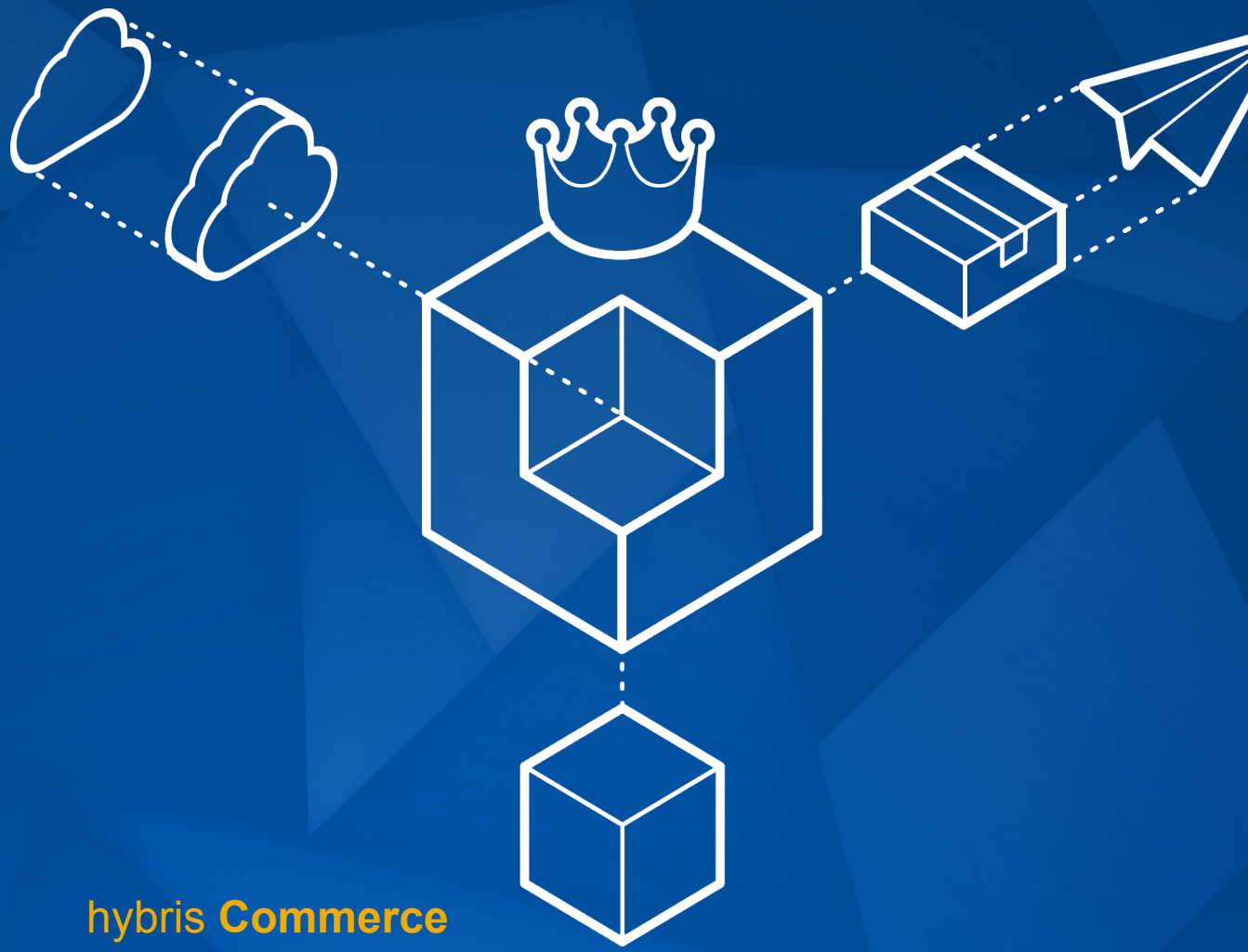
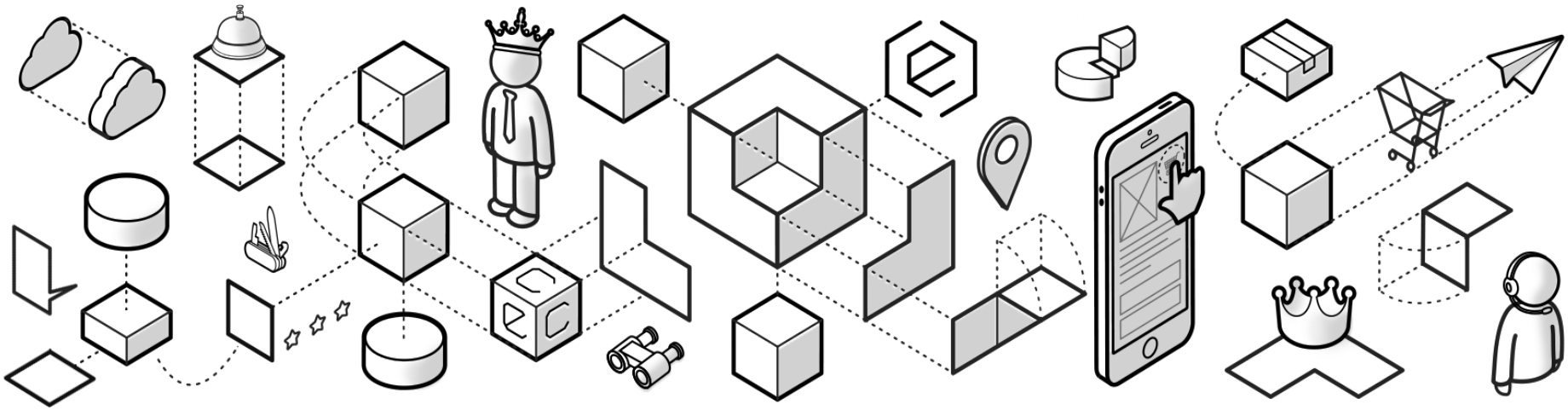
hybris software

An SAP Company

ImpEx



hybris **Commerce**
Developer Training
– Part I



Overview

Overview
Syntax and Examples
Invoking

ImpEx – Overview

- ImpEx is an out-of-the-box import / export framework
- It's an interface between CSV files and the hybris Commerce Suite's Type System
  - you can "import" instances of types from CSV.
  - you can "export" instances of types into CSV.
- You can create, update, export, and remove items

ImpEx – Typical fields of use

- In live operation:
  - to import customer data into a production system
  - to synchronize data with other systems, such as an ERP or LDAP
  - to create backups
  - to update data at runtime
  - can be run from CronJobs
- In migrations:
  - to migrate data from one hybris installation to another
- In development:
  - to import sample data (e.g. on system initialization)
  - to import test data into testing system

ImpEx – Features

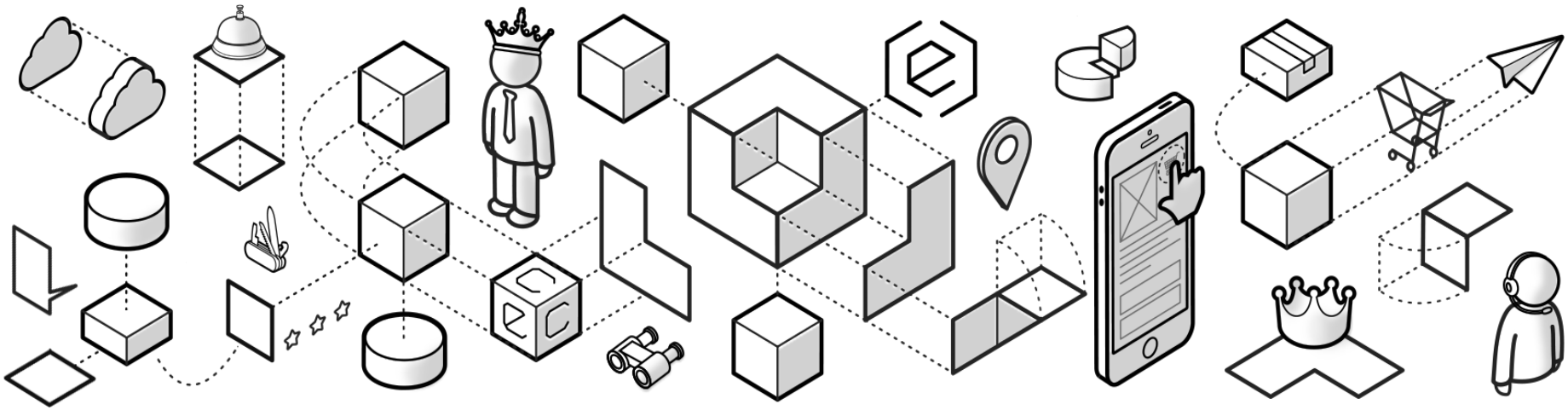
- Impex abstracts from database
  - No table information (deployment)
  - No foreign keys (use “business keys,” which we will discuss in a moment)
- Impex simplifies imports
  - The order of the imported data is irrelevant! (Failed lines are retried)
  - Validation constraints may be disabled

```
impex.legacy.mode=true
```

- ImpEx concerns
  - no transactional imports
  - Performance – use multithreaded imports:

```
impex.import.workers=4
```

- Note: ImpEx does not provide XML import out-of-the-box



Syntax and Examples

Overview
Syntax and Examples
Invoking

Syntax Basics

Header syntax:

```
Operation      itemType; attributes(refAttr)[modifiers];...

INSERT         Product;   code; name[lang=en];
UPDATE         Car;       code[unique=true]; name[lang=en];
INSERT_UPDATE  Customer;  customerID[unique=true]; groups(uid);
REMOVE         Media;     code[unique=true];
```

Data row syntax:

```
;attr1value; attr2value; ...
;Peugeot 403; Columbo's Car;
;FrankColumbo; customergroup;
;P403Pic;
```

Basic syntax example

```
INSERT_UPDATE Promo; code[unique=true]; name[lang=en]; name[lang=fr]; country(code)
;Maranello3; Antarctica Ferrari launch; Lancement Ferrari en Antartique; AQ
;DeLorean_CN; De Lorean China Campaign; Campagne De Lorean en Chine; CN
```

Key points:

- The `code[unique=true]` is so called “key attribute” or “business key”. ImpEx will search for product with code *`Maranello3`* before triggering import. If more than one column is marked as unique, then ImpEx will consider the business key to be multi-column.
- The `[lang=en]` qualifier indicates the language of the value provided. This is only valid for localized attributes, and many languages can be loaded on the same line.
- The header field `country(code)` is a reference to another item using its code (“business key”). In this example, the *country* property of Promo item *`Maranello3`* is a reference to another hybris item, whose code attribute has the value *`AQ`*. Here, hybris will look that item up, and use its PK in the Promo table.

ImpEx Syntax Elements

- Macros
  - Allows aliases to stand in for frequently used statements
- BeanShell, Groovy, and Javascript scripting
  - Allows script to be added to a CSV file.
  - Predefined `hooks beforeEach, afterEach, getLastImportedItem()` etc.
- Translators
  - Implement custom ImpEx logic e.g to import *medias* (binary files).
- Inclusion of data
  - Allows you to split your ImpEx operations over several files.
- Collections and HashMaps:
  - Allows you to use these types as attributes
- Different validation modes for export
  - E.g. the mode “Strict (Re)Import” ensures that the export is re-importable

Catalog example

```
$catalogVersion=catalogVersion(Catalog(id),version)[unique=true]

INSERT_UPDATE Car; code[unique=true]; name[lang=en];
                   unit(code); $catalogVersion

;DB5;Aston Martin DB5; pieces; Default:Staged
;ES1;Lotus Esprit S1; pieces; Default:Online
```

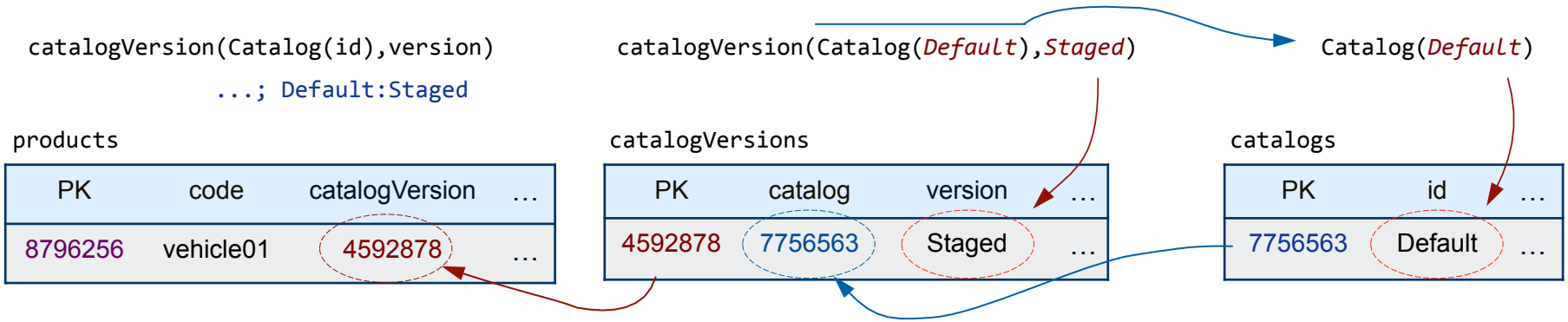
- This example uses a macro, which is substituted verbatim.
- A catalog-aware item like product uses a composite key, since more than one instance can exist (in different catalog versions).
  - The composite key is denoted by having two header fields listed as unique (code and catalogVersion).
  - The catalog version itself uses a composite business key — so we need to reference it using a pair of values.
    - The value pair is separated by commas in the header, and a colon (:) in the data line.

Catalog reference details

```
$catalogVersion=catalogVersion(Catalog(id),version)[unique=true]
INSERT_UPDATE Car; code[unique=true]; name[lang=en]; unit(code); $catalogVersion
;DB5;Aston Martin DB5; pieces; Default:Staged
;ES1;Lotus Esprit LS1; pieces; Default:Online
```

References

- The product item references a catalogVersion item, which is identified using two keys: a catalog reference and a *version* string. The catalog reference, in turn, is identified by an *id* string.



Document Id

Normally, all business keys must be supplied when cross-referencing

```
$catalogVersion=catalogVersion(Catalog(id),version)[unique=true]
INSERT_UPDATE Car; code[unique=true]; name[lang=en]; $catalogVersion
;DB5;Aston Martin DB5; Default:Staged
;ES1;Lotus Esprit LS1; Default:Online

INSERT_UPDATE Employee; uid[unique=true]; car(code, $catalogVersion)
;FrankColumbo; DB5:Default:Staged
```

Use Document ID to simplify cross-reference imports

```
 catalogVersion=catalogVersion(Catalog(id),version)[unique=true]
INSERT_UPDATE Car; code[unique=true]; name[lang=en]; $catalogVersion; &CarRef
;DB5;Aston Martin DB5; Default:Staged; db5
;ES1;Lotus Esprit LS1; Default:Online; es1

INSERT_UPDATE Employee; uid[unique=true]; car(&CarRef)
;FrankColumbo; db5
```

Using Macros and Defaults

```
$prodCat=myCatalog
$version=Staged
INSERT Category;code;catalogVersion(catalog(id),version)
;cars;$prodCat:$version
;convertibles;$prodCat:$version

$catVersion=catalogVersion(catalog(id[default=$prodCat]),
                           version[default=$version])
INSERT Category;code;$catVersion
;cars;
;cars;myCatalog
;cars;myCatalog:$version
;cars;:Staged
```

Every line here is equivalent

Notes

- macros can be used in both header and data rows
- use default values to simplify data rows

Advanced Qualifiers

Use ‘append’ mode to avoid overriding existing references

```
INSERT_UPDATE Employee; uid[unique=true]; groups(uid)[mode=append]
;FrankColumbo; approvers,dummygroup,reviewers
```

Use ‘translators’ for custom interpretation of imported values

```
INSERT_UPDATE Employee;uid[unique=true];@password[translator=PasswordTranslator]
;FrankColumbo;aVeryStrongPassword;

INSERT_UPDATE Media;code[unique=true]; @media[translator=MediaDataTranslator]
;media01;/path/to/my/picture.jpg;
```

Batch update

```
 UPDATE Product [batchmode=true]; itemType(code)[unique=true];status
;Product; approved
```

Importing Classification Attributes

- A translator is necessary to import classification attributes
  - hybris provides the ClassificationAttributeTranslator
  - The translator needs the classification system and version
- For example, importing the classification attribute *trait* :

```
UPDATE Product;code[unique=true];
               @trait[system='ClassificationSystem',version='1.0',
                      translator=de.hybris.platform.catalog.jalo.classification.
                                 impex.ClassificationAttributeTranslator];
```

- OOTB, hybris normally uses macros for this:

```
$sysName=ClassificationSystem
$sysVer=1.0
$translator=de.hybris...ClassificationAttributeTranslator
$clAttrModifiers=system=$sysName,version=$sysVer,translator=$translator
$capacityFeature=@capacity[$clAttrModifiers]
$wetbarFeature=@wetbar[$clAttrModifiers]

UPDATE Car;code[unique=true];$capacityFeature;$wetbarFeature
;car01;4;vodka,bourbon
```

Shortcut for Importing Classification Attributes

- hybris defines a shortcut for importing classification attributes
  - Defined OOTB in the platform/resources/advanced.properties as a regular expression substitution string

```
impex.header.replacement.<priority>=<src pattern> ... <target pattern>
```

```
impex.header.replacement.1=
   C@(\\w+) ...
   @$1[system='\\$systemName', version='\\$systemVersion',
   translator='de...impex.ClassificationAttributeTranslator']
```

- Use in your impex file:

```
$systemName=MySys
$systemVersion=1.0
INSERT_UPDATE Product; code[unique=true]; C@attr1; C@attr2 ; ...
```

ImpEx Script For Export

- Specify the target file:

```
"#%beanshell% impex.setTargetFile( ""Product.csv"" );"
```

- Specify the attributes to be exported via an ImpEx header:

```
INSERT_UPDATE Product; code[unique=true]; description[lang=en];
                       name[lang=en]; unit(code)
```

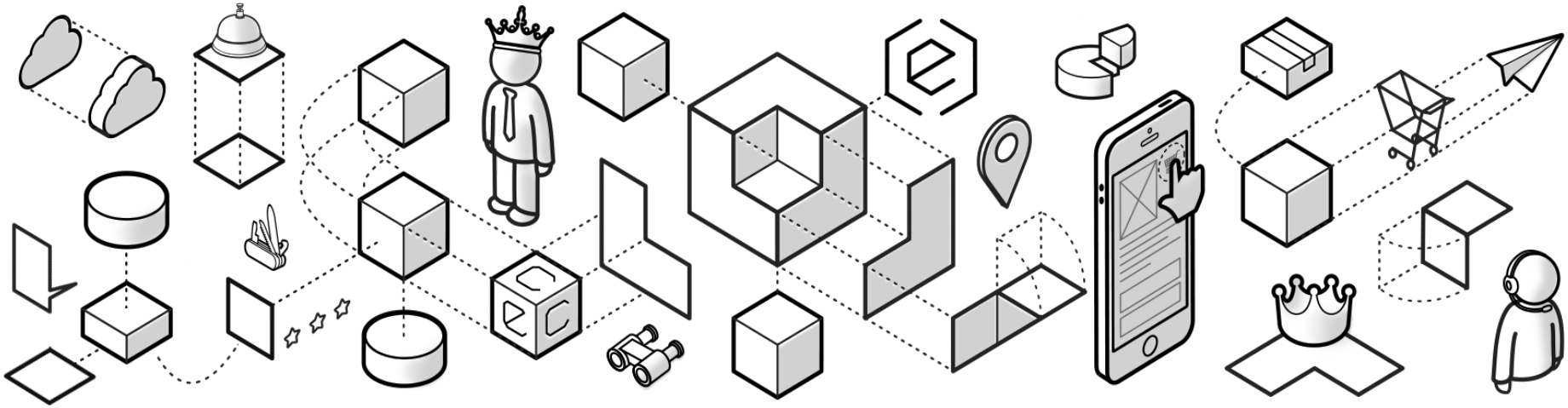
  - You can use the same header for re-import.
  - Consider using the Script Generator feature of the hMC

- Full export

```
"#%beanshell% impex.exportItems( ""Product"" , false );"
```

- Selective export

```
"#%beanshell% impex.setTargetFile( ""Product.csv"" );"
INSERT_UPDATE Product; code[unique=true]; name[lang=en]
"#%beanshell% impex.exportItemsFlexibleSearch(
   ""select {pk} from {Product} where {code} like '%happy%'"");"
```



Invoking

Overview
Syntax and Examples
Invoking

Where Can You Launch an Import?

- In the hybris Administration Console
  - Test area for ImpEx scripts
  - Multiple files cannot be imported by a single ImpEx script
  - No external data is allowable
  - Limitted configuration possibilities
- In the hybris Management Console (hMC)
  - Create an `ImpExImportCronJob`
- Via the API
  - You can use the `ImportService`

Where Can You Launch an Export?

- In the hybris Administration Console
  - Test area for ImpEx scripts
- In the hybris Management Console
  - Select search results and export them via the context menu
  - Create an `ImpExExportCronJob`.
- Via the API
  - Use the `ExportService`
  - Create an `ImpExExportCronJob`

ImpEx Import in hAC

Exercise 4