

# SWIFT Resources

---

Version 10g Release 3 (10.3)

- SWIFT RESOURCES..... 3**
- CURRENCY CODES ..... 4
- COUNTRY CODES ..... 5
- IBAN COUNTRY CODES ..... 6
- IBAN STRUCTURE..... 6
- BIC-IBAN DIRECTORIES ..... 7
- BICPlusIBAN Directory*..... 7
- BIC Directory* ..... 10
- UPDATING AND MANAGING RESOURCE FILES..... 13**
- CONVERTING SWIFT DIRECTORY FILES TO RESOURCE FILES ..... 14
- UPDATING CSV FILES ..... 14

## SWIFT Resources

SWIFT Plug-in runtime depends on external resources/classes for validating the following,

- Country Code
- Currency Code
- BIC
- IBAN Country Codes

For instance, every occurrence of BIC in a message instance needs to be validated against the BIC directory. The SWIFT runtime supports class hooks/resources which can be customized depending on the client's requirement.

The primary way to provide a directory of these entities is through resource files. The directory files for the above entities are stored as independent CSV files and made available to the runtime as resource file by packaging it in a Jar and adding it to the classpath. In all cases the resource file should be packaged in a top level directory with in the Jar.

Entity	Resource File Name	Comments
Country Code	swift-country-codes.csv	Mandatory. Bundled with swiftresource.jar
Currency Code	swift-currency-codes.csv	Mandatory. Bundled with swiftresource.jar
BICPlusIBAN	<i>BICPlusIBAN.txt</i>	This is optional. If such a resource is not available BIC codes and IBAN will not be validated.
BIC	swift-bic-codes.csv	This is optional. If such a resource is not available BIC codes are not validated. Only one of BIC or BICPlusIBAN directory must be present

IBAN country codes	swift-IBAN-country-codes.csv	Mandatory. Bundled with swiftresource.jar
IBAN Structure	IBANStructure.txt	Mandatory. Bundled with swiftresource.jar

**Note:**

The SWIFT runtime loads these resource files using getResource() method in java.lang.Class. This assumes that the resource file will be loaded by the same or ancestor class loader of the class loader that loads SWIFT runtime.

If multiple resource files by the same name are available in the classpath one of them will be loaded and used. Avoid such a scenario.

The format (fields in CSV) of each of the above resource file is documented later in this document.

The installation includes a file swiftresources.jar, which contains resource files for country code, currency code and IBAN country codes. The suggested mechanism is to extract this Jar file, [update the resource files](#) and repackage it.

The resource file for BIC can be bundled in a separate Jar and made available to the runtime by adding it to the classpath.

There is also a [utility](#) to convert SWIFT provided files to resource file (CSV) used by SWIFT runtime.

**See Also:**

[Currency Codes](#)

[Country Codes](#)

[IBAN Country Codes](#)

[BIC Directory](#)

[Updating and Managing Resource Files](#)

## Currency Codes

Currency codes must be packaged in a resource file by name swift-currency-codes.csv. This is comma separate value file with the following fields

country name  
currency name  
currency code

number of decimal digits allowed for the currency.

Each line in the file represents a country; the file should not header/trailer line.

**Example:**

```
UNITED KINGDOM,Pound Sterling,GBP,2
UNITED STATES,US Dollar,USD,2
```

The currency codes list was updated based on the list provided in <http://www.iso.org/iso/en/prods-services/popstds/currencycodeslist.html>

This resource is used for validating the currency code that appears in SWIFT messages and for verifying the number of allowed digits for the amounts specified in that currency. More specifically it is used for applying the rules

Currency code must be a valid ISO Currency code (Error code: T52).

The number of digits following the comma must not exceed the maximum number allowed for the specified currency (Error code: C03).

The resource file is also used for supporting the following formula functions.

```
GetCurrencyCode
GetCurrencyDecimals
GetCurrencyName
```

**See Also:**

[Country Codes](#)  
[SWIFT Resources](#)

## Country Codes

Currency codes must be packaged in a resource file by name swift-country-codes.csv. This is comma separate value file with the following fields,

```
Country Name
Country Code
```

**Example:**

```
UNITED KINGDOM,GB
UNITED STATES,US
```

This resource is used for validating the country code that appears in SWIFT messages. Specifically, it is used for applying the rule,

Country code must be a valid ISO country code (Error code: T73).

The resource file is also used for supporting the following formula functions.

```
GetCountryCode(countryName)
GetCountryName(countryCode)
IsISOCountryCode (countryCode)
```

See Also:

[IBAN Country Codes](#)

[Currency Codes](#)

[SWIFT Resources](#)

## IBAN Country Codes

IBAN country codes must be packaged in a resource file by name swift-IBAN-country-codes.csv. This is a comma separate value file with the following fields,

```
ISO country code
Country Name
IBAN code
BBAN Length
```

### Example:

```
AD,Andorra,AD,24
AT,Austria,AT,20
BE,Belgium,BE,16
```

This resource is not used in any validation; it is primarily used to support the following formula functions

isISOIBAN() and other related functions for verifying the allowed length and code.

**See Also:**

[Country Codes](#)

[SWIFT Resources](#)

## IBAN Structure

SWIFT provides a file which contains the details about the structure of IBAN for each country. The file provided by SWIFT is a tab separated file and can be included as a

resource without changes. The latest version (Mar 2008) of this file has been included as resource with name IBANStructure.txt in the product (hence users don't have to do anything).

This resource is not used in any validation; it is primarily used to support the following formula functions

DeriveBICFromIBAN () and other related IBAN functions for decomposing IBAN into its constituent parts.

**See Also:**

[BICPlusIBAN Directory](#)

[SWIFT Resources](#)

## **BIC-IBAN Directories**

SWIFT supports two BIC related directories; BIC+ directory and the BIC+IBAN directory. The former (BIC+) was deprecated in 2008 and is now superseded by BIC+IBAN directory which provides more information. For backward compatibility, designer supports both the BIC+ directory and the BIC+IBAN directory. Both of these are supported as a resource file as well as a user specified provider extension. Since BIC+IBAN is a super set of BIC+ directory all the formula function that were based on BIC+ directory now work with BIC+IBAN as well. But there some new functions such as DeriveBICFromIBAN, which require the presence of BIC+IBAN directory.

Users are encouraged to migrate to the new BIC+IBAN directory. At runtime the application expects only one of these directories to be present.

## **BICPlusIBAN Directory**

BICPlusIBAN is a SWIFT directory that lists institution identifiers recognized by the financial industry, for example, Bank Identifier Codes, CHIPS UIDs, national clearing codes, and IBAN related information. It also provides the names and addresses of the corresponding entities.

BICPlusIBAN is used to identify correspondents and counterparties accurately, and to allocate the correct code when sending messages, thus improving straight-through processing (STP). Initiators of cross-border payments within Europe are required to submit the BIC and IBAN codes to the receiver in order to benefit from reduced payment transactions charges.

The BICPlusIBAN files are available as download packages. You can download BICPlusIBAN: from swift.com, over the Internet: browse to the **BIC downloads** section in **Products & Services > Information Products > BIC Portal**.

Note that BICPlusIBAN directory supersedes BIC Directory which has been deprecated (can still be used) as of release 3.5.

SWIFT runtime supports two different ways of customizing the BIC+IBAN directory.

1. Using BIC+IBAN resource file. The resource file should be the same the file downloaded from SWIFT site and contains a list of BIC and IBAN codes and other related information such as Bank Name address etc.
2. Using a BICPlusIBAN provider class. This user specified class is responsible for providing information about BIC. This class needs to implement an interface which lets the SWIFT runtime to check validity of BIC code and to get additional information about a BIC.

If a BICPlusIBAN provider is available in the classpath it is used in preference to the resource provider. The default option, that is, if no provider class is available, is to use the BICPlusIBAN resource provider which uses a resource file (option 1). If the resource file is missing, BICs and IBANs are not validated.

## Usage

The BICPlusIBAN provider or resource is used for validating the BIC that appears in SWIFT messages and also to cross check BIC and IBAN. Specifically, it is used for applying the following rules.

The BIC/BEI must be a SWIFT registered address, either connected or non-connected (Error code(s): T27, T28, T29, T45)

The BIC must not be a BEI, i.e. must not be of subtype BEID, MCCO, TESP or TRCO (Error code(s): C05).

The provider/resource file is also used for supporting the following formula functions.

```
IsValidBIC(bicCode)
GetBICAddress(bicCode)
GetBICInfos(category)
GetBICName(bicCode)
IsBICNotBEI(bicCode)
DeriveBICFromIBAN(iban)
IsValidBankID(iban)
IsValidBICIBAN(bic, iban)
```

These functions are implemented based on the information provided by BIC+IBAN provider. If there is no custom provider and if the BIC+IBAN resource file is missing, all BICs are treated as valid. The functions that provide additional information like name, address etc return an empty string.

**See Also:**

[BICPlusIBAN Resource File](#)

[BICPlusIBAN Provider Class](#)

[SWIFT Resources](#)

## BICPlusIBAN Resource File

BIC+IBAN resource must be packaged in a resource file by name *BICPlusIBAN.txt* (this file name is case sensitive). This is a tab separated file provided by SWIFT and no changes to it should be made.

**See Also:**

[BICPlusIBAN Provider Class](#)

[BIC/IBAN Directory](#)

[Updating and Managing Resource Files](#)

## BICPlusIBAN Provider Class

This user specified class is responsible for providing information about BIC and IBAN. This hook lets you to completely customize the location or source from which BIC/IBAN related information are loaded. Depending on the size of the BIC+IBAN directory you can also decide on when to load the BIC entries; some of the possibilities are, load at startup, on demand, on demand with the entries cached etc.

The user defined class needs to implement the `com.tplus.transform.runtime.swift.BICPlusIBANProvider` interface and should be made available to the SWIFT runtime service loader mechanism. This interface is used by the SWIFT runtime to check validity of BIC code and to get additional information about a BIC. The implementation class can get this information from any source it prefers. For instance, it is possible to manage the BIC+IBAN directory in a database and the provider class can fetch information for there as and when the SWIFT runtime requires it.

BIC+IBAN Service providers can be installed in an implementation of the Java platform in the form of extensions, that is, jar files placed into any of the usual extension directories. Providers can also be made available by adding them to the application's class path or by some other platform-specific means.

The BIC+IBAN service provider is identified by placing a *provider-configuration file* in the resource directory META-INF/services. The file's name is the fully-qualified binary name of the service's type. The file contains a list of fully-qualified binary names of concrete provider classes, one per line. Space and tab characters surrounding each name, as well as blank lines, are ignored. The comment character is '#' ('\u0023', NUMBER SIGN); on each line all characters following the first comment character are ignored. The file must be encoded in UTF-8.

For example, implement the BICPlusIBANProvider in class `com.bic.DBBICPlusIBANProvider`. Create a file by name `'com.tplus.transform.runtime.swift.BICPlusIBANProvider'` and place it under META-INF/services folder.

This file contains the single line:

```
com.bic.DBBICPlusIBANProvider # Database BIC+IBAN Provider
```

### Implementor's Guide:

3. The BIC and branch code are passed separately to the function `getBICPlusIBANFromBIC`. The implementing class must use "XXX" if branch code is empty.

The normalized BIC must be compared with the BIC in the BIC directory, which typically contains 11 characters BIC code.

Refer to the [API documentation](#) and [BICPlusIBAN provider sample](#) for further details.

### See Also:

[BIC/IBAN Directory](#)

[Updating and Managing Resource Files](#)

## BIC Directory

SWIFT runtime supports two different ways of customizing the BIC directory.

1. Using BIC resource file. The resource file contains a list of BIC codes and other related information such as Bank Name address etc. This is very similar to resource file support provided for other code values (like country, currency etc).
2. Using a BIC provider class. This user specified class is responsible for providing information about BIC. This class needs to implement an interface which lets the SWIFT runtime to check validity of BIC code and to get additional information about a BIC.

If a BIC provider is available in the classpath it is used in preference to the resource provider. The default option, that is, if no provider class is available, is to use the BIC resource provider which uses a resource file (option 1). If the resource file is missing, BICs are not validated.

The BIC provider or resource is used for validating the BIC that appears in SWIFT messages. Specifically, it is used for applying the following rules.

The BIC/BEI must be a SWIFT registered address, either connected or non-connected (Error code(s): T27, T28, T29, T45)

The BIC must not be a BEI, i.e. must not be of subtype BEID, MCCO, TESP or TRCO (Error code(s): C05).

The provider/resource file is also used for supporting the following formula functions.

IsValidBIC(bicCode)  
GetBICAddress(bicCode)  
GetBICInfos(category)  
GetBICName(bicCode)  
IsBICNotBEI(bicCode)

These functions are implemented based on the information provided by BIC provider. If there is no custom provider and if the BIC resource file is missing, all BICs are treated as valid. The functions that provide additional information like name, address etc return an empty string.

Note that [BICPlusIBAN directory](#) supersedes BIC Directory which has been deprecated (can still be used) as of release 3.5.

**See Also:**

[BIC Resource File](#)

[BIC Provider Class](#)

[SWIFT Resources](#)

## **BIC Resource File**

BIC codes must be packaged in a resource file by name swift-bic-codes.csv. This is comma separate value file with the following fields,

BIC Code  
Name of the institution  
Address of the institution  
SWIFT Sub type (BANK, NSWB, BEID etc)

Each line is an entry for a specific BIC code. You can choose to ignore the trailing fields (except the BIC code itself) in each line. Trailing commas are also optional. So if you don't have detailed information about a BIC, you can ignore all the columns except the BIC code itself.

**Example:**

```
BARCKENXBIS,"BARCLAYS BANK OF KENYA, LTD.",BARCLAYS PLAZA BUILDING,BANK  
CITITMMXXX,CITIBANK INTERNATIONAL PLC,FORO BUONAPARTE 16,BANK  
CLAAFRP1XXX,CLAAS FINANCIAL SERVICES,5 AVENUE KLEBER,NSWB
```

**Note:**

If a value in the resource has an embedded comma, it should be properly quoted.

**See Also:**

[BIC Provider Class](#)

[BIC Directory](#)

[Updating and Managing Resource Files](#)

## BIC Provider Class

This user specified class is responsible for providing information about BIC. This hook lets you to completely customize the location or source from which BIC related information is loaded. Depending on the size of the BIC directory you can also decide on when to load the BIC entries; some of the possibilities are, load at startup, on demand, on demand with the entries cached etc.

The user defined class needs to implement the `com.tplus.transform.runtime.swift.BICProvider` interface and should be made available to the SWIFT runtime service loader mechanism. This interface is used by the SWIFT runtime to check validity of BIC code and to get additional information about a BIC. The implementation class can get this information from any source it prefers. For instance, it is possible to manage the BIC directory in a database and the provider class can fetch information for there as and when the SWIFT runtime requires it.

BIC Service providers can be installed in an implementation of the Java platform in the form of extensions, that is, jar files placed into any of the usual extension directories. Providers can also be made available by adding them to the application's class path or by some other platform-specific means.

The BIC service provider is identified by placing a *provider-configuration file* in the resource directory META-INF/services. The file's name is the fully-qualified binary name of the service's type. The file contains a list of fully-qualified binary names of

concrete provider classes, one per line. Space and tab characters surrounding each name, as well as blank lines, are ignored. The comment character is '#' ('\u0023', NUMBER SIGN); on each line all characters following the first comment character are ignored. The file must be encoded in UTF-8.

For example, implement the BICProvider in class `com.bic.DBBICProvider`. Create a file by name '`com.tplus.transform.runtime.swift.BICProvider`' and place it under META-INF/services folder.

This file contains the single line:

```
com.bic.DBBICProvider # Database BIC Provider
```

### Implementor's Guide:

1. The BIC passed to all the functions can be 8 character (4!a2!a2!c) or 11 character (4!a2!a2!c3!c) or 12 character (4!a2!a2!c1!a3!c) BIC code. The implementing class must normalize the BIC before validating it. This is done by,

Adding "XXX" for 8 character BIC codes

Removing the LT Code (9th character) from 12 character BIC codes

The normalized BIC must be compared with the BIC in the BIC directory, which typically contains 11 characters BIC code.

2. Implement `getBICInfo` method first. Rest of the methods can be implemented based on it
3. If one of the fields (Name, Address, BICSubType) is not available the corresponding getter should return an empty string (do not return null).

Refer to the [API documentation](#) and [BIC provider sample](#) for further details.

### See Also:

[BIC Directory](#)

[Updating and Managing Resource Files](#)

## Updating and Managing Resource Files

This section describes how you can,

1. Convert SWIFT provided directory files to CSV resource files
2. Update CSV resource files in `swiftresources.jar`

### See Also:

[Converting SWIFT Directory files to Resource Files](#)

[Updating CSV Files](#)

[SWIFT Resources](#)

## Converting SWIFT Directory files to Resource Files

SWIFT (<http://www.swift.com>) maintains a directory database of BIC, Country code and Currency codes and makes it available to partners. The directory is made available in multiple formats of which SWIFT recommends the tab delimited format.

The SWIFT plug-in includes a resource conversion utility that lets you convert the SWIFT provided files to resource file (CSV) used by SWIFT runtime. Refer to the documentation for this utility for details.

### See Also:

[Updating CSV Files](#)

[Updating and Managing Resource Files](#)

## Updating CSV Files

The swift plug-in installs a file `swiftresources.jar`, which contains resource files for country code, currency code and IBAN country codes. This file is installed under `<installdir>/lib/runtime`. The information contained in these resources is relatively stable and does not require frequent updating.

To update resource files:

1. Extract the Jar file.
2. Modify/replace the one or more CSV files.
3. Repackage the resource files again as `swiftresource.jar`.
4. Replace the file under `<installdir>/lib/runtime` with the updated one.

To quickly update any file in `swiftresources.jar` the following steps need to be done.

1. Open the Jar file with Winzip.
2. Drag and drop the file that needs to be updated to any text editor.
3. Make changes that need to be done.
4. Close the text editor.
5. Switch to the Winzip. Close it. A dialog will be displayed with asking whether to update the archive with the updated file. Click 'Yes'. The file will be updated in the `.jar` file.

### See Also:

[Converting SWIFT Directory files to Resource Files](#)

[Updating and Managing Resource Files](#)  
[SWIFT Resources](#)