# **Integration Guide**

This guide covers the following options for integrating your service, with the primary use case being the integration of a new transfer service.

* Integrate a new transfer service
* Integrate a new data model
* Integrate a new cloud provider

## **Integrate a new transfer service**

A new service can be integrated into the Data Transfer Project by creating a the following extensions:

1) Transfer Extension

2) Auth Extension

Optionally, a new Data Model may be required if your does not exist. Please see Integrate a new Data Model for more information.

### **Transfer Extension**

Transfer Extensions are located in the extensions/data-transfer [module](https://github.com/google/data-transfer-project/tree/master/extensions/data-transfer), where they are organized by service provider and then by data type.

1. Create a new directory for your provider under extensions/data-transfer
   1. mkdir extensions/data-transfer/portability-data-transfer-instagram
2. Create a build.gradle file in your new directory
   1. touch extensions/data-transfer/portability-data-transfer-instagram/build.gradle
3. Add the spi and cloud dependencies to your build file along with any sdk or related dependencies specific to your code
   1. Common dependencies include logging, see example below

/\*  
 \* Copyright 2018 The Data Transfer Project Authors.  
 \*  
 \* Licensed under the Apache License, Version 2.0 (the "License");  
 \* you may not use this file except in compliance with the License.  
 \* You may obtain a copy of the License at  
 \*  
 \* https://www.apache.org/licenses/LICENSE-2.0  
 \*  
 \* Unless required by applicable law or agreed to in writing, software  
 \* distributed under the License is distributed on an "AS IS" BASIS,  
 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 \* See the License for the specific language governing permissions and  
 \* limitations under the License.  
 \*/  
  
dependencies {  
 compile project(':portability-spi-cloud')  
 compile project(':portability-spi-transfer')

// logging  
 compile("org.slf4j:slf4j-api:${slf4jVersion}")  
 compile("org.slf4j:slf4j-log4j12:${slf4jVersion}")

compile(YOUR DEPS HERE)  
}

1. Add an entry for your extension in root settings.gradle
   1. // Instagram transfer extension
   2. include ':extensions:data-transfer:portability-data-transfer-instagram'
   3. // Instagram auth extension
2. Create the directory for your main code
   1. mkdir -p extensions/data-transfer/portability-data-transfer-instagram/src/main/java/org/dataportabilityproject/transfer/instagram/
3. Create InstagramTransferExtension extending TransferExtension
   1. See class javadoc for TransferExtension for full documentation
   2. Return service id
   3. Best practices
      1. Validate the TransferExtension isn’t validated more than once in a running binary
      2. Most providers will interact with their api via a ‘registered’ app providing a client id/key and secret. Those will be stored in a key management system and retrieved via an identifier for the key and one for the secret,
      3. In initialize, retrieve the JobStore from the context
      4. In initialize, retrieve the AppCredentialStore
   4. Return an importer or exporter for a given transfer type
4. Create the folder for the specific data type
   1. mkdir -p extensions/data-transfer/portability-data-transfer-instagram/src/main/java/org/dataportabilityproject/transfer/instagram/photos
5. Create an InstagramPhotoExporter implementing Exporter<TokenAndUrlAuthData, PhotosContainerResource>
6. Create an InstagramPhotoImporter implementing Importer<TokenAndUrlAuthData, PhotosContainerResource>
7. Create the test directory for the importer and exporter
   1. mkdir -p extensions/data-transfer/portability-data-transfer-instagram/src/test/java/org/dataportabilityproject/transfer/instagram/photos
8. Create InstagramPhotoExporterTest with stuff
9. Create InstagramPhotoImporterTest with stuff
10. Create
11. Create
12. Create
13. Create
14. Create
15. Create
16. Create
17. Create
18. Create
19. Create
20. Create

### **Auth Extension**

Auth Extensions are located in the extensions/auth [module](https://github.com/google/data-transfer-project/tree/master/extensions/auth), where they are organized by service provider.

1. Create a new directory for your provider under extensions/auth
   * mkdir extensions/auth/portability-auth-instagram
2. Create a build.gradle file in your new directory
   * touch extensions/auth/portability-auth-instagram/build.gradle
3. Add the spi and cloud dependencies to your build file along with any sdk or related dependencies specific to your code
   * Common dependencies include logging, see example below

/\*  
 \* Copyright 2018 The Data Transfer Project Authors.  
 \*  
 \* Licensed under the Apache License, Version 2.0 (the "License");  
 \* you may not use this file except in compliance with the License.  
 \* You may obtain a copy of the License at  
 \*  
 \* https://www.apache.org/licenses/LICENSE-2.0  
 \*  
 \* Unless required by applicable law or agreed to in writing, software  
 \* distributed under the License is distributed on an "AS IS" BASIS,  
 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 \* See the License for the specific language governing permissions and  
 \* limitations under the License.  
 \*/  
  
dependencies {  
 compile project(':portability-spi-cloud')  
 compile project(':portability-spi-transfer')

// logging  
 compile("org.slf4j:slf4j-api:${slf4jVersion}")  
 compile("org.slf4j:slf4j-log4j12:${slf4jVersion}")

compile(YOUR DEPS HERE)  
}

1. Add an entry for your extension in root settings.gradle
   * // Instagram auth extension
   * include ':extensions:data-transfer:portability-auth-instagram'
2. META-INF file
   * Create directory for your META-INF file
     + mkdir -p extensions/auth/portability-auth-instagram/src/main/resources/META-INF/services
   * Create the file
     + touch extensions/auth/portability-auth-instagram/src/main/resources/META-INF/services/org.dataportabilityproject.spi.auth.extension.AuthServiceExtension
   * Add a line for the fully qualified name of your AuthServiceExtension
     + org.dataportabilityproject.auth.instagram.InstagramAuthServiceExtension
3. Create the directory for your main code
   * mkdir -p extensions/auth/portability-auth-instagram/src/main/java/org/dataportabilityproject/auth/instagram/
4. Create InstagramAuthServiceExtension extending AuthServiceExtension
   * See class javadoc for AuthServiceExtension for full documentation
   * Return service id
   * Best practices
     + Validate the AuthServiceExtension isn’t validated more than once in a running binary
     + Most providers will interact with their api via a ‘registered’ app providing a client id/key and secret. Those will be stored in a key management system and retrieved via an identifier for the key and one for the secret,
     + In initialize, retrieve the AppCredentialStore
   * Return an AuthDataGenerator for a given transfer type and mode
5. Create an InstagramAuthDataGenerator implementing AuthDataGenerator
   * This class
6. Create the test directory for the InstagramAuthDataGeneratorTest
   * mkdir -p extensions/auth/portability-auth-instagram/src/test/java/org/dataportabilityproject/auth/instagram
7. Create InstagramAuthDataGeneratorTest with stuff

## **Integrate a new Data Type**

* The shared model classes for supported Transfer Data Types are located in the portability-types-transfer[module](https://github.com/google/data-transfer-project/tree/master/portability-types-transfer/src/main/java/org/dataportabilityproject/types/transfer/models)
* The first thing to do is look for any existing model classes that may support your use case or that can be extended to support your use case
* If a new model is required, *TODO: complete this documentation*

## **Integrate a new cloud provider**

*TODO: complete this documentation*