDBC drivers (`*.jar` files) to the `/ib_data/lib` directory in your workspace.
8. Follow the steps to install a license key.

### Platform independent

To install the platform independent .zip file:

1. Download `identitybridge-1.4.zip` and extract the contents to the required directory (the Window installer's default install location is `c:/Program Files/identitybridge`).
2. Follow the above task from step 3.

## Installing a license key

You can use Identity Bridge Studio without a license key, but you will be unable to run any tasks or projects. To obtain an Identity Bridge license key (`license.xml`), contact ViewDS Support through viewds.zendesk.com or by email support@viewds.com.

A license key can be installed either:

- through Identity Bridge Studio, or
- from the command prompt.

### Identity Bridge Studio

To install the license key through Identity Bridge Studio:

1. From the **Tools** menu, click **Options**. The Options window is displayed.
2. At the top of the window, click **Identity Bridge**. The License tab is displayed.
3. Enter the contents of your license-key file into the box.
4. Click **Apply** followed by **OK**. License information is displayed in the title bar.

### Command prompt

To install the licence key from the command prompt:

- Copy the `license.xml` file to the `<install dir>/config` directory (the default Windows `<install dir>` is `c:/Program Files/identitybridge`).

# Upgrading Identity Bridge

**WARNING**: Back up the current workspace directory before installing version 1.4.

**NOTE**: Task variables are redundant in this version of Identity Bridge. During the upgrade to version 1.4, any task variables in your existing projects will become project variables (displayed in the Properties area of the Project tab). The processing of these projects will be unaffected.

To upgrade from a previous version of Identity Bridge:

1. Back up the current workspace directory. Its location is displayed at the top of the Workspace Explorer.
2. Copy the license file `<install dir>/config/license.xml` to a temporary location (the default Windows `<install dir>` is `c:/Program Files/identitybridge`).
3. For Windows, use **Add or remove programs** to uninstall the existing version of **ViewDS Identity Bridge**; or for Linux, use a comparable process.
4. Follow the steps to install Identity Bridge.
5. Reinstate the license key by either:
   - Copying `license.xml` from its temporary location to `<install dir>/config`.
   - Installing the license key through Identity Bridge Studio.

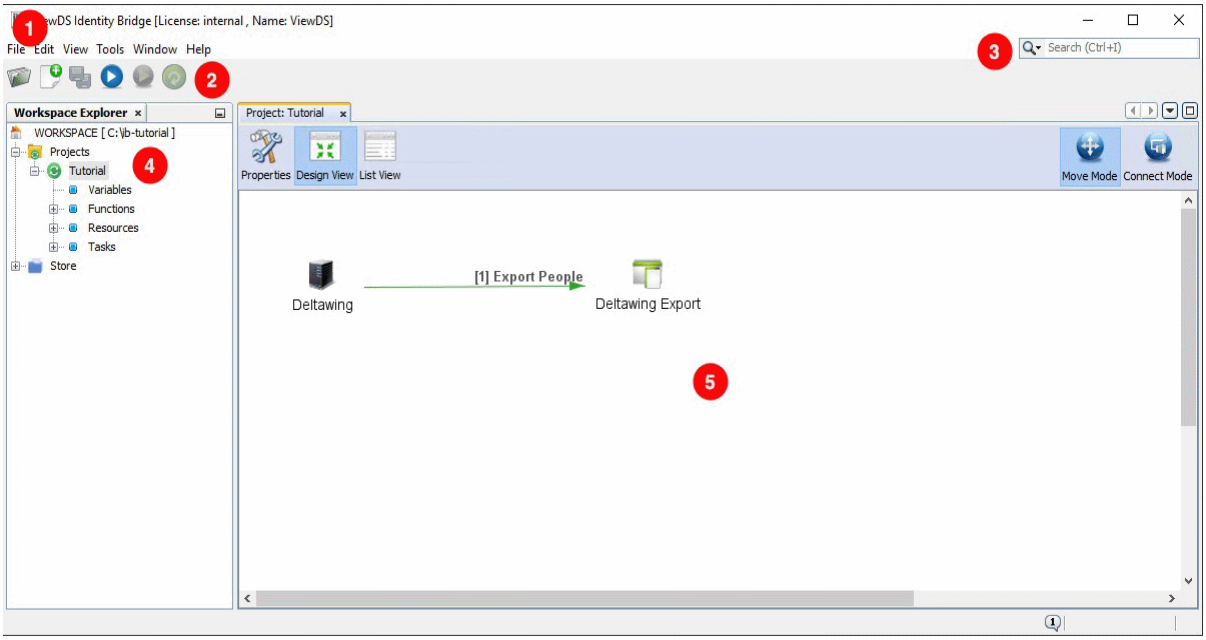
# Investigating the application

This section provides a brief introduction to Identity Bridge Studio. It includes:

- Application overview
- Workspace Explorer
- Main work area

## Application overview

The main areas of the application are numbered below.

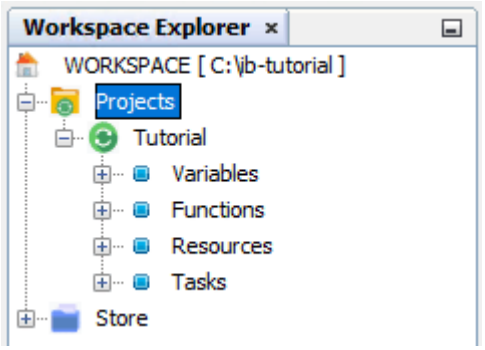



The numbered areas are:

1. Main menu – provides access to commands.
2. Toolbar – provides quick access to commands also available from the main menu.
3. Search box – allows you to search within a range of scopes including actions, symbols, and help topics.
4. Workspace Explorer – provides access to different aspects of projects in the current workspace (more detail below).
5. Main work area – allows you define and modify projects (more detail below).

# Workspace Explorer

The Workspace Explorer provides access to the components of projects in the current workspace ( `c:\ib-tutorial` in the example below).




The Workspace Explorer has two main areas that can be expanded or collapsed:

- Projects
- Store

The Projects area contains the projects in the workspace – in the above example, there is one project called Tutorial. Each project contains the following project-specific components:

- Variables
- Functions
- Resources
- Tasks

The Store contains variable sets and function libraries, which are components that can be used in many different projects. They are described in more detail in the Advanced options section of this guide.

Right-click a component to access a context-sensitive menu.

Double-click a component to view its details in the main work area.

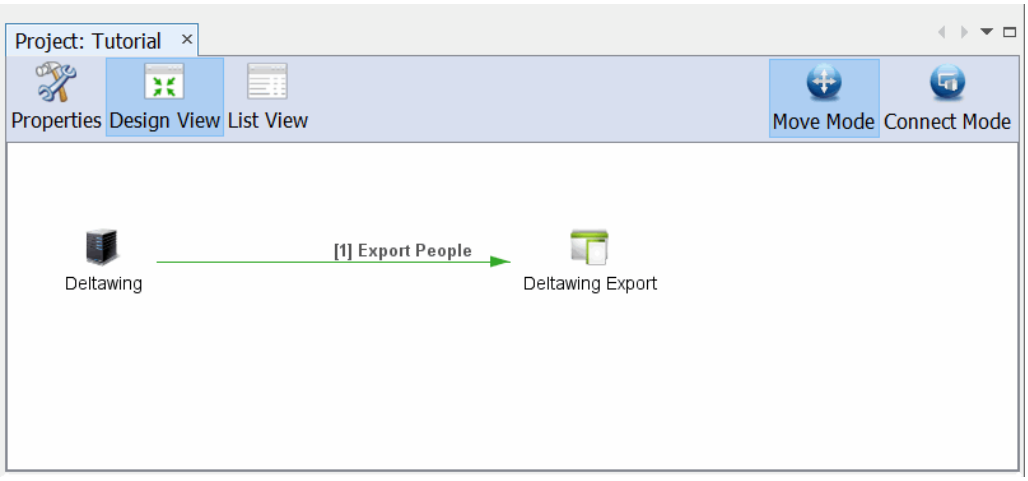
# Main work area

The main work area contains tabs that define different components of an Identity Bridge project. The tabs include:

- Project tab
- Task tab
- JavaScript Function Editor

## Project tab

This tab allows you to define the relationship between the tasks and resources in a project.




The tab has three views:

- Properties
- Design View (default)
- List View

### Properties

Properties provides access to the project's properties including its description and the location of its log file. It also allows you to define variables with a project-wide scope, which can be used to pass values between tasks in the project.

### Design View

Design View provides a graphical representation of a project.

It allows you to build a project that comprises resources, which can be a source or target, and link them with tasks. (Note that a resource can be both a source and target if required.)

Design View has two modes:

- Move Mode – allows you to move a resource to a new location in the work area. You can also add resources and tasks by clicking and dragging them from the Workspace Explorer, and access context-sensitive menus to manage resources and tasks.

- Connect Mode – allows you to create new tasks by clicking the source and dragging to the target resource. (Note that you can drag the connector back to the source resource if you want the source and target to be the same.)

Right-click a component, or in empty space, to access a context-sensitive menu.

Double-click a task arrow to open it in the Task tab.

### List View

List View displays a list of the project's tasks.

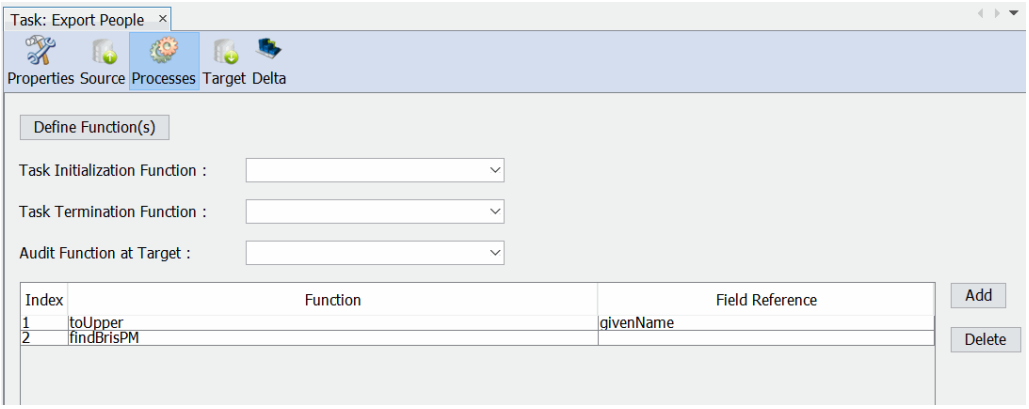
When there are multiple tasks, the list signifies the order in which they will be executed. You can click and drag a task to a new position in the list.

Right-click a task to access a context-sensitive menu (such as 'Run Task').

Double-click a task to open it in the Task tab.

## Task tab

The Task tab allows you to define the details of a task, including the data processing to be applied during synchronization.




It has the following areas:

- Properties
  Allows you to set the general properties of the task including its name and description.
- Source
  Allows you to set parameters that define how to connect to the source resource, and to identify the data to be extracted from the source.

- Processes
  Allows you to assign JavaScript functions that will, for example, process the source data before it is written to the target resource.
- Target
  Allows you to set parameters that define how to connect to the target, and to identify the data to be written to the target.
- Delta
  Allows you to configure a Delta Database to be used by the task. For example, this could be used to trigger the task whenever an entry is modified in the source resource.

## JavaScript Function Editor

The JavaScript Function Editor allows you to write and maintain functions that can be applied by a synchronization task.

Functions can be applied in several contexts, which are available in the Processes area of the Task tab.

Example uses of functions include:

- manipulating data
- handling errors
- conditional processing
- sending notifications
- calling external Java functions
- calling applications
- perform secondary lookups

# Tutorial

This tutorial takes you through the process of defining a simple synchronization project.

The project's source is a directory and its target is a flat file. Its task extracts all the `organizationalPerson` entries from an LDAP directory and writes them to a CSV file.

The tutorial has nine stages:

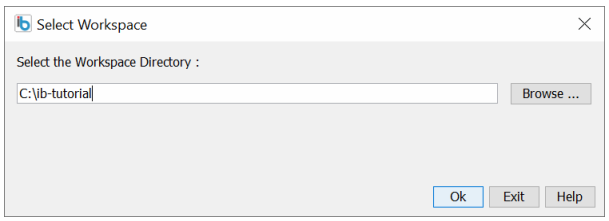
1. Select the Workspace
2. Create the project
3. Create the source resource
4. Create the target resource
5. Create the task
6. Define the task
7. Run the project
8. Apply functions
9. Distribute the project

## Select the Workspace

The Workspace is the location where project files are stored.

To select a Workspace for this tutorial:

1. Create a new directory called `ib-tutorial` in a location of your choice (for example, `c:\ib-tutorial`).
2. From Identity Bridge Studio's **File** menu, click **Select Workspace**. The Select Workspace window is displayed.
3. Click **Browse** and select the Workspace directory, then click **OK**.




   The selected location is displayed at the top of the Workspace Explorer.

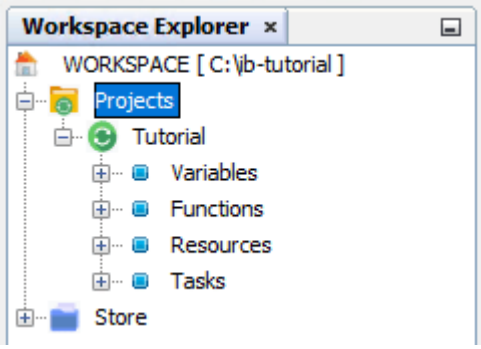
Also note that the application creates a directory structure below the workspace directory. These directories will contain files relating to your projects.

# Create the project

A project is an ordered collection of one or more tasks.

To create a project for the tutorial:

1. From the **File** menu, click **New Project**. The New Project window is displayed.
2. Enter the **Name** `Tutorial` and a brief **Description**.
3. Click **Save**. Your project is added to the Workspace Explorer under Projects.
   Your new project contains Variables, Functions, Resources and Tasks that will be within the project's scope. The components in the Store are global and are available to all projects.




# Create the source resource

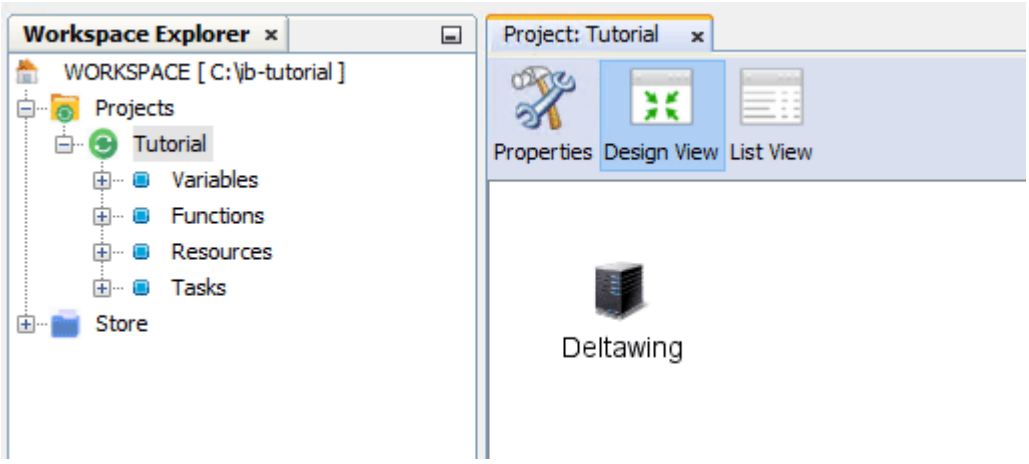
A project comprises one or more synchronization tasks. Each task has a source and target resource, while each resource defines the type of data store and how to access it.

This tutorial's source is a ViewDS demonstration directory called *Deltawing*, which Identity Bridge will connect to through its LDAP connector plugin.

To create the source resource:

1. In the **Workspace Explorer**, double-click the **Tutorial** project. An empty Project tab opens in Design View with Move Mode selected by default.
2. Right-click anywhere in the empty Project tab, then click **New Resource in Project**. The New Resource Wizard is displayed.
3. In the **Resource Name** box, enter `Deltawing`.
4. In the **Connector Plugin** box, select **LDAP**.
5. Click **Next**. The Parameters page is displayed.
6. In **Directory LDAP URL**, enter `ldap://corporate.demo.viewds.com:3006`
7. In **Security Level**, select **Anonymous**.
8. In **Base DN**, enter `o=Deltawing`.
9. Click **Test Connection**. The Connection Status window is displayed.

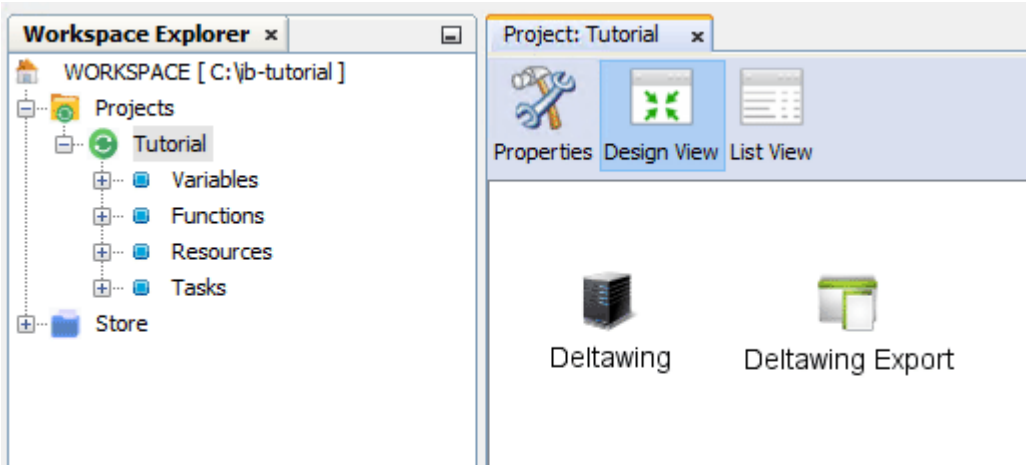
10. Click **Finish**. The resource is displayed in the Project tab, and also below the project's Resources in the Workspace Explorer.




## Create the target resource

To create a flat-file target resource:

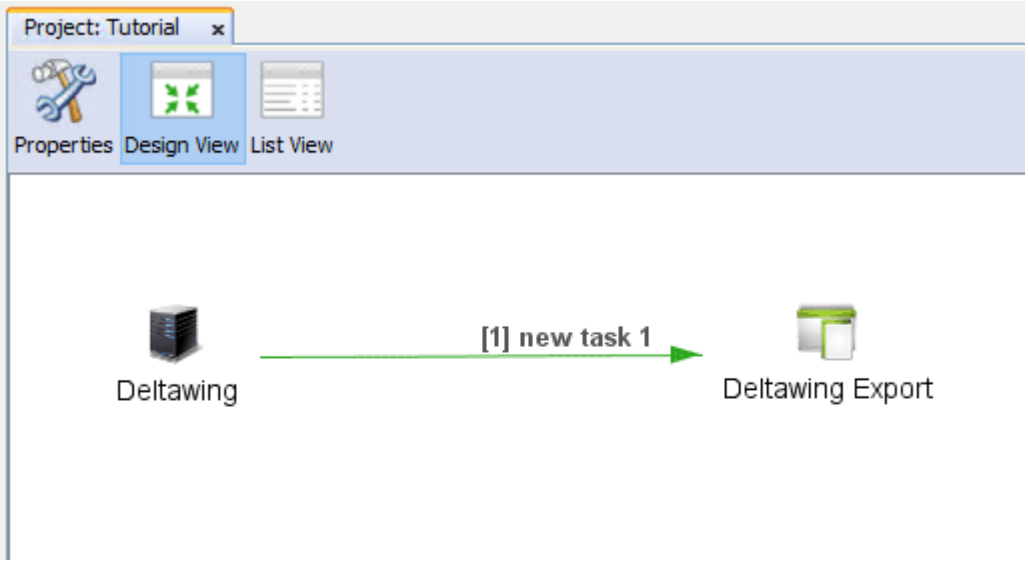
1. Right-click anywhere in the **Project** tab, then click **New Resource in Project**. The New Resource Wizard is displayed.
2. In the **Resource Name** box, enter `Deltawing Export`.
3. In the **Connector Plugin** box, select **Flat File**.
4. Click **Next**. The Parameters page is displayed.
5. In **File** enter `[Workspace]\deltawing_export.csv` where `workspace` is the location you declared in the first stage of this tutorial (for example, `c:\ib-tutorial`) and is displayed at the top of the Workspace Explorer.
6. In the **File Format** box, select **csv**.
7. Click **Finish**. The resource is displayed in the Project tab, and also below the project's Resources in the Workspace Explorer.



# Create the task

To create the synchronization task:

1. In the **Project** tab, click and drag the **Deltawing Export** resource so that there is some space between it and the Deltawing resource.
2. In the top-right corner of the **Project** tab, click **Connect Mode**.
3. Click and drag from the **Deltawing** resource to the **Deltawing Export** resource. A task arrow is displayed between the two resources.



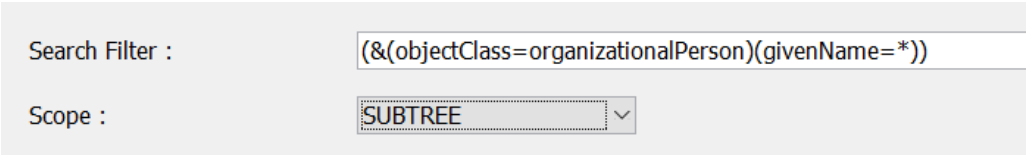

4. Before moving onto the next stage of the tutorial, it is worth taking a little time to get acquainted with the other two areas of the Project tab:
   a. At the top of the tab, click **Properties**. This area allows you to modify project's properties including its description and the location of its log file. It also allows you to define variables that have a project-wide scope.
   b. Click **List View**. This area lists the project's tasks. When there are multiple tasks, the list determines the order in which they will be executed. You can click and drag a task to change its position in the list, and right-click a task to apply commands such as 'Disable Task' and 'Run Task'.
   c. Click **Design View**.

# Define the task

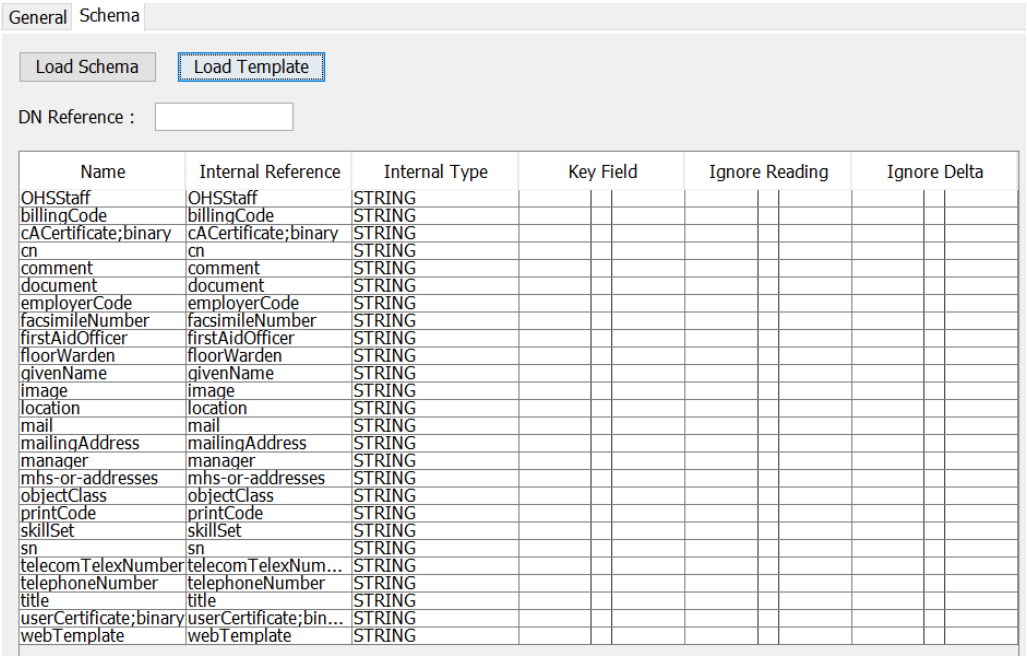
To define the task's source:

1. Double-click the task arrow that connects the Deltawing resource to the Deltawing Export resource. The Task tab is displayed. It contains five areas: Properties, Source, Processes, Target, and Delta.
2. In the **Properties** area, change the default **Task Name** to `Export People`.

3. Enter a **Description** of the task, `Export all organizationalPerson entries to CSV`.
4. At the top of the **Project** tab, click **Source**. This area contains two tabs: General and Schema.
5. In the **Search Filter** box, enter the following (no spaces):
   `(&(objectClass=organizationalPerson)(givenName=*))`
6. In the **Scope** box, select **SUBTREE**.



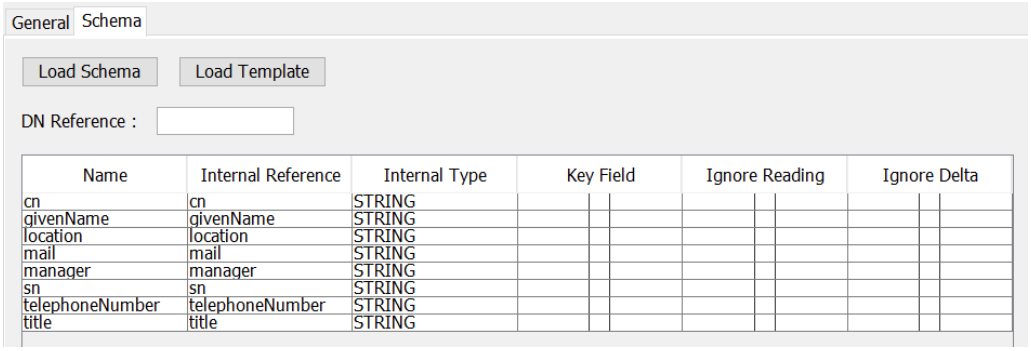

7. At the top of the **Source** area, click the **Schema** tab. This tab allows you to populate a table that maps the names of source attributes to internal references used by Identity Bridge.
8. Click **Load Template**. The table is populated with schema mappings. (If the table remains empty, check that the contents of the Search Filter box has been entered correctly.)




| Name | Internal Reference | Internal Type | Key Field | Ignore Reading | Ignore Delta |
|---|---|---|---|---|---|
| OHSStaff | OHSStaff | STRING | | | |
| billingCode | billingCode | STRING | | | |
| cACertificate;binary | cACertificate;binary | STRING | | | |
| cn | cn | STRING | | | |
| comment | comment | STRING | | | |
| document | document | STRING | | | |
| employerCode | employerCode | STRING | | | |
| facsimileNumber | facsimileNumber | STRING | | | |
| firstAidOfficer | firstAidOfficer | STRING | | | |
| floorWarden | floorWarden | STRING | | | |
| givenName | givenName | STRING | | | |
| image | image | STRING | | | |
| location | location | STRING | | | |
| mail | mail | STRING | | | |
| mailingAddress | mailingAddress | STRING | | | |
| manager | manager | STRING | | | |
| mhs-or-addresses | mhs-or-addresses | STRING | | | |
| objectClass | objectClass | STRING | | | |
| printCode | printCode | STRING | | | |
| skillSet | skillSet | STRING | | | |
| sn | sn | STRING | | | |
| telecomTelexNumber | telecomTelexNum... | STRING | | | |
| telephoneNumber | telephoneNumber | STRING | | | |
| title | title | STRING | | | |
| userCertificate;binary | userCertificate;bin... | STRING | | | |
| webTemplate | webTemplate | STRING | | | |

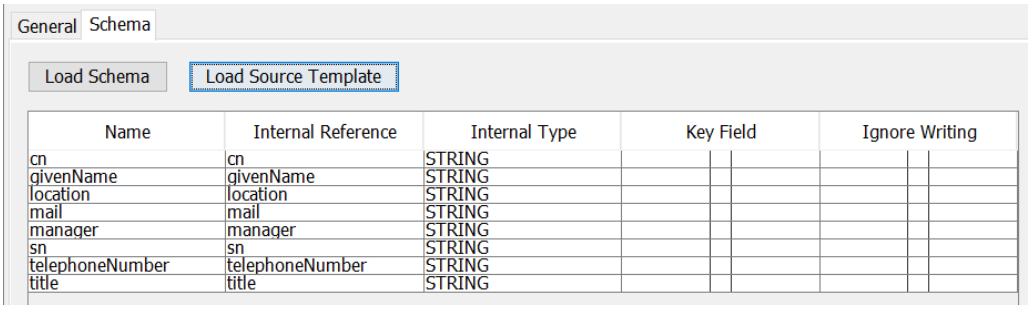
   Note that by default, Identity Bridge sets every Internal Type to `STRING`. This can be modified as required.
9. Holding down the **Ctrl** key, select all entries in the table except the following eight:
   `cn`, `givenName`, `location`, `mail`, `manager`, `sn`, `telephoneNumber`, `title`

10. Click **Delete**. The selected entries are removed from the table.




To define the task's target:

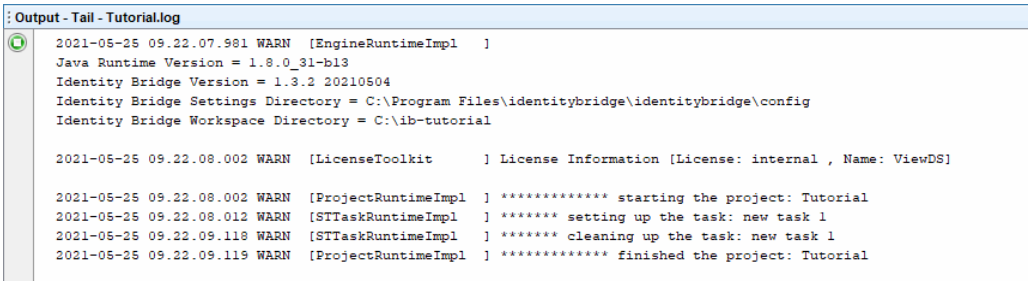
1. At the top of the **Task** tab, click **Target**.
2. Select **Write Header Columns**.
3. At the top of the **Target** area, click the **Schema** tab.
4. Click **Load Source Template**. The table is populated with source attributes that will be written to the target resource.




# Run the project

To run the project:

- From the toolbar, click **Save All** followed by **Run Project**. The Output pane is displayed, which shows the project's output log.

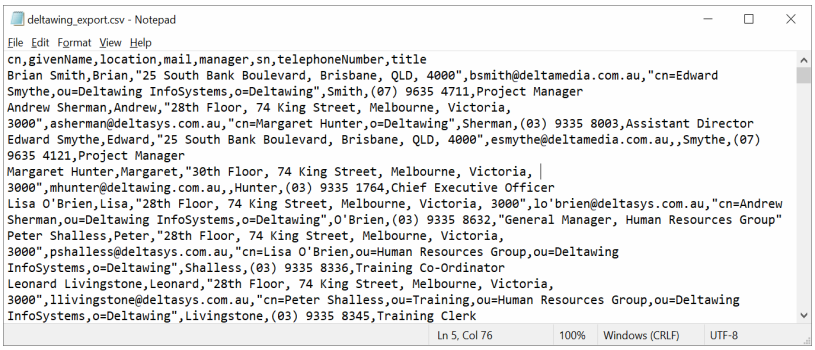



The project's output log is a text file with the same name as the project. Its default location is the `[workspace]/ib_data/logs` directory.

## View the result

As a result of running the project, Identity Bridge also creates the target-resource file, `deltawing_export.csv`, in your workspace folder.

The file includes the header columns along with the specified attributes for each `organizationalPerson` entry extracted from the source.




# Apply functions

This stage of the tutorial demonstrates how to apply JavaScript functions in a task.

There are several contexts where functions can be applied:

- **Task initialization function**
  Applied before reading any source data. For example, a function might be used to initialize temporary files, check resources are available, or apply a condition to running the task.
- **Task termination function**
  Applied after all data has been written to the target. For example, a function could clean up temporary files, report results, or send notifications.
- **Audit function**
  Applied after each record has been written to the target.
- **Per entry/record function**
  Applied to each database record before it is written to the target, which allows data manipulation such as adding, removing or merging attributes within a record. The function receives and returns a single record.
- **Per attribute/field function**
  Applied to a specific attribute before its record is written to the target. The function receives and returns a single attribute.

Functions are written and maintained using Identity Bridge's *JavaScript Function Editor*. They can use variables whose scope is an entire project, and can call built-in functions provided by Identity Bridge. Additionally, functions can call Java libraries.

This tutorial includes two simple examples that demonstrate how to apply a per-attribute and per-record function.

## Apply a per-attribute function

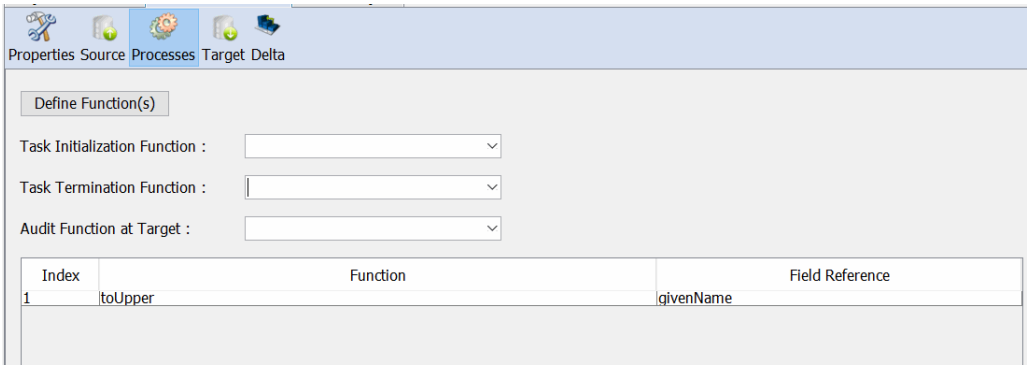
To apply a very simple function that converts a string attribute to upper-case:

1. At the top of the **Task** tab, click **Processes**. The different contexts for functions are displayed, along with the function table for per-record and per-attribute functions.
2. Click **Define Functions**. The JavaScript Function Editor is displayed in a new tab. It is labelled with the name of your project followed by the `.js` extension.
3. Enter the following:

```
function toUpper(attr){
  attr[0] = attr[0].toUpperCase();
  return attr;
}
```

   Note that the input parameter for a per-attribute function is always an array.

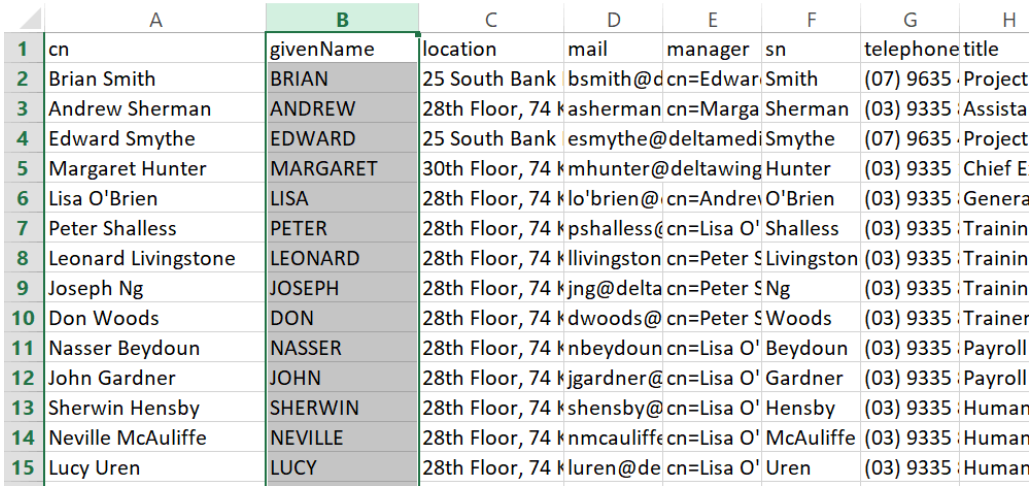
   The argument in the above example is an array with a single element because its corresponding database attribute holds a single value.
4. Press **Ctrl**+**S** or from the toolbar, click **Save All**.
5. Click the **Task** tab.
6. Click **Add**. A new row is added to the function table.
7. In the new row, double-click the **Function** cell. A drop-down arrow is displayed in the cell.
8. Click the drop-down arrow followed by **toUpper**. The function is added to the table.
9. Double-click the **Field Reference** cell. A drop-down arrow is displayed.
10. Click the drop-down arrow followed by **givenName**. This is the attribute that will be passed to, and returned by, the function.
11. Press the **Tab** key.



### Test the function

To run the function:

1. Ensure that you completed the last step in the above task.
2. From the toolbar, click **Save As** followed by **Run Project**. The Output pane is displayed, which shows the project's output log.
3. In your workspace folder, open your target-resource file (`deltawing_export.csv`) and check that the values in the `givenName` column are now all in upper-case.



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | cn | givenName | location | mail | manager | sn | telephone | title |
| 2 | Brian Smith | BRIAN | 25 South Bank | bsmith@d | cn=Edwar | Smith | (07) 9635 | Project |
| 3 | Andrew Sherman | ANDREW | 28th Floor, 74 k | asherman | cn=Marga | Sherman | (03) 9335 | Assista |
| 4 | Edward Smythe | EDWARD | 25 South Bank | esmythe@deltamedi | | Smythe | (07) 9635 | Project |
| 5 | Margaret Hunter | MARGARET | 30th Floor, 74 k | mhunter@deltawing | | Hunter | (03) 9335 | Chief E |
| 6 | Lisa O'Brien | LISA | 28th Floor, 74 k | lo'brien@ | cn=Andre | O'Brien | (03) 9335 | Genera |
| 7 | Peter Shalless | PETER | 28th Floor, 74 k | pshalless@ | cn=Lisa O' | Shalless | (03) 9335 | Trainin |
| 8 | Leonard Livingstone | LEONARD | 28th Floor, 74 k | llivingston | cn=Peter S | Livingston | (03) 9335 | Trainin |
| 9 | Joseph Ng | JOSEPH | 28th Floor, 74 k | jng@delta | cn=Peter S | Ng | (03) 9335 | Trainin |
| 10 | Don Woods | DON | 28th Floor, 74 k | dwoods@ | cn=Peter S | Woods | (03) 9335 | Trainer |
| 11 | Nasser Beydoun | NASSER | 28th Floor, 74 k | nbeydoun | cn=Lisa O' | Beydoun | (03) 9335 | Payroll |
| 12 | John Gardner | JOHN | 28th Floor, 74 k | jgardner@ | cn=Lisa O' | Gardner | (03) 9335 | Payroll |
| 13 | Sherwin Hensby | SHERWIN | 28th Floor, 74 k | shensby@ | cn=Lisa O' | Hensby | (03) 9335 | Human |
| 14 | Neville McAuliffe | NEVILLE | 28th Floor, 74 k | nmcauliffe | cn=Lisa O' | McAuliffe | (03) 9335 | Human |
| 15 | Lucy Uren | LUCY | 28th Floor, 74 k | luren@de | cn=Lisa O' | Uren | (03) 9335 | Human |

## Apply a per-record function

To apply a very simple function that processes the directory record for each Project Manager located in Brisbane:

1. In the **Processes** area of the **Task** tab, click **Define Functions**. The JavaScript Function Editor is displayed.
2. Enter the following (below the existing `toUpper` function):

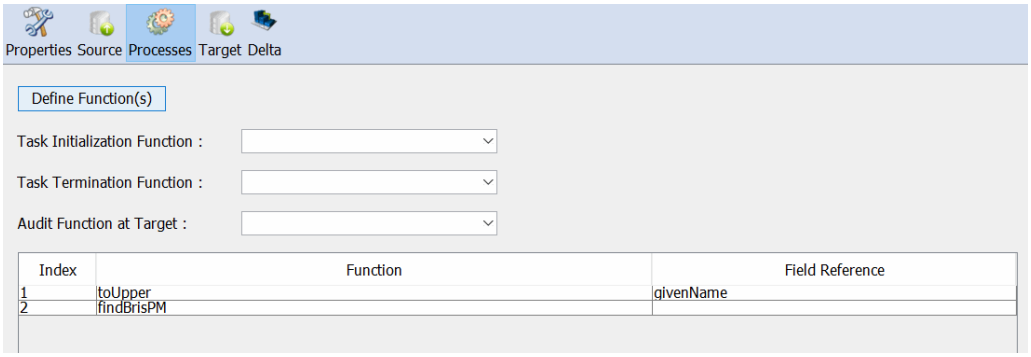
```
function findBrisPM(record){
  if((record['title'][0] == 'Project Manager') && (record
['location'][0].search('Brisbane'))){
    record['givenName'][0] = record['givenName'][0].toLowerCase();

    record['cn'][0] = record['cn'][0].toUpperCase();
  }
  return record;
}
```

Note that the input parameter for a per-record function is always a two-dimensional array.

The first dimension holds the record's attribute names, and the second holds the attribute values.

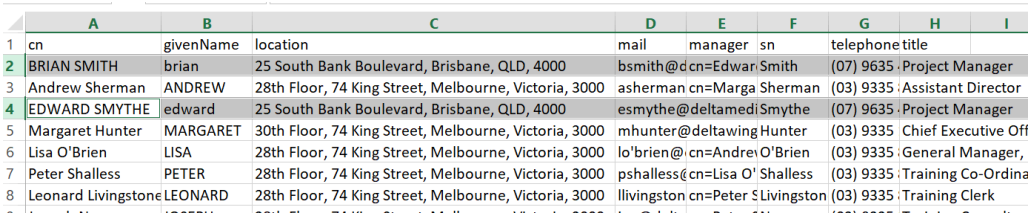
3. As a quick aside, complete the following steps to access the application's built-in functions:
   a. Enter `$context.` (including the dot at the end).
   b. Press **Ctrl**+**Space**. The list of built-in functions is displayed. They are described in the application's help.
   c. Now that you know how to access the built-in functions, delete `$context.`
4. Press **Ctrl**+**S** or from the toolbar, click **Save All**.
5. Click the **Task** tab.
6. Click **Add**. A new row is added to the function table.
7. In the new row, double-click the **Function** cell. A drop-down arrow is displayed in the cell.
8. Click the drop-down arrow followed by **findBrisPM**. The function is added to the table.
9. Press the **Tab** key.
   Note that the empty Field Reference cell means that the function will be applied to an entire record. Also note that Index column defines the order in which functions will be processed. You can, however, click and drag a function to a new position in the table.




## Test the function

To run the function:

1. Ensure that you completed the last step in the above task.
2. From the toolbar, click **Save As** followed by **Run Project**. The Output pane is displayed, which shows the project's output log.
3. In your workspace folder, open your target-resource file (`deltawing_export.csv`).
4. For the Project Managers in Brisbane, check that their `givenName` is all in upper-case and their `cn` is all lower-case.



| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | cn | givenName | location | mail | manager | sn | telephone | title | |
| 2 | BRIAN SMITH | brian | 25 South Bank Boulevard, Brisbane, QLD, 4000 | bsmith@d | cn=Edwar | Smith | (07) 9635 | Project Manager | |
| 3 | Andrew Sherman | ANDREW | 28th Floor, 74 King Street, Melbourne, Victoria, 3000 | asherman | cn=Marga | Sherman | (03) 9335 | Assistant Director | |
| 4 | EDWARD SMYTHE | edward | 25 South Bank Boulevard, Brisbane, QLD, 4000 | esmythe@deltamedi | | Smythe | (07) 9635 | Project Manager | |
| 5 | Margaret Hunter | MARGARET | 30th Floor, 74 King Street, Melbourne, Victoria, 3000 | mhunter@deltawing | | Hunter | (03) 9335 | Chief Executive Off | |
| 6 | Lisa O'Brien | LISA | 28th Floor, 74 King Street, Melbourne, Victoria, 3000 | lo'brien@ | cn=Andre | O'Brien | (03) 9335 | General Manager, | |
| 7 | Peter Shalless | PETER | 28th Floor, 74 King Street, Melbourne, Victoria, 3000 | pshalless@ | cn=Lisa O' | Shalless | (03) 9335 | Training Co-Ordina | |
| 8 | Leonard Livingstone | LEONARD | 28th Floor, 74 King Street, Melbourne, Victoria, 3000 | llivingston | cn=Peter S | Livingston | (03) 9335 | Training Clerk | |

### Accessing Java

Also worth highlighting is that functions can access Java libraries directly (third-party libraries can be accessed from the `[workspace]/lib` directory). Here is a simple example to illustrate:

```
// Use JavaScript Date object to get current date and time
var now = new Date();
// Format dates suitable for ViewDS timestamp comparison
var viewdsDateFormat = new java.text.SimpleDateFormat
('yyyyMMddHHmmss.SSSZ');
var thisRunDate = viewdsDateFormat.format(now);
// Create new Active Directory Password
var password = new java.lang.String("\"NewP@55w0rd\"");
var encPwd = password.getBytes("UTF-16LE");
rec['unicodePwd'] = [ encPwd ];
```

## Distribute the project

The final stage is to distribute your project to a production environment.

A production environment should have Identity Bridge installed. The Identity Bridge Engine can then be run independently to execute projects as batch processes scheduled by the likes of Windows Scheduler or Linux *cron*, for example.

To distribute the tutorial project:

- Compress the tutorial's workspace directory (for example, `c:\ib-tutorial`) and copy it to the same location in the production environment's directory structure.

Later, if you need to distribute an updated version of the project:

- Copy the project files, `c:\ib-tutorial\ib-data\projects\Tutorial.js` and `Tutorial.xml`, to the same location in the production environment's directory structure.

To run a project from the command line:

1. Navigate to the `<install dir>\identitybridge\modules` directory (the default `<install dir>` on Windows is `c:\Program Files\`).
2. Enter the following command:
   `java -jar com-viewds-identitybridge-engine-runner.jar -workspace` **`c:\ib-tutorial`** `-project` **`Tutorial`**
   The command structure is as follows:
   `java -jar com-viewds-identitybridge-engine-runner.jar -workspace <workspace> -project <project> [-task <task>]`
   Where the `<workspace>` includes its full path, `<project>` is the name of the project, and `<task>` is optional in case you want to run just one task within the project.

# Advanced options

Along with the functionality introduced in the Tutorial, the following can be included in projects:

- Project variables
- Variable sets and function libraries
- Delta updates
- Change logs
- JavaScript connector

## Project variables

You can create variables for a project. They are initialized when the project starts and persist from one task to another. A project variable could be used, for example, to maintain an error count within a project.

Project variables can be used in the project's functions. They can also be entered into text boxes to declare, for example, project and task parameters.

A project variable is declared in a text box as follows:

```
${[variable name]}
```

To create a project variable:

1. From the Workspace Explorer, double-click a project. The Project tab is displayed.
2. At the top of In the Project tab, click **Properties**.
3. In the **Project Variables** section, click **Add**. A new row is added to the variable table.
4. Double-click the **Variable Name** cell in the new row and enter a variable name.
5. Double-click the adjacent **Value** cell and enter an initial value for the variable. The following types are supported:
   - String
     Example value: `"abc"`
   - Simple array
     Example value: `["top", "person", "organizationalPerson"]`
     Where: `variableName[0]` returns `top`
   - Associative array
     Example value: `{"id":100, "is_vip":true, "name":"Tom"}`
     Where: `variableName["id"]` returns `100`

- Multidimensional array
  Example value: `[ {"id":100, "is_vip":true, "name":"Tom"} {"id":200, "is_vip":false, "name":"Jerry"} ]`
  Where: `variableName[0]["id"]` returns `100`

6. From the main toolbar, click **Save All**.

# Variable sets and function libraries

The Store in the Workspace Explorer allows you to define variable sets and function libraries. Each is discussed below with simple examples to illustrate.

## Variable sets

A variable set is a collection of variables that can be reused in your projects. The variables can be used in functions; and also in text boxes to declare, for example, resource and task parameters.

An example of how they might be used is to have different variable sets for different environments, such as *Development* and *Production*. Each set would declare the connection details for a project's source and target resources, which would allow you to switch between environments quickly and easily.


This example is illustrated in the following simple walk-through.

### Walk-through

The walk-through has four stages and builds on aspects of this guide's Tutorial:

- Define variable sets
- Select a variable set
- Use a set's variables
- Switch between sets

#### *Define variable sets*

To define a 'Development' variable set:

1. From the Workspace Explorer, expand the **Store**. It contains Variable Sets and Function Libraries.
2. Right-click **Variable Sets** and select **New**.

3. Enter `Development` followed by **OK**. The new variable set is added to the Workspace Explorer.
4. Double-click **Development**. The Variable Set tab is displayed.
5. To the right of the screen, click **Add**. A new row is added to the table of variables.
6. Double-click the **Variable Name** cell in the new row, and enter `sourceLDAP`.
7. Double-click the **Value** cell in the new row and enter the URL for the LDAP source used in the tutorial: `ldap://corporate.demo.viewds.com:3006`.
8. Repeat this task from step 5 above to declare the variable `targetCSV` with the value `c:\ib-tutorial\deltawing_export.csv` (this is the target in the tutorial).
9. Click **Save All**.
10. Repeat this task to create a variable set called `Production`. It should contain variables with the same names as those declared in the Development set (`sourceLDAP` and `targetCSV`), but with different values.

### *Select a variables set*

To select a variable set so that it can be applied to the tutorial project:

1. From the Workspace Explorer, double-click the **Tutorial** project. The Project tab is displayed.
2. At the top of the Project tab, click **Properties**.
3. In **Variable Set Configuration** select **Development**.

### *Use a set's variables*

To use the variables in the selected set:

1. At the top of the Project tab, click **Design View**.
2. Double-click the resource **Deltawing**. Its Resource Configuration is displayed.
3. Replace the value in **Directory LDAP URL** with `${sourceLDAP}` followed by **OK**. (This syntax can be used in any text box.)
4. Double-click the resource **Deltawing Export**. Its Resource Configuration is displayed.
5. Replace the value in the **File** box with `${targetCSV}` followed by **OK**.

### *Switch between sets*

To apply a different variable set:

1. At the top of the Project tab, click **Properties**.
2. From **Variable Set Configuration** select **Production**.

# Function library

A function library can be included in multiple projects.

Implementing a library function is illustrated through the following simple walk-through.

## Walk-through

The walk-through has three stages and builds on aspects of this guide's Tutorial:

- Declare a function library
- Include the library function
- Call the wrapper function

### *Declare a function library*

To declare a library function:

1. From the Workspace Explorer, expand the **Store**. It contains Variable Sets and Function Libraries.
2. Right-click **Function Libraries** and select **New**.
3. Enter `test` followed by **OK**. The new function library is added to the Workspace Explorer.
4. Double-click **test**. The JavaScript Function Editor is displayed.
5. Enter the following. Note that the last line makes the function library available to be included in your projects.

```
function toLower(attr){
  attr[0] = attr[0].toLowerCase();
  return attr;
}
Module.exports = { toLower };
```

6. Click **Save All**.

### *Include the function library*

To include the above library in the Tutorial project:

1. From the Workspace Explorer, expand the **Tutorial** project.
2. Below the Tutorial project, double-click **Functions**. The `Tutorial.js` tab is displayed. It contains the functions `toUpper()` and `findBrisPM()` that were declared during the tutorial.
3. Enter the following line to include the `test` function library:

```
var testModule = require(test);
```

4. Enter the following function, which wraps the library function `toLower()`:

```
function buildNewEmail(record){
  var firstName = record["givenName"][0];
  var lastName = record["sn"][0];
  var newEmail = firstName + testModule.toLower(lastName) +
"@viewds.com";
  record["email"] = [ newEmail ];
  return record;
}
```

5. Click **Save All**.

### *Call the wrapper function*

To call the function `buildNewEmail`:

1. Double-click the **Tutorial** project. The Project tab is displayed.
2. At the top of the tab, click **Design View** if not already selected.
3. Double-click the task arrow **Export People**. The Task tab is displayed.
4. At the top of the tab, click **Processes**.
5. Click **Add**. A new entry is added to the function table.
6. Double-click the new entry's Function cell, select **buildNewEmail**, then press the tab key. The empty cell results in the function being applied to entire records.
7. Click **Save All**.

# Delta updates

A delta update performs a partial synchronization of any changes since the last synchronization. It is therefore very quick and reduces the load on the target resource.

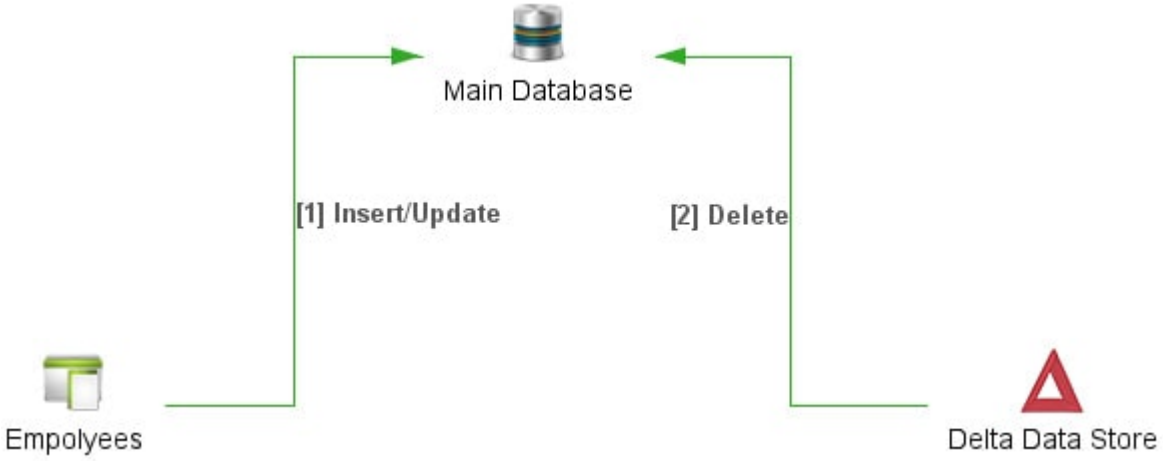
A delta update can:

- Identify new and modified source records, and then write them to the target.
- Identify records that have been deleted from a source, and then delete (or otherwise process) the corresponding records stored by the target.

**NOTE**: The delta-update feature is also required by the change-log connectors.

The rest of this section walks through how to implement a project that includes delta updates.

The project has three resources: *Employees* (CSV file), *Main Database* (SQL database), and *Delta Data Store* (SQL database). It also includes two tasks that apply delta updates: *Insert/Update* and *Delete*.




When an entry in the Employees source is either created or updated, the Create/Update task synchronizes these changes in the Main Database. When an entry is deleted from Employees, the Delete task sychronizes the change in the Main Database.

The walk-through to implement the project has three main stages:

- Create a project and resources
- Define the insert/update task
- Define the delete task

## Create a project and resources

The tasks in this section are:

- Installing a delta database
- Creating a source resource
- Creating a new project

- Defining the Delta Data Store
- Adding the target
- Adding the source

### Installing a delta database

A delta update requires a delta database. This allows Identity Bridge to identify when there have been changes to a source resource.

Identity Bridge can use most SQL databases as a delta database, including MSSQL, MYSQL and the H2 Database Engine. This walk-through uses the H2 Database Engine.

To install the H2 Database Engine:

1. Download the H2 installer and install the H2 Database Engine in a location of your choice. (Make a note of this location as you will need it later during this walk-through.)
2. Copy the `.jar` file from `../H2/bin/` to the `<workspace>/ib_data/lib` directory (see Installing Identity Bridge).

The H2 Database Engine fulfils two roles in this walk-through. It provides the Delta Data Source and target Main Database.

### Creating a source resource

The source resource for this walk-through is a CSV file of employees' details.

Create a text file named `employees.csv` that contains the following data, and store it in the project's workspace.

```
employeeNumber,givenName,sn,department,title,telephoneNumber,mail,location,manager
10175846,Andrew,Sherman,Deltawing Infosystems,Managing Director,(03) 9335
8003,asherman@deltasys.com.au,"28th Floor, 74 King Street, Melbourne, Victoria,
3000",10937986
10268498,Brian,Smith,Deltawing Infosystems,Project Manager,(07) 9635
4711,bsmith@deltamedia.com.au,"25 South Bank Boulevard, Brisbane, QLD,
4000",10852068
10018023,Ian,Campbell,Deltawing,Executive Assistant,(03) 9335
1778,icampbell@deltawing.com.au,"30th Floor, 500 Hill View Road, Melbourne,
Victoria, 3000",
10852068,Edward,Smythe,Deltawing Infosystems,Project Manager,(07) 9635
4121,esmythe@deltamedia.com.au,"25 South Bank Boulevard, Brisbane, QLD,
4000",10175846
10267758,Kerri,Smith,Human Interfaces,Product Researcher : Human Interfaces,(07)
9635 3263,ksmith@deltamedia.com.au,"9th Floor, 25 South Bank Boulevard, Brisbane,
QLD, 4000",10870756
10293166,Lisa,O'Brien,Human Resources Group,"General Manager, Human Resources
Group",(03) 9335 8632,lo'brien@deltasys.com.au,"28th Floor, 74 King Street,
Melbourne, Victoria, 3000",10175846
10464844,Peter,Shalless,Training,Training Co-Ordinator,(03) 9335
8336,pshalless@deltasys.com.au,"28th Floor, 74 King Street, Melbourne, Victoria,
3000",10293166
```

```
10655283,Leonard,Livingstone,Training,Training Clerk,(03) 9335
8345,llivingstone@deltasys.com.au,"28th Floor, 74 King Street, Melbourne,
Victoria, 3000",10464844
10251265,Joseph,Ng,Training,Training Consultant,(03) 9335
8390,jng@deltasys.com.au,"28th Floor, 74 King Street, Melbourne, Victoria,
3000",10464844
10382028,Don,Woods,Training,Trainer,(03) 9335 8365,dwoods@deltasys.com.au,"28th
Floor, 74 King Street, Melbourne, Victoria, 3000",10464844
```

### Creating a new project

To create a project for this walk-through:

1. In the **Workspace Explorer** below **Workspace**, right-click **Projects** and click **New Project**. The New Project window is displayed.
2. Enter a **Name** such as `Delta Example`, and click **OK**. The project is added to the Workspace Explorer.
3. Double-click the new project. An empty Project tab is displayed.

### Defining the Delta Data Store

To define a Delta Data Store:

1. In the **Project** tab, right-click anywhere in the tab's canvas and then click **New Resource in Project**. The New Resource Wizard is displayed.
2. Enter a **Resource Name** such as `Delta Data Store`.
3. From the list of **Connector Plugins** click **Delta Data Store**.
4. Click **Next**. The Parameters are displayed.
5. In **Database JDBC URL**, enter `jdbc:h2:/<H2 path>/test`.
   Where `<H2 path>` is the path to where you installed H2; and `test` is the name of the delta database that will be created for this walk-through.
6. In **JDBC Driver Class**, enter `org.h2.Driver`.
7. In **User Name**, enter a username of your choice (for example, `su`).
8. In **User Password**, enter a password of your choice (for example, `admin`). These credentials will apply to the `test` database.
9. Click **Test Connection**.
10. Click **Finish**. The new resource is added to the Project tab.

### Adding the target

To add the target resource (H2 database) to the project:

1. In the **Project** tab, right-click anywhere in the tab's canvas and then click **New Resource in Project**. The New Resource Wizard is displayed.
2. Enter a **Resource Name** such as `Main Database`.
3. From the list of **Connector Plugins** click **SQL**.

4. Click **Next**. The Parameters are displayed.
5. In **Database JDBC URL**, enter `jdbc:h2:/<H2 path>/test`.
6. In **JDBC Driver Class**, enter `org.h2.Driver`.
7. In **User Name**, enter the username that you declared in the previous task (Defining the Delta Data Store).
8. In **Password**, enter the username that you declared in the previous task.
9. Click **Finish**. The new resource is added to the Project tab.

### Adding the source

To add the source resource (CSV flat file) to the project:

1. In the **Project** tab, right-click anywhere in the tab's canvas and then click **New Resource in Project**. The New Resource Wizard is displayed.
2. Enter a **Resource Name** such as `Employees`.
3. From the list of **Connector Plugins** click **Flat File**.
4. Click **Next**. The Parameters are displayed.
5. Click **Browse** and select the `employees.csv` file you created earlier.
6. For the **File Format**, click **csv**.
7. Select **Read Header Columns**.
8. Click **Finish**. The new resource is added to the Project tab.

## Define the insert/update task

The tasks in this section are:

- Creating the task
- Defining the source
- Defining the target
- Defining the delta
- Running the project
- Inspecting the H2 database
- Testing the task

### Creating the task

To create the task:

1. Reposition the three resources in the **Project** pane, and then click **Connect Mode**.
2. Click and drag from **Employees** to **Main Database**. An arrow is displayed between the resources.
3. Double-click the arrow. The Task tab is displayed. It has five areas: Properties, Source, Processes, Target, and Delta.
4. In **Task Name**, enter `Insert/Update`.

### Defining the source

To define the task's source:

1. At the top of the tab, click **Source**.
2. Click the **Schema** tab.
3. Click **Load Template**. The structure of the CSV file is displayed in the table.

### Defining the target

To define the target:

1. At the top of the tab, click **Target**.
2. In **Table Name**, enter `EMPLOYEES`.
3. In **Run Mode**, click **INSERT_OR_UPDATE**.
4. In **DDL Script Execution**, click **CREATE_TABLE_IF_NOT_EXISTS**.
5. Click the **Schema** tab.
6. Click **Load Source Template**. The structure of the source CSV file is displayed in the table.
7. In the **EMPLOYEENUMBER** row, select the **Key Field** checkbox.
8. Click the **General** tab.
9. Click the **Load DDL Template** button. The SQL command to create an 'Employees' table in the target is displayed in the box.

### Defining the delta

To define the task's delta update:

1. At the top of the tab, click **Delta**.
2. In **Delta Store Resource**, click **Delta Data Store**.
3. In **Delta Table Identifier**, enter `deltaEmployees`.

   This table will be created in H2 when the task runs for the first time. It will store a hashed 'fingerprint' for each record in the source. These fingerprints allow the task to identify new and modified records in the source. (They also allow the 'delete' task to determine when a record has been deleted from the source.)
4. In **Source Key Reference**, enter `employeeNumber`. This is the field in the source that uniquely identifies each entry.
5. In **Target Key Reference**, enter `employeeNumber`.
6. Select **Enable Delta Evaluation** and **Enable Delta Confirmation**.

   A delta update involves evaluation and confirmation. These two stages are described below.

   The delta database contains a hashed 'fingerprint' for every record in the source resource. When the insert/update task is executed, the delta process will *evaluate* each source record against its fingerprint. This allows it to identify whether a record has been modified. When a modified record is detected, the delta process *confirms* the change by writing a new fingerprint for the record to the delta database.

The confirmation stage can be deferred when more complex processing is required in multiple tasks.

7. Click **Save All**.

### Running the project

To run the project:

- From the **Project** tab, click the **Run Project** button in the toolbar. The project log is displayed.

You can now inspect the H2 database to confirm that the task created an 'employees' table and that it contains the same data as the source.

### Inspecting the H2 database

To inspect the H2 database:

1. Enter the following at the command line:

   ```
   <workspace>\ib_data\lib>java [filename].jar
   ```

   A new browser session opens containing the H2 login dialog. If your browser displays a blank page, cut and paste the URL to another type of browser (such as Google Chrome).

   The dialog is populated by the values you entered when Defining the Delta Data Source.
2. In **Password**, enter the password you declared when Defining the Delta Data Source.
3. Click **Connect**. The H2 Console is displayed. The navigation pane on the left includes the database **test**, which includes the **EMPLOYEES** table.
4. In the navigation pane, click **EMPLOYEES**. A statement is displayed in the SQL statement box.
5. Click **Run**. The contents of the table is displayed.
6. In the toolbar, click the disconnect button. The **Login** dialog is displayed.

> **NOTE**: There can only be one connection to the H2 Database Engine when it is running in embedded mode.

### Testing the task

To test the task:

1. Open the `Employees` CSV file in a text editor.
2. Modify an existing entry and add a new entry, then save and close the file.
3. From the **Project** tab in Identity Bridge, click the **Run Project** button in the toolbar. The project log is displayed.
4. Inspect the H2 database to confirm that the your changes to the source have been applied to the target resource.

## Define the delete task

The delete task will maintain tables in the H2 Database Engine. The tables allow it to detect when a record has been removed from the source resource.

The tasks in this section are:

- Creating the task
- Defining the source
- Defining the target
- Defining the delta
- Testing the task

### Creating the task

To create the task:

1. From the **Project** pane, click **Connect Mode**.
2. Click and drag from **Delta Data Source** to **Main Database**. An arrow is displayed between the resources.
3. Double-click the arrow. The Properties area of the Task tab is displayed.
4. In **Task Name**, enter `Delete`.

### Defining the source

To define the task's source:

1. At the top of the tab, click **Source**.
2. In **Delta Table Identifier**, enter `deltaEmployees`.
3. Select **Fetch untouched delta entries from last run**.
4. Click the **Schema** tab followed by **Load Template**. The table is populated with two rows: **source_key** and **target_key**. Identity Bridge uses these tables to identify changes at the source resource that need to be synchronized to the target resource.

### Defining the target

To define the task's target:

1. At the top of the tab, click **Target**.
2. In **Table Name**, enter `EMPLOYEES`.
3. In **Run Mode**, click **DELETE_EXIST**.
4. Click the **Schema** tab.
5. Click **Load Schema** to copy the source's schema definition.
6. Click **Add**. An empty row is added to the table.
7. Double-click the row's **Name** cell. A drop-down list is displayed in the cell.
8. From the drop-down list, click **EMPLOYEENUMBER** and then press the return key.
9. Double-click the row's **Internal Reference** cell.

10. Replace the contents of the cell with `target_key`.
11. Select the row's **Key Field** checkbox.
12. Click **Save All**.

### Defining the delta

To define the task's delta update:

1. At the top of the tab, click **Delta**.
2. In **Delta Store Resource**, click **Delta Data Store**.
3. In **Delta Table Identifier**, enter `deltaEmployees`.
4. In **Source Key Reference**, enter `source_key`.
5. In **Target Key Reference**, enter `target_key`. Identity Bridge uses the key references to identify when the target needs to be synchronized with the source.
6. Select **Enable Delta Confirmation** and **Delete Delta entry on Confirmation**.
7. Click **Save All**.

### Testing the task

To test the task:

1. Open the `HR Data` CSV file in a text editor.
2. Delete an entry and then save and close the file.
3. From the **Project** tab in Identity Bridge, click the **Run Project** button in the toolbar. The project log is displayed.
4. Inspect the H2 database to confirm that the your changes to the source have been applied to the target resource.

# Change logs

A change log records all modifications to a source and can therefore be used to provide 'on change' updates. This is particularly useful when frequent updates are made, or when the dataset is very large, as it ensures that Identity Bridge only processes modified data.

Identity Bridge includes connectors for *LDAP Changelog* and *Active Directory Change Detection*. Both require a total-update task to synchronize the source and target initially, and to run if they become mismatched later, plus a delta database to map source and target entries.

Overviews of the steps required to set up both connectors are below.

## LDAP Changelog

1. Create a total-update task using the **LDAP** connector for an LDAP source (see Create resource and Create task). Ensure that a delta-database resource is defined and delta updates are configured for the total-update task (see Delta updates).
2. Create a change-log update task using the **LDAP Changelog** connector for the LDAP source. Ensure that the appropriate run mode for your chosen target is selected (for example, **CHANGE_RECORD** for an LDAP target). Also, ensure that delta updates are configured for the change-log update task.

Note that you do not need to create a separate 'deletes task' for the LDAP Changelog connector. These are handled automatically.

## Active Directory change detection

1. Create a total-update task using the **Active Directory** connector for an Active Directory source. Ensure that a delta-database resource is defined and delta updates are configured for the total-update task.
2. Create a change-log add and update task using the **Active Directory Change Detection**connector. Ensure that source **Scope** for the task is **ADDED_OR _UPDATED**; that the appropriate run mode for your target is selected (for example, **CHANGE_RECORD** for an LDAP target); that delta updates are configured for the add and update task.
3. Create a change-log delete task using the **Active Directory Change Detection** connector. Ensure that source **Scope** for the task is **DELETED**; that the appropriate run mode for your chosen target is selected (for example, **CHANGE_RECORD** for an LDAP target); that delta updates are configured and **Delete Delta Entries on Confirmation** is selected for the **Delta Confirmation** task.

# JavaScript connector

Identity Bridge connects to a resource through a *connector plugin*. Each connector plugin corresponds to a particular resource, such as an LDAP directory, SQL database, or flat file. The exception is the JavaScript connector plugin.

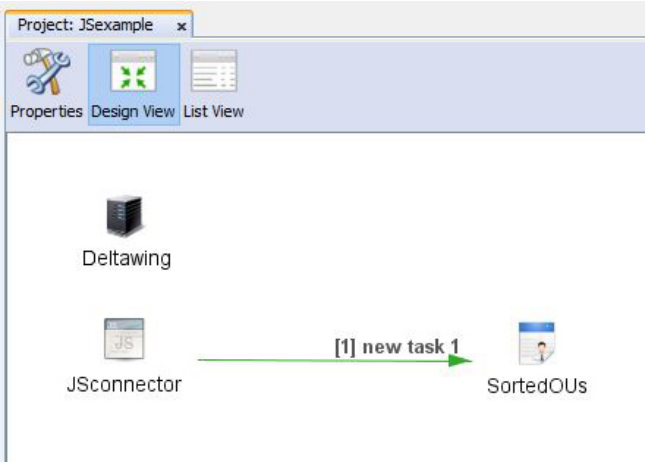
The JavaScript connector plugin allows you to connect to any type of resource and to include custom functionality in the connection. It is defined through a series of JavaScript functions, which you can develop through the JavaScript Functions Editor.

This subsection describes an example project that includes a JavaScript connector:

- Example project
- Example functions
- Creating the example project

## Example project

The Design View for the example project is shown below.




The project's source is the JavaScript connector plugin, JSconnector. It connects to a demonstration directory called Deltawing (also used in the Tutorial), extracts the organizational units from the directory, and sorts them according to character length.

The project's target is an LDIF file.

## Example functions

A JavaScript connector is defined by a series of functions.

There are two optional functions:

- Connector Initialization function
  This function runs before connecting to the resource and before any records are processed by the task.
- Connector Termination function
  This function runs when disconnecting from the resource after all records have been processed by the task.

And two mandatory functions, when the plugin connects to the project's source:

- `hasNextRecord` function – identifies whether there is another record to be processed in the source resource. The function must return a Boolean value. If it returns True, then the getNextRecord function will be called; if it returns False, then the task will terminate.
- `getNextRecord` function – obtains the next record in the source resource.

The example project's Connector Initialization function is shown below.

```
function Init_SyncOuStructure () {
  /* Fetch OU structure using '$context.searchByLDAP' and
  sort on the length of distinguishedName (dn) */
  nextOU = null;
  ouList = []; //Initialise ouList as global
  var myLDAP = "Corporate Directory"; // Resource definition to
search as source
  var attrs = [ "ou", "objectClass", "description" ]; // attributes
to be returned
  var searchBase = "o=Deltawing";
  var searchScope = "SUBTREE";
  var searchFilter = "(objectClass=organizationalUnit)";
  var res = $context.searchByLDAP(myLDAP, JSON.stringify(attrs),
searchBase, searchScope, searchFilter);
  // Convert result object (res) to array ready for sorting
  for (var i = 0; i < res.length; i++) {
    ouList[i] = res[i];
  }
  ouList.sort(function(a,b) {
    var retval = 0;
    if(a['dn'][0].length > b['dn'][0].length) retval = 1;
    if(b['dn'][0].length > a['dn'][0].length) retval = -1;
    return retval;
  });
}
```

The `hasNextRecord` function is as follows.

```
function ouHasNext() {
  // Get next entry to from sorted list else return false
  while( nextOU = ouList.shift() ) {
    return true;
  }
  return false;
}
```

Finally, the `getNextRecord` function.

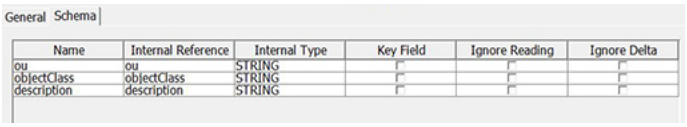
```
function ouGetNext() {
  // return 'nextOU' from Init_SyncOuStructure result
  return nextOU;
}
```

## Create the example project

To create the example project in Identity Bridge Studio:

1. Create a new project called `JSexample`.
2. Create a source resource called `Deltawing`. Aside from Resource Name, use the same settings used in the tutorial.
3. Create the target resource:
   a. Right-click anywhere in the **Project** tab, then click **New Resource in Project**. The New Resource Wizard is displayed.
   b. In the **Resource Name** box, enter `SortedOUs`.
   c. In the **Connector Plugin** box, select **LDIF File**.
   d. Click **Next**. The Parameters page is displayed.
   e. In **LDIF File**, enter `[Workspace]\sortedOUs.ldif` where `[Workspace]` is the location displayed at the top of the Workspace Explorer.
   f. Click **Finish**. The resource is added to the project.
4. Add the example JavaScript functions to the following text file: `[Workspace]\ib_data\projects\JSexample.js`
5. Create the JavaScript connector:
   a. Right-click anywhere in the **Project** tab, then click **New Resource in Project**. The New Resource Wizard is displayed.
   b. In the **Resource Name** box, enter `JSconnector`.
   c. In the **Connector Plugin** box, select **JavaScript**.
   d. Click **Next** followed by **Finish**. The resource is added to the project.
6. Create the task:
   a. In the top-right corner of the **Project** tab, click **Connect Mode**.
   b. Click and drag from **JSconnector** to the **SortedOUs**. A task arrow links the two resources.
7. Double-click the task arrow. The Task tab is displayed. It contains five areas: Properties, Source, Processes, Target, and Delta.

8. In the **Source** area, select the following:
   - **Connector Initialization Function** select `Init_SyncOuStructure`
   - **hasNextRecord Function** select `ouHasNext`
   - **getNextRecord Function** select `ouGetNext`
9. Click the **Schema** tab, and use the **Add** button to populate the table with three entries:




10. In the **Target** area, select **CONTENT_RECORD** for the **Run Mode** box.
11. Click **Schema** followed by **Load Source Template**. The table is populated with the source's schema.
12. Save and Run the project and then inspect the contents of the target-resource file: `[Workspace]\sortedOUs.ldif`.