Development of Service Setting Migration Framework for CWS

Author	岩根　周平，
Updated	2013-07-01 16:15:45
Created	2013-05-12 18:02:58

Outline

1. Providing function for setting data migration to reduce our customers' setting cost
Most of our customers have a test environment and a production environment. Now they have to config the setting again in production environment or use the DB copy tool to migrate the setting data after testing in test enviroment.
CWS created CWSRegistry framework which can be used for data migration. However, CWSRegistry save all data into one DB table which makes the CWSRegistry become slow with the increasing number of data records. We need a new way to implement the data migration function.
2. New function for setting data migration can be used in 2 scenarios:

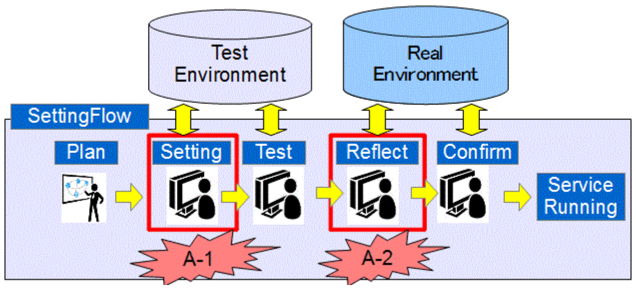


chart 1-1 service setting scenarios

A-1) Initialize service setting
For example: import recommended service settings to customer's environment.
A-2) Migrate settings between multiple environments
Migrate service settings between test environment and production environment when administrators change the settings of CWS or upgrade to new version.
By implementing functions in this catalog, we can reduce our customers' setting cost
3. Migration requests from our customers
This is the list of inquiries from our customers about setting migration between test and production environment from catalog CWS4.7. Search key words in @SUPPORT: 移行 (migration)　テスト(test)　本番(production)
The column named <implemented> means the migration function has been provided by CWS or not.

Ranking	Function	Result records	Implemented
1	Form setting[フォーム設定]	9	
1	Performance review[人事考課]	9	
3	Dictionaly setting[ディクト設定]	8	
4	Report setting[帳票設定]	6	other function
5	Self assessment[自己申告]	5	
5	Service option setting[サービスオプション]	5	
7	personal information [個人情報]	4	
7	Year-end adjustment[年末調整]	4	
9	WorkFlow[ワークフロー]	3	
9	Access priviledge[アクセス権]	3	
11	Boss and subordinate setting [上司部下設定]	2	○
11	Common criteria[共通条件]	2	
11	Mail setting[メール設定]	2	
11	Try setting[トレイ設定全般]	2	

chart 1-2 inquiries about setting migration

From CWS4.7, we created CWSRegistry framework to implement the migration function. However, currently, only a few settings have been implemented by CWSRegistry because of the performance problem and the cost of changing data structure.

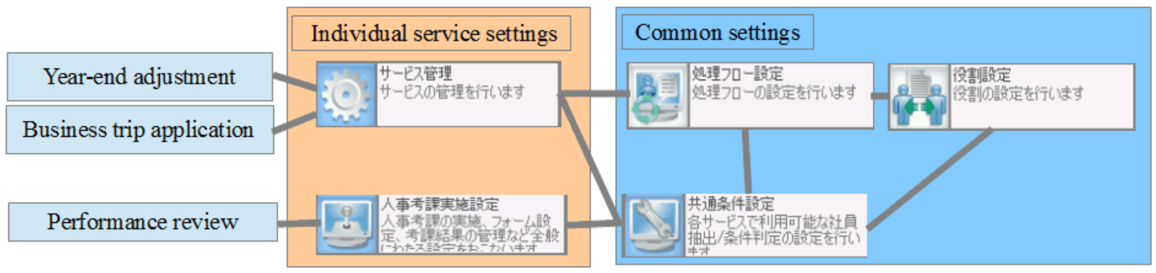
The goal of this catalog is to provide a new way to implement all the setting migration requests listed in chart 1-2.

4. New framework for data migration
To reduce the developing and maintenance cost for implementing setting migration, we create a framework which can be used in migrating DB tables. Basically, we can implement data migration function by using this framework. And new framework can work with CWSRegistry framework. Therefore, we can use both CWSRegistry framework and new framework.
The framework provide following merits:
1) save setting item and its associated tables in Java
Developers can write code for relationship of setting and its associated DB tables in a Java object. It is easy for developers to know the whole relation from Java than DB.
2) provide common DAO for selecting and inserting datas
In most of cases, developers do not need to write SQL for data migration.

Business Investigation

1. Current problems of setting migration
In this section, we will talk about the tools that used by our customers and constants for setting migration and the ideal unit of setting migration for our customers.

1.1 current DB migration tool
Currently, our consultants are using DB copy tools which can copy whole DB schema or a DB table from one DB to another to help our customers migrate their settings.
There are two different flows that customers migrate their settings:
A. At installation of CWS
1) Set up service settings in a test environment
2) Copy all database tables from the test DB to production DB
3) Clean up the application data in production DB
B. Version up or modify service settings
1) Copy all database tables from production DB to test DB
2) Modify service setting and test in test environment
3) Copy some database tables from test DB to production DB
Problems of using DB copy tools on flow B
1) Can only migrate table or DB schema
DB copy tools can migrate the whole settings. However, in most cases, our customers want to migrate only a specific service setting or a workflow setting.
2) Affection on the bussiness
To use DB copy tools, our costomers have to stop their AP servers which would affect their business.
1.2. Setting data migration unit
1) ideal unit of data migration is that our customer's business can run immediately after migration.
When starting to use a new service, our customers config and test the service in test environment and then migrate the service configuration to production environment. If we can make our customers able to use the service without any configuration after the migration, it would reduce our customers' setting cost.
2) Current CWS's service structure.
CWS service setting contains individual service settings (Service Manager page) and common settings (Workflow, Common Criteira) which can be used in different services.


Individual service settings
Common settings
Year-end adjustment
Business trip application
Performance review
サービス管理
サービスの管理を行います
人事考課実施設定
人事考課の実施、フォーム設定、考課結果の管理など全般にわたる設定をおこないます
処理フロー設定
処理フローの設定を行います
役割設定
役割の設定を行います
共通条件設定
各サービスで利用可能な社員抽出/条件判定の設定を行います
chart 2-1 CWS's services structure.

3) two different units of setting migration
a. service settings and related common settings
Migrate related common settings when our customers start a new service and new related common settings, especially at CWS installation stage.
b. migrate settings according to setting page (Service Manager, Workflow, Common Criteria)
Migrate different kinds of settings individually. For example, after configurating and testing the workflow, migrate the workflow to production environment on workflow setting page.

2. Research on data migration tools
CJK and CSR have already provided the data migration tools. In CWS there is also CWSRegistry framework which can be used for data migration. Followings are the functions that provided by those tools or framework.
-Functions-
2.1 Backup setting data before importing
1) Function outline

Ask users to backup the setting data which will be overwritten by the import file
2) Merits
When error happens, users can recover the settings by the backup file
3) Implemented by
[CJK multi copy], [CSR multi copy], [CWSRegistry]
2.2 Migration log function
1) Function outline
Write logs when exporting and importing setting data
2) Merits
Users can show log and roll back by migration logs.
3) Implemented by
[CJK multi copy], [CSR multi copy]
-Reduce human error-
2.3 Confirm import data information
1) Function outline
Before importing, users will be able to confirm the information of import setting file. For example, exported environment name, export time and file comment. The file comment can be written by exporting operator.
2) Merits
Users can confirm whether the file is right or not before importing operation
3) Implemented by
[CSR's multi copy]
2.4 Changed parts confirmation function
1) Function outline
After importing, users are able to check changed parts before reflecting.
2) Merits
Usesr can confirm whether the file is right or not before importing operation.
3) Implemented by
[CJK's multi copy], [CSR's multi copy], [CWSRegistry]
-Usability-
2.5 Choose data migration process
1) Function outline
Users can choose migration option from merging, adding or overwriting
2) Merits
More flexible when migrating setting data
3) Implemented by
[CJK multi copy], [CSR multi copy], [CWSRegistry]
2.6 Multi units data migration function
1) Function outline
Be able to migrate multi units in one migrate operation
For example, when migrating service setting, users can choose several associated services and migrate at the same time.
2) Merits
Reduce users' operation cost.
3) Implemented by
[CJK multi copy], [CSR multi copy]

User Investigation

1. Problem example of current CWSRegistry

Customer	Nanano prefecture [長野県]
@Support no	1187437
Case for	Performance problem when importing [Boss Sub Setting]
Business Background	Our customer use CSV to import [Boss Sub Setting] for their performance measurement.
Real problem	
When our customer tried to import a csv file which contains 15,000 records, response speed degraded. It was caused by a flaw of CWSRegistry framework.	

2. Migrate setting data between environments

	KDDI CORPORATION [KDDI株式会社様]

Customers	Intelligence, Ltd. [株式会社インテリジェンス様] Mitsubishi Electric Corporation [三菱電機株式会社様] Kirin Brewery Company, Limited [麒麟麦酒]
Detail	
In those companies, they kept two environments: test environment and production environment. Currently, there are no way to transport setting data between two environments. Because the setting time is limited in usual cases, setting mistakes sometimes occur. And it is a waste of time for our customers configuring the same setting both in test environment and production environment.	

Function Abstract

1. Setting data migration function
【purpose】Reduce setting cost
【specification】Users can export/import setting data
1.1 Data migration unit.
1) Service settings and related common settings
2) Migrate settings according to setting page (Service Manager, Workflow, Common Criteria)
Now, CWS are converting setting page from Company Client to web page. We will make those parts can be migrated first. The detail items that can be migrated are listed in implementation plan.
2.Error handling when migrating setting data
When migrating setting data, we need to consider about the data consistency, error handling, etc to make the migration be executed successfully. We will provide following functions to ensure the data migration operation.
-Functions-
2.1 Backup setting data before importing
1) Function outline
Ask users to backup the setting data which will be overwritten by the import file
2) Merits
When error happens, users can recover the settings by the backup file
-Reduce human error-
2.2 Confirm import data information
1) Function outline
Before importing, users are able to confirm the information of import setting file. For example, exported environment name, export time and file comment. The file comment can be written by exporting operator.
2) Merits
Users can confirm the file is right or not before doing the import operation.
2.3 Confirm import result
1) Function outline
After importing, show import results, failed settings and reason.
Show information to notice users if there are failed settings that need to be configed.
2)Merits:
Prevent an omission of setting.
-Usability-
2.4 Multi units data migration function
1) Function outline
Be able to migrate multi units in one migration operation. For example, users can choose several associated services and migrate at the same time.
2) Merits
Reduce users' operation cost.

2.5 Choose data migration type
1) Function outline
Users can choose migration type for workflow and common criteria setting.
When importing, users can choose to add new setting for workflow and common criteria setting or merge the setting with the existed settings.
2) Merits
More flexible when migrating setting data.
3. Our customers do not need to stop APs when migrating the setting data.
When using DB copy tools to migrate the setting data, our customers have to stop the APs to avoid the users' operation. Stopping the APs will affect our customers' business.
When users use the new way to migrate settings, they do not need to stop the servers.

Implementation Plan

1. Implementation priority

From our customers' requests, we can know our customers want to migrate the service settings in Service Manager most.
◯ : will implement in this term
△※part : Implement part of the service

Ranking	Function	Implemented	Implement plan
1	Performance review [人事考課]		
1	Form setting [フォーム設定]		◯
3	Dictionary setting [ディクト設定]		△※part
4	Report setting [帳票設定]	other function	
5	Self assessment [自己申告]		
5	Service option setting [サービスオプション]		◯
7	personal information [個人情報]		
7	Year-end adjustment [年末調整]		△※part
9	WorkFlow [ワークフロー]		△※part
9	Access privilege [アクセス権]		◯
11	Boss and subordinate setting [上司部下設定]	◯	
11	Common Condition [共通条件]		
11	Mail setting [メール設定]		
11	Try setting [トレイ設定全般]		

chart 5-1 implementation priority

When CWS v5.5, some functions of Service Manager will be converted to Web application. Administrators can confirm those functions on Web page. Therefore, we decide to make those functions can be migrated. List of setting items that can be migrated:

Japanese Name	Data migration	English name
再認証詳細設定	◯	Re-Authentication Setting
検索画面設定	◯	Search View Setting
アクセスログ設定	◯	Access Log Setting
アクセス権の設定	◯	Authorization Setting
帳票設定	◯	Ledger Sheet Setting
申請書印刷設定	◯	Application Print-out Setting
処理フロー設定	◯	Process Flow Setting
メール文面設定	◯	Mail Content Setting
フォーム設定	◯	Form Setting
コンボボックスの編集	◯	Combo Box Editing
サービス設定	◯	Service Setting
サービスの開始	X	Service Start
サービスの停止	X	Service Stop
タイムアウト設定	◯	Time-out Setting
添付ファイル設定	◯	Attachment File Setting
添付書類設定	◯	Attachment Document Setting
発令内容フィルタリング	◯	Announcement Contents Filter
滞留基準日設定	◯	Retention Base Date Setting
経路自動計算設定	◯	Pay Statement Headder Editing
経路分割設定	◯	Commuting Type Setting
経路検索画面設定	◯	Duty-trip Purpose Setting
明細ヘッダ編集	◯	Traffuc Expense Setting
通勤タイプ設定	◯	Route Auto-Calc Setting
交通費各種設定	◯	Route Search View Setting

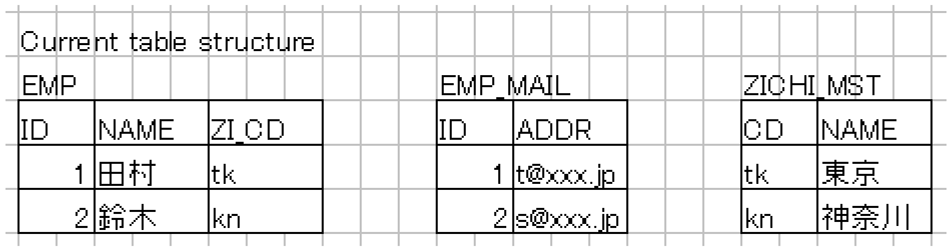
chart 5-2 setting items that can be migrated

Only migrate the setting items which can be configured on Service Manager page.

2. Problems of current CWSRegistry framework

2.1 performance problem of CWSRegistry framework

If we use CWSRegistry to manage service settings, all the pages that use CWSRegistry will become slow.
Reasons:
A. CWSRegistry changed "horizontal" data to "vertical"
Saving same information, the number of data records will increase after using CWSRegistry framework
B. Current CWSRegistry save all data into one DB table
This will make all the functions which use CWSRegistry become slow with the increasing number of data records



Current table structure

EMP

ID	NAME	ZI_CD
1	田村	tk
2	鈴木	kn

EMP_MAIL

ID	ADDR
1	t@xxx.jp
2	s@xxx.jp

ZICHI_MST

CD	NAME
tk	東京
kn	神奈川

chart 5-3 tables before using CWSRegistry

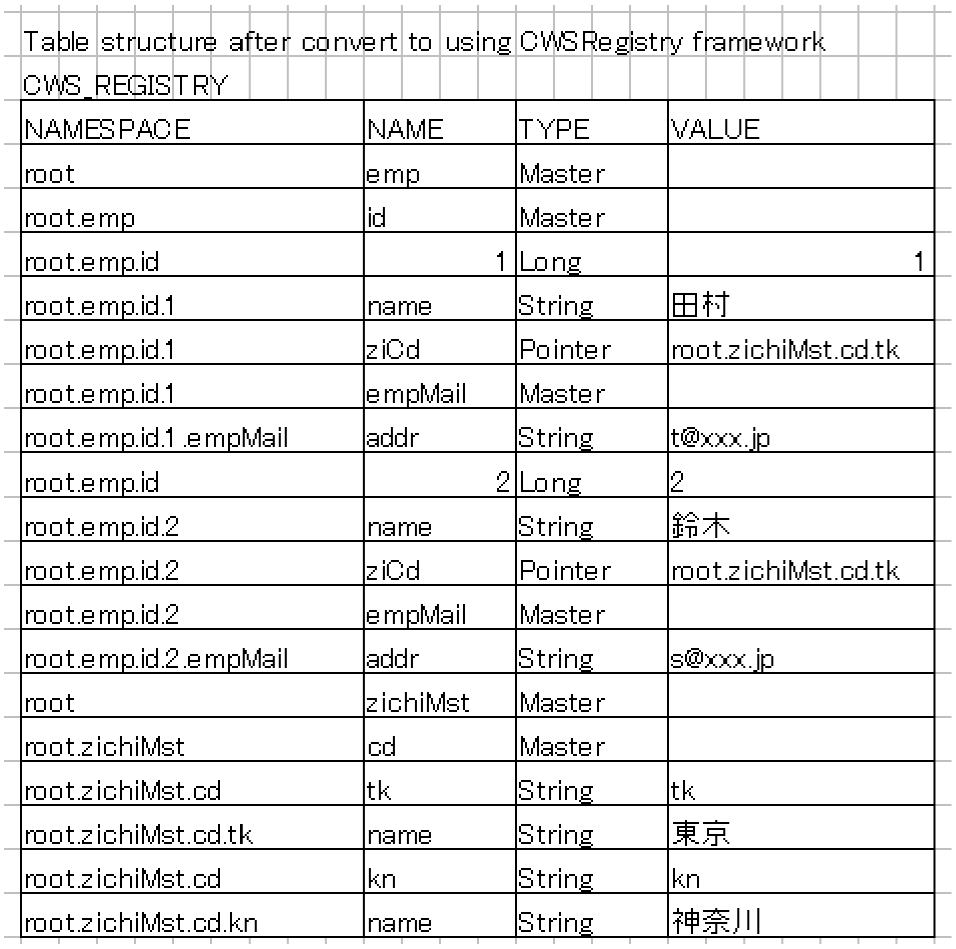


Table structure after convert to using CWSRegistry framework

CWS_REGISTRY

NAMESPACE	NAME	TYPE	VALUE
root	emp	Master	
root.emp	id	Master	
root.emp.id	1	Long	1
root.emp.id.1	name	String	田村
root.emp.id.1	ziCd	Pointer	root.zichiMst.cd.tk
root.emp.id.1	empMail	Master	
root.emp.id.1.empMail	addr	String	t@xxx.jp
root.emp.id	2	Long	2
root.emp.id.2	name	String	鈴木
root.emp.id.2	ziCd	Pointer	root.zichiMst.cd.tk
root.emp.id.2	empMail	Master	
root.emp.id.2.empMail	addr	String	s@xxx.jp
root	zichiMst	Master	
root.zichiMst	cd	Master	
root.zichiMst.cd	tk	String	tk
root.zichiMst.cd.tk	name	String	東京
root.zichiMst.cd	kn	String	kn
root.zichiMst.cd.kn	name	String	神奈川

chart 5-4 after converting tables to cws_registry table

For example, chart 5-5 and 5-6, if there are 10 records in cws_registry table, it will cost 1 to select. After we move dict and item settings to cws_registry table, the number of data records will be increased by 10,000,000 ˜ 100,000,000, and the select time will increase to 7 ˜ 8. The more the data records there are in cws_registry table, the more time our system cost in selecting data.

Table	select condition	number of Records	number of column	Increment of records in cws_registry table
item	schem = root	392,445	34	12,950,685
dict	schem = 'root' and lang = 'ja'	1,031,974	9	9,287,766

chart 5-5 simulation of convert current tables to cws_registry table.

select times /Number of records	1	5	10	15	20
1	1	5	10	15	20
:	:	:	:	:	:
100,000	5	25	50	75	100
1,000,000	6	30	60	90	120

10,000,000	7	35	70	105	140
100,000,000	8	40	80	120	160
1,000,000,000	9	45	90	135	180

chart 5-6 simulation of oracle'sO(calculation amount)[log n]

2.2 Developing risk of current CWSRegistry.

In most cases, if we want to make some setting data able to be migrated by CWSRegistry, we have to convert the old data structure to CWSRegistry, and modify the logic both for setting page (used by administrator) and application page (used by common users). Also we need to drop old service setting tables and move all setting data to cws_registry table.

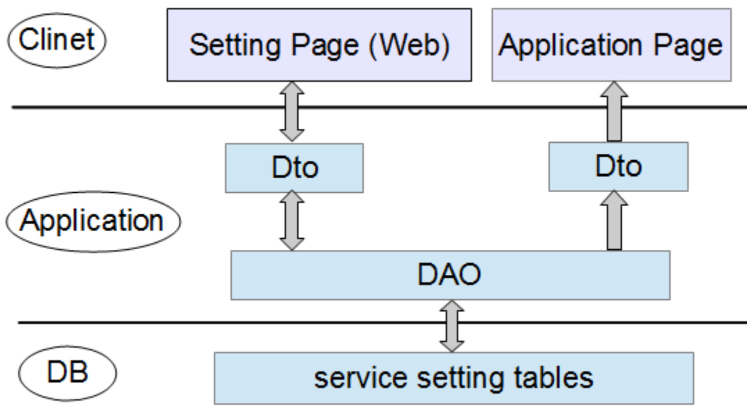


chart 5-7 current data flow for settings and application page

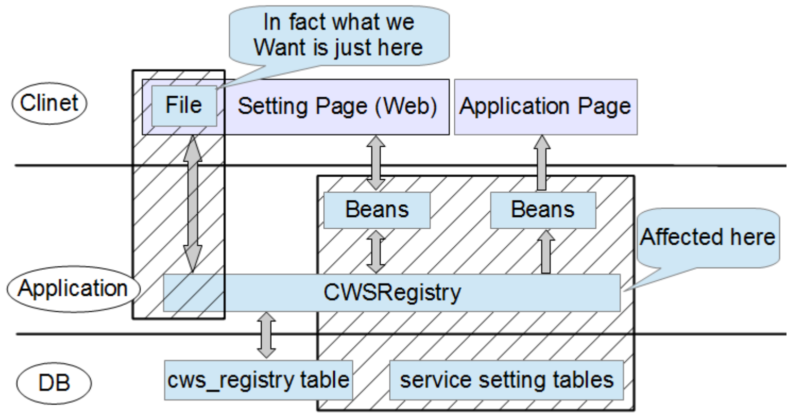


chart 5-8 data flow after converting setting data to CWSRegistry

However, the cost for changing data structure and modifying logic of setting page and application page will be; high and it will be easy to cause degradation.

Conclusion: can not save and migrate all service setting data by CWSRegistry framework

Because of data amount of the setting data, we can not use CWSRegistry framework to save and migrate setting data. We need to create a new way to migrate setting data.

3. Introduction of Data Migration Framework

DB copy tools only can move tables from one DB to another which can achieve the goal of setting data migration.

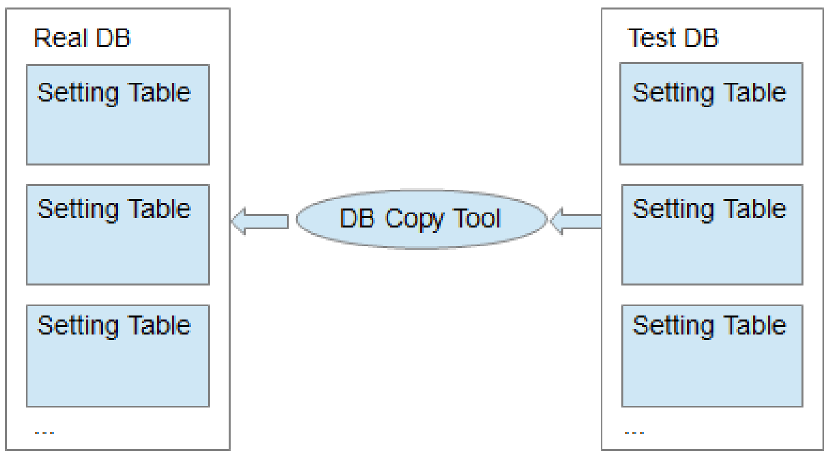


chart 5-9 migrate DB tables by DB copy tool

Like what we mentioned in Business Investigation, our customers want to migrate a unit of setting items, such as a service or a workflow. If we can migrate only the setting data which are associated with the unit, we can meet our customers' demand of setting migration.

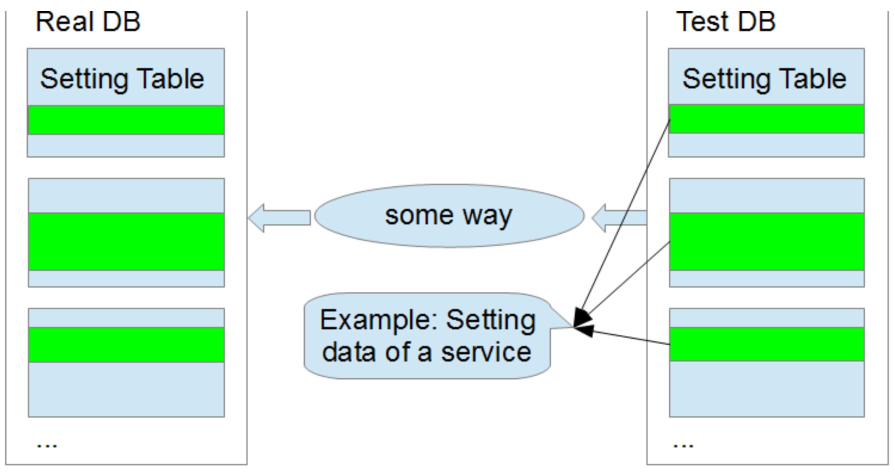


chart 5-10 migrate only the setting data of a given unit

The ideas of our new framework:

1) keep the relationship of the unit and the info of associated tables

For example, service setting data are saved in TEMP_SERVICE, TEMP_ITEM, TEMP_INPUT... tables, we keep the keys which can be used for selecting the setting data of a given unit. Therefore, when we want to export the setting data of the given unit, we can know what and how to retrieve the setting data. The following example, when exporting the setting data of a service, we can use the given schem=root, namespace=root.cws.wage to retrieve setting data from SERVICE table. (SQL: select * from SERVICE where schem='root' and namespace='root.cws.wage')

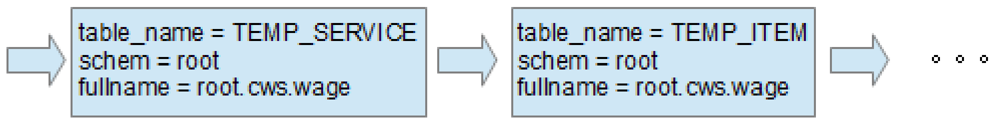


chart 5-11 example: service setting and related tables

2) export DB data which is associated with the migration unit to files

① Request to export given service setting (schem and namespace)

② Search the relation of the service and associated DB tables

③ Select data from the associated tables

④ Export the data to file

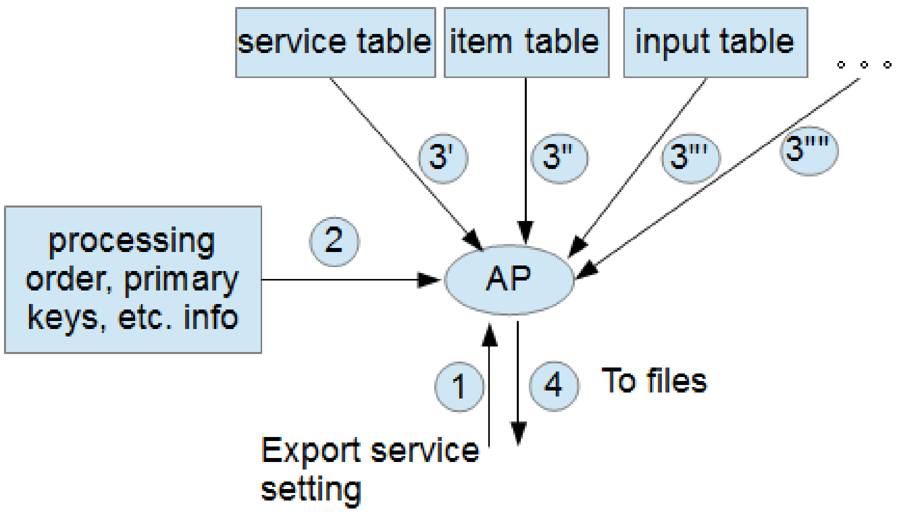


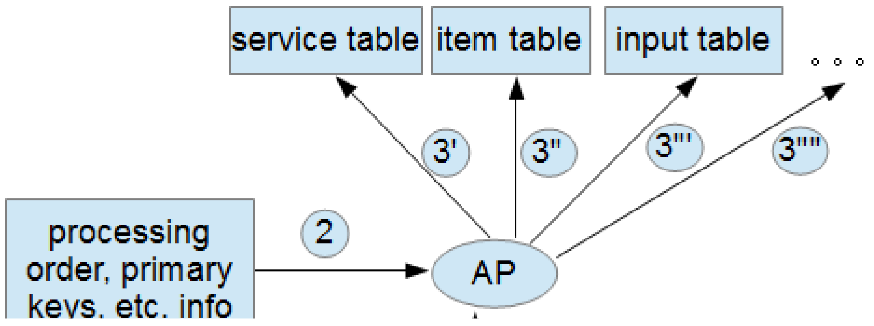
chart 5-12 export setting data according to the RELATION info

3) import the files to another DB

① Import the file to AP

② Search the relation of the unit and associated DB tables

③ Analyse the file and update the DB tables



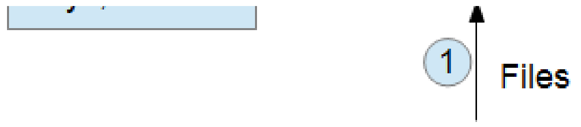


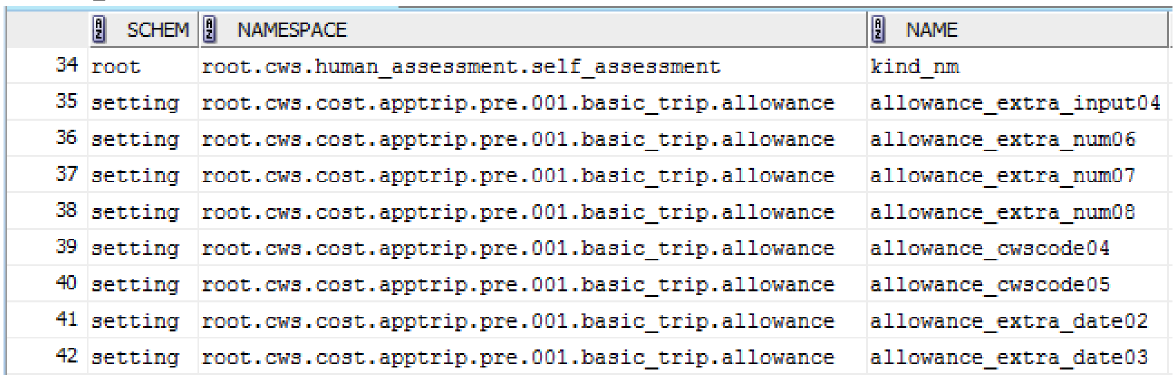
chart 5-13 import setting data saccording to the RELATION info

4. Research on DB structure of service setting

The following parts will introduce the data structure of current DB tables for saving service settings and the problems that we need to solve when we create framwork to implement data migration function.

4.1 Using schema, namespace and service name as the primary keys

Most of the tables which are saving service setting data are using schema, namespace and service name as the primary keys. For example, we can use SCHEM and NAMESPACE to retrieve item settings of an given service from TEMP_ITEM table.



	SCHEM	NAMESPACE	NAME
34	root	root.cws.human_assessment.self_assessment	kind_nm
35	setting	root.cws.cost.apptrip.pre.001.basic_trip.allowance	allowance_extra_input04
36	setting	root.cws.cost.apptrip.pre.001.basic_trip.allowance	allowance_extra_num06
37	setting	root.cws.cost.apptrip.pre.001.basic_trip.allowance	allowance_extra_num07
38	setting	root.cws.cost.apptrip.pre.001.basic_trip.allowance	allowance_extra_num08
39	setting	root.cws.cost.apptrip.pre.001.basic_trip.allowance	allowance_cwscode04
40	setting	root.cws.cost.apptrip.pre.001.basic_trip.allowance	allowance_cwscode05
41	setting	root.cws.cost.apptrip.pre.001.basic_trip.allowance	allowance_extra_date02
42	setting	root.cws.cost.apptrip.pre.001.basic_trip.allowance	allowance_extra_date03

chart 5-14 example of TEMP_ITEM table

Most of the service setting data can be exported by schem, namespace and service name.

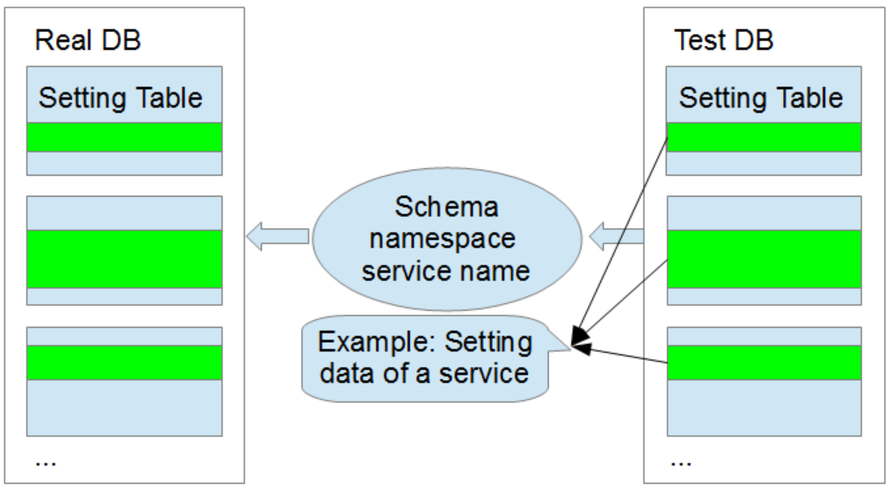
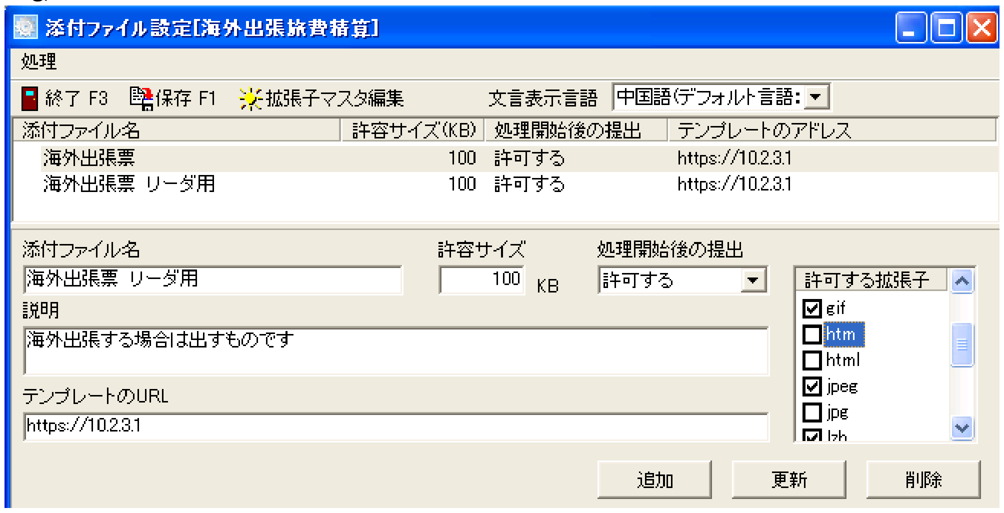

Real DB
Setting Table
...
Schema namespace service name
Example: Setting data of a service
Test DB
Setting Table
...

chart 5-15 use SQL to export setting data of given schem, namespace and service name

4.2 Auto increase column in setting table

For some tables, they have auto increase column which we need to consider when importing the setting data to DB.

For example, ATTACHFILE_MST table which is used to save the attachfile info of [Attachment Setting] in service setting, the FILE_ID column will be increased when users add a new attachment file in the setting.



文言表示言語 中国語(デフォルト言語:

添付ファイル名	許容サイズ(KB)	処理開始後の提出	テンプレートのアドレス
海外出張票	100	許可する	https://10.2.3.1
海外出張票 リーダ用	100	許可する	https://10.2.3.1

添付ファイル名: 海外出張票 リーダ用

許容サイズ: 100 KB

処理開始後の提出: 許可する

説明: 海外出張する場合は出すものです

テンプレートのURL: https://10.2.3.1

許可する拡張子:
- ✓ gif
- ☐ htm
- ☐ html
- ✓ jpeg
- ☐ jpg
- ✓ lzh

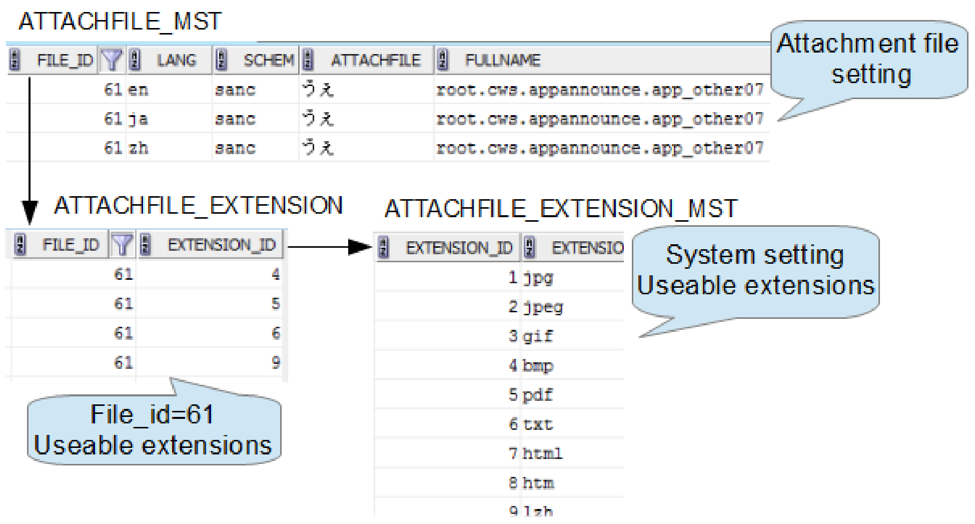
chart 5-16 screen shot of service attachment setting

FILE_ID	LANG	SCHEM	ATTACHFILE	FULLNAME

61	ja	sanc	うえ	root.cws.appannounce.app_other07
61	en	sanc	うえ	root.cws.appannounce.app_other07
61	zh	sanc	うえ	root.cws.appannounce.app_other07

chart 5-17 example of ATTACHFILE_MST table

4.3 Auto increase column is associated with another table
For some tables, when importing the setting data into DB, we need to rebuild the relationship of the auto increase column with other tables.
For example, [Attachment Setting], we can also config the extension types for each attachment. The relationships of attachment and it's extensions are saved in another table named ATTACHMENT_EXTENSION. Therefore, if the FILE_ID is changed when importing the setting data, we also need to rebuild the relationship of FILE_ID and EXTENSION_ID in ATTACHFILE_EXTENSION table.



FILE_ID	LANG	SCHEM	ATTACHFILE	FULLNAME
61	en	sanc	うえ	root.cws.appannounce.app_other07
61	ja	sanc	うえ	root.cws.appannounce.app_other07
61	zh	sanc	うえ	root.cws.appannounce.app_other07

FILE_ID	EXTENSION_ID
61	4
61	5
61	6
61	9

EXTENSION_ID	EXTENSIO
1	jpg
2	jpeg
3	gif
4	bmp
5	pdf
6	txt
7	html
8	htm
9	lzh

chart 5-18 example of updating the associated tables

4.4 About TEMP table
There are two kinds of tables to save the setting data for some setting items in service setting, TEMP table and real table. Currently, only service basic setting (e.g, service name, service description, etc.), Re-Auth setting, Form setting, Access Log setting, Timeout seting and Access Privilege have TEMP tables. When user saves the changes in Service Manager, the changes will be saved into TEMP table. When users click <Reflect Button>, the changes will be copied to real table.
The data structure of TEMP table and real table for service manager is same.

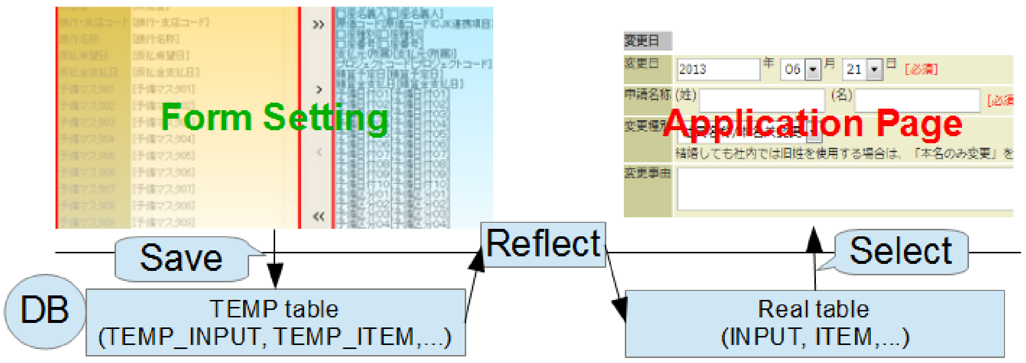

Form Setting
Application Page
Save
Reflect
Select
DB
TEMP table (TEMP_INPUT, TEMP_ITEM,...)
Real table (INPUT, ITEM,...)

chart 5-19 relation of Real table and TEMP table

4.5 Should not update some columns when migration
Some columns are managed by the system and can not be changed by the user. For example, the ORD column in SERVICE table means the order of the service when showing the service list. Those kinds of columns should not be overwritten when importing the setting data.

4.6 Migrate associated setting items
When migrating service setting, there are also setting items in the service kind node. We need to move the settings in the parent node of the service at the same time. For example, in Commute Application Setting, the setting items such as Service Setting and Application Route Auto Calculation Setting can affect the application page. In this case, we need to migrate those settings when migrating the service settings.

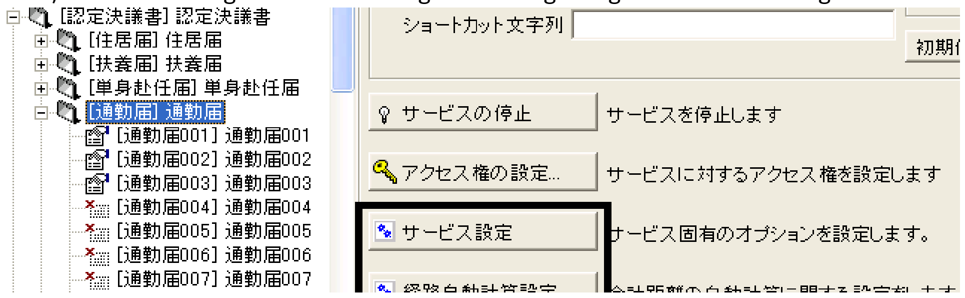


chart 5-20 migrate the setting items on the service type

5. Framework design for Data Migration

There are three ways to insert the new setting data to DB.
1) Update the data records by given primary key-values
2) Delete the data records in DB and insert the data from file
3) Check if the data exists; update by given column and values if data exists; insert the data if the data does not exist

To make developers can achieve data migration easily, we make the following framework by which developers only need to define the way of exporting and importing tables.

5.1 Classes which are used to save the configuration for setting tables

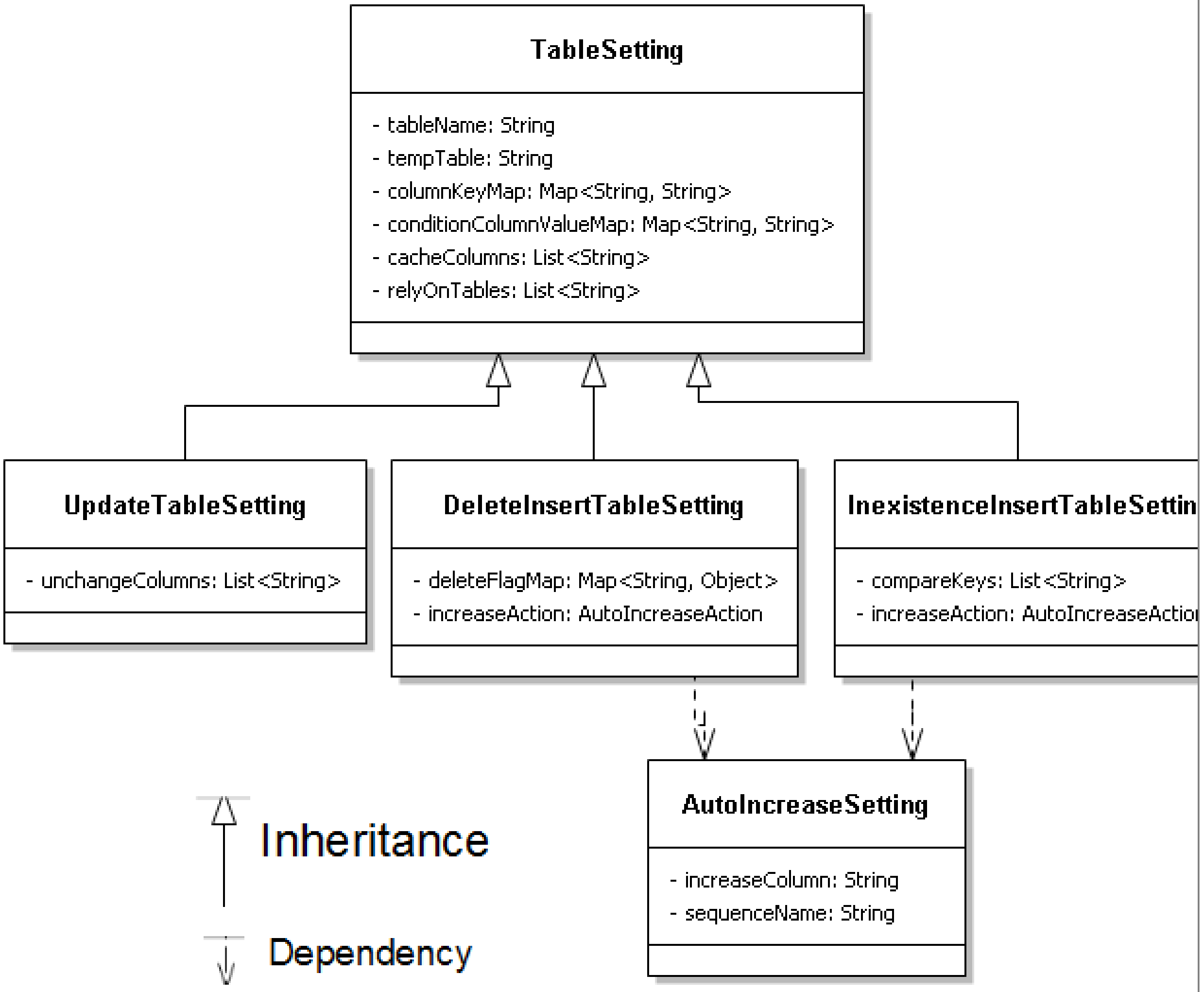
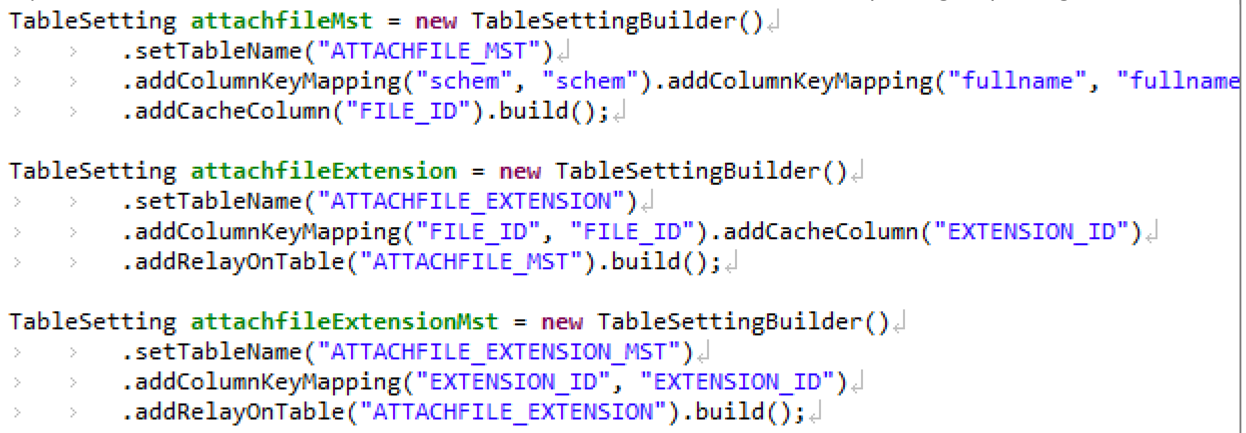

TableSetting
- tableName: String
- tempTable: String
- columnKeyMap: Map<String, String>
- conditionColumnValueMap: Map<String, String>
- cacheColumns: List<String>
- relyOnTables: List<String>
UpdateTableSetting
- unchangeColumns: List<String>
DeleteInsertTableSetting
- deleteFlagMap: Map<String, Object>
- increaseAction: AutoIncreaseAction
InexistenceInsertTableSettin
- compareKeys: List<String>
- increaseAction: AutoIncreaseActio
AutoIncreaseSetting
- increaseColumn: String
- sequenceName: String
Inheritance
Dependency

chart 5-21 Class design for saving table info

1) TableSetting class
This class is used in both exporting and importing.
When exporting the setting data in one table, we need to know the followings.
A) table name
B) primary keys which can be used to export the specific setting records
C) do not take the records as targets if the value of some columns meet the conditions ①
D) column name that will be used by other tables as parimary keys ②
E) refer to the table when export/import ③
Comments
① for example, when delete_flag = 1 which means the record has been logically deleted
② like FILE_ID column in ATTACHFILE_EXT table will be used when exporting ATTACHFILE_EXTENSION table
③ like ATTACHFILE_EXTENSION table depend on the value of FILE_ID column in ATTACHFILE_MST table
To solve the problem mentioned in 4.4, we provide tempTable field in TableSetting class. Because the Real table and TEMP table have the same data structure, the framework can export/import the table and its TEMP table by the same way.
Description of each field in TableSetting class:
tableName: export setting data from which table (SELECT * FROM TABLE_NAME)
tempTable: associated temp table name, can be empty (TEMP_ITEM, TEMP_INPUT, etc.)
columnKeyValue: mapping of column name and key name in Java side (WHERE schem=root and namespace=root.cws.personalInfo...)
conditionColumnValueMap: ignore the record if the value of the given column is same with the one in this object (AND delete_flg=0)

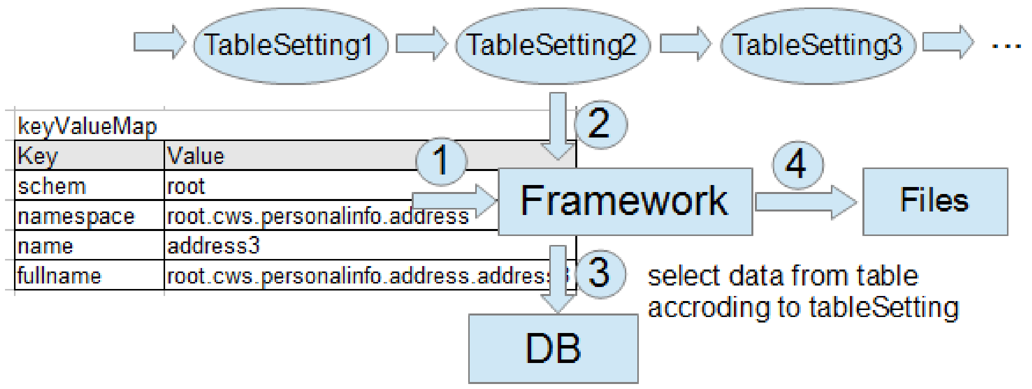
cacheColumns: column names which you want to cache the values of the columns
relyOnTables: table names. Need to use the cached data of the table when exporting/importing current table



```
TableSetting attachfileMst = new TableSettingBuilder()
        .setTableName("ATTACHFILE_MST")
        .addColumnKeyMapping("schem", "schem").addColumnKeyMapping("fullname", "fullname
        .addCacheColumn("FILE_ID").build();

TableSetting attachfileExtension = new TableSettingBuilder()
        .setTableName("ATTACHFILE_EXTENSION")
        .addColumnKeyMapping("FILE_ID", "FILE_ID").addCacheColumn("EXTENSION_ID")
        .addRelayOnTable("ATTACHFILE_MST").build();

TableSetting attachfileExtensionMst = new TableSettingBuilder()
        .setTableName("ATTACHFILE_EXTENSION_MST")
        .addColumnKeyMapping("EXTENSION_ID", "EXTENSION_ID")
        .addRelayOnTable("ATTACHFILE_EXTENSION").build();
```

chart 5-22 example of table setting class

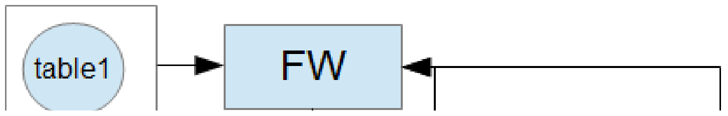
How to use TableSetting class when exporting setting data.



Key	Value
schem	root
namespace	root.cws.personalinfo.address
name	address3
fullname	root.cws.personalinfo.address.addres

chart 5-23 workflow of how FW export setting data

How the framework export setting data.
① Pass the keyValueMap to FW (Java side, what users want to export. example export a service setting)
② Get the table settings for export service setting
③ Retrieve the data according to the configuration in table setting object
④ Export selected data to file
2) UpdateTableSetting class
Use update to import the data from file to DB. As described in 4.5, there are some columns that should not be updated when importing.
Description of each field in UpdateTableSetting class:
unchangeColumns: column names which should not be updated
3) DeleteInsertTableSetting class
Delete the old data first and then insert the new data.
There are two types of delete in current CWS DB: hard delete and logical delete. If the table uses delete flag, developer can add the column name and the value for delete flag in <deleteFlagMap>. When inserting the new data, we may need to get the sequence for a column. In this case, developer can config the increaseAction to achive the goal of creating new sequence for the column.
Description of each filed in DeleteInsertTableSetting class:
deleteFlagMap: delete flag column name and value map
increaseAction: column name and sequence name for auto increase column when inserting
4) InexistenceInsertTableSetting class
Insert the data only when the data does not exist in the table.
Description of each field in InexistenceInsertTableSetting class:
compareKeys: comlumn names which are used to check the data is exist or not
increaseAction: column name and sequence name for auto increase column when inserting
5.2 Hook Point Classes
There are many different kinds of patterns which need to be deal dealt with when migrating data between DB tables. To make our framework more flexible, we provide the Hook Point function to make developers able to write their own Java code which can be called during the export and import.
1) Table Hook Point
When migrating some tables, we need to migrate the associated tables which may not use the given primary key-values. When some tables or sub tables can not use the given primary keys to migrate, developers can customize their own Java class to achive the migration.



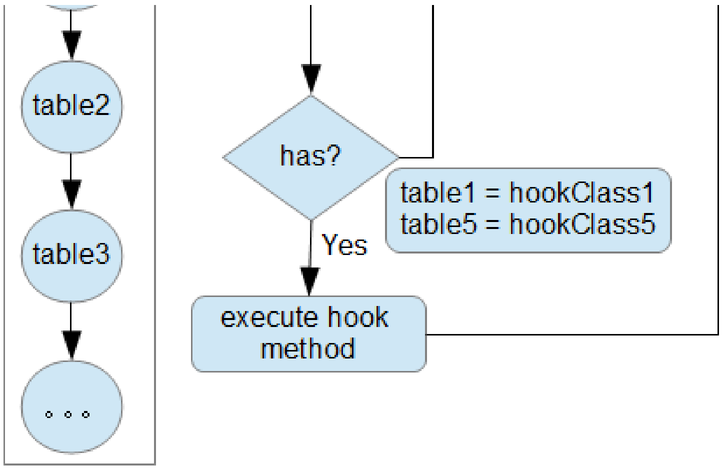


chart 5-24 workflow of execute hook point method

For example, when we export [Attachment Setting], we need to export data in ATTACHFILE_MST table and ATTACHFILE_EXTENSION table and the relationship in ATTACHFILE_EXTENSION_MST table. Therefore, developers need to define logic to migrate data in those tables. Data Migration Framework provides AbstractMigrateHookPoint class which contains exportHookPoint method (will be called during export DB data to file) and importHookPoint method (will be called during import setting data from file to DB) to make developers can customize their own logic for migration very easily.

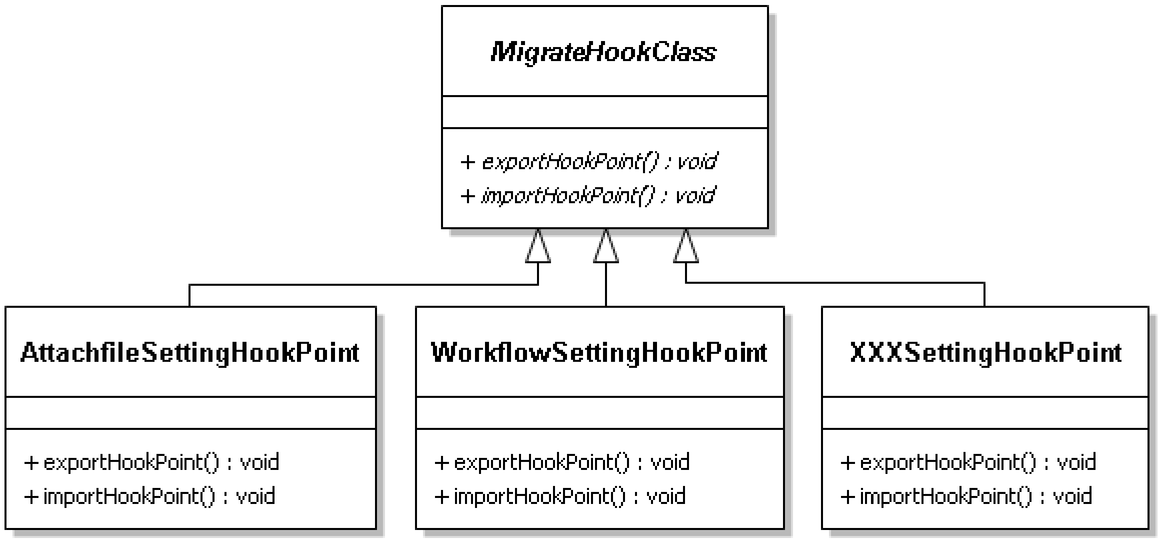

MigrateHookClass
+ exportHookPoint() : void
+ importHookPoint() : void
AttachfileSettingHookPoint
+exportHookPoint() : void
+importHookPoint() : void
WorkflowSettingHookPoint
+exportHookPoint() : void
+importHookPoint() : void
XXXSettingHookPoint
+exportHookPoint() : void
+importHookPoint() : void

chart 5-25 example of implementation of Hook Point

2) Condition Hook Point

In some cases, when we migrate some setting items, we need to migrate the associated setting items at the same time. Likewise the problem mentioned in implementation plan 4.6, when we migrate [commute application setting] we also need to migrate [service setting] and [application route auto calculation setting] at the same time. To solve the problem, we provide the Condition Hook Point function. When the key and value are matched with the key-values in Java side, the method in hook point class will be exexuted.

Key_Value_Mapping	Hook_Class
namespace=root.cws.nintei.tukin	jp.co.worksap.migrate.hook.MigrateCommuteSetting

chart 5-26 example of condition hook pint

6. How to use the framework

6.1 Developing workflow

1) Analyze the DB tables and primary keys of the setting unit
2) Decide the hook point class if necessary
3) Insert the information of 1) and 2) to Java object
4) Implement the page for the setting unit

6.2 Maintenance and developing cost

1) When adding new column of a setting table

If the primary keys of the table which is used in the migration are changed, we need to modify the information about the primary keys in the Java object. If the new items have some special logic that need to be dealt with during the migration, we need to add info in the tables for migration.

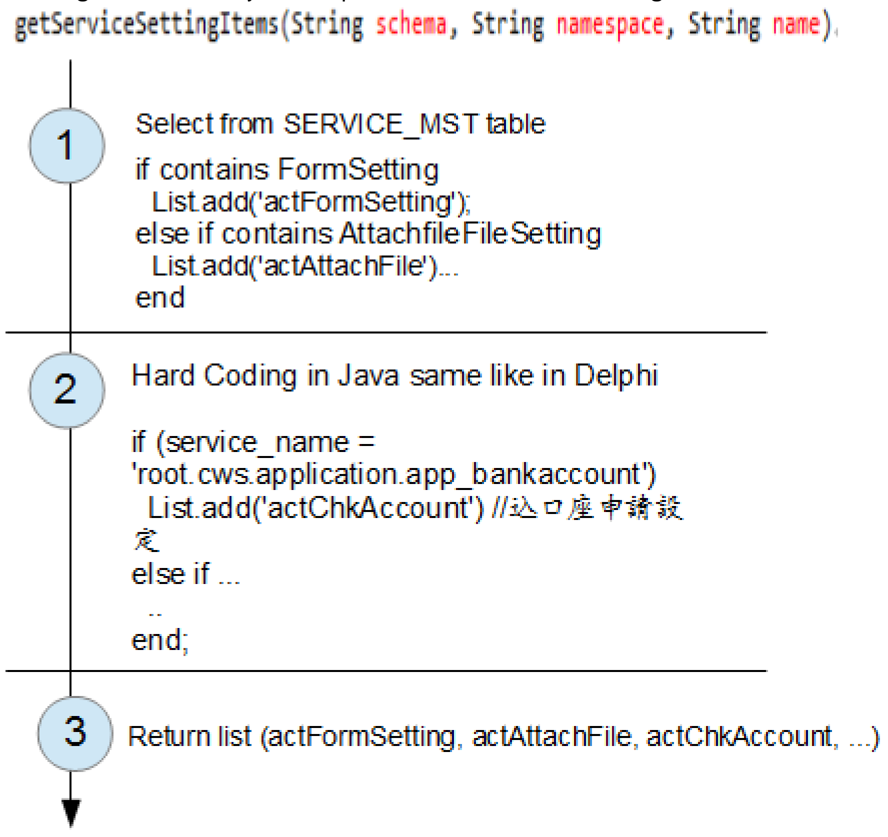
2) Add new table for setting unit

In most cases, we only need to add the info to the tables for migration and add hook point class if necessary. it will not affect on the old migration function.

7. Specification for migrating service setting

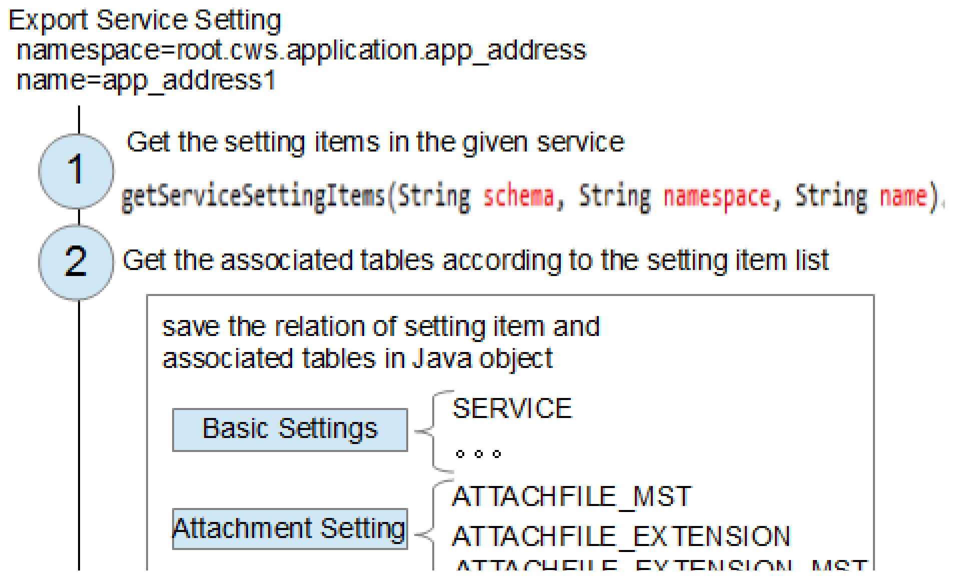
7.1 About multi schema settings

Adminsitrator can config the setting items for each schema. And for some special setting items, they can use the setting of their parents schema (or root schema) or config individual settings.
Policies to deal with the multi schema settings
1) Migrate only the setting items which are enabled in the schema
2) Individual setting items
Form setting and search view setting can set schema individual settings. Administrator can migrate the settings only when he has the authority of editing them.
7.2 About the relationships of setting items and their setting tables
The setting items (e.g., Form Setting, Re-auth Setting, Combox Setting, etc.) for each kind of service may be different. For example, some services do not have [Attachment Setting]. When we migrate the service setting, we need to know the setting items which are contained by the service.
Currently, there are two places that keep the relationship of service and its setting items. For common setting items, the relationships are saved in SERVICE_MST table. However, for some special setting items, the relationships are saved in source code by hard coding. Besides, in this term, when converting service manager to web page, we also need to consider what kinds of setting items does the service contain. So we need to provide a method to get the list of setting items for a given service.
After researching SERVICE_MST table and Delphi client, we decide to provide a common method which can return the list of setting item names by the steps described in the following chart.



```
getServiceSettingItems(String schema, String namespace, String name).

1  Select from SERVICE_MST table
   if contains FormSetting
     List.add('actFormSetting');
   else if contains AttachfileFileSetting
     List.add('actAttachFile')...
   end

2  Hard Coding in Java same like in Delphi

   if (service_name =
   'root.cws.application.app_bankaccount')
     List.add('actChkAccount') //込口座申請設
   定
   else if ...
     ..
   end;

3  Return list (actFormSetting, actAttachFile, actChkAccount, ...)
```

chart 5-27 flow of judging setting items of a given service
When exporting service setting of a given service, we need to
1) call getServiceSettingItems method to get the setting item list of a given service
2) get associated table list
3) export associated data to file



```
Export Service Setting
 namespace=root.cws.application.app_address
 name=app_address1

1  Get the setting items in the given service
   getServiceSettingItems(String schema, String namespace, String name).

2  Get the associated tables according to the setting item list
```

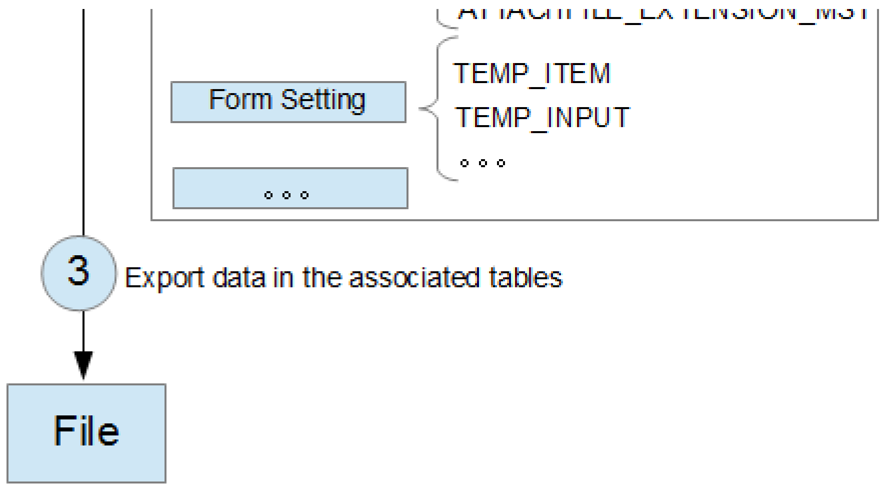


chart 5-28 relation of setting items and their setting tables

7.3 About the reflection

To make sure the service settings are totally the same with the one before migration, we will migrate all the tables that associated with the service setting.

Specification for the importing of service setting in Service Manager

1) Migrate both real table and TEMP table

2) Users need to use cache clear function (Delphi Company Client) to reflect the Dict setting

7.4 Migrate the relationship with another setting item which can not be migrated at the same time

For example: workfolw setting of a service

Currently, we save the relationship of service and workflow by schem, namespace of service and pattern_id of workflow. When migrating the service setting, if we only migrate the workflow pattern_id, we can not find the right workflow by the pattern_id in another DB. Our customers has to config the workflow for the service again after the migration.

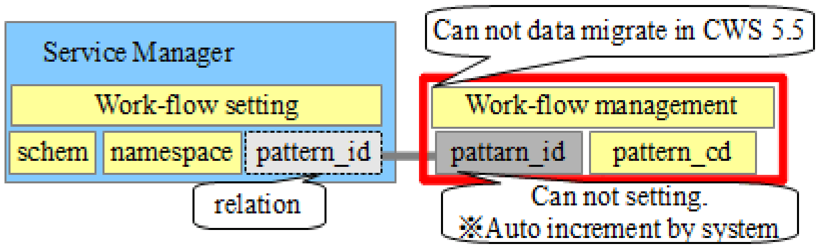


chart 5-29 relation of service and workflow

To solve the problem, we plan to use pattern_cd which is a unique string for workflow when migrating the relationship of service and workflow. Therefore, that if our customers created the same workflow in test and real environment, we can use the pattern_cd to migrate the relationship of service and workflow.

7.5 About output file structure

1) Use Zip to compress export files

2) Output data in each DB tables to CSV file

3) Keep an config file in each export file (including table info, cws version info, etc.)

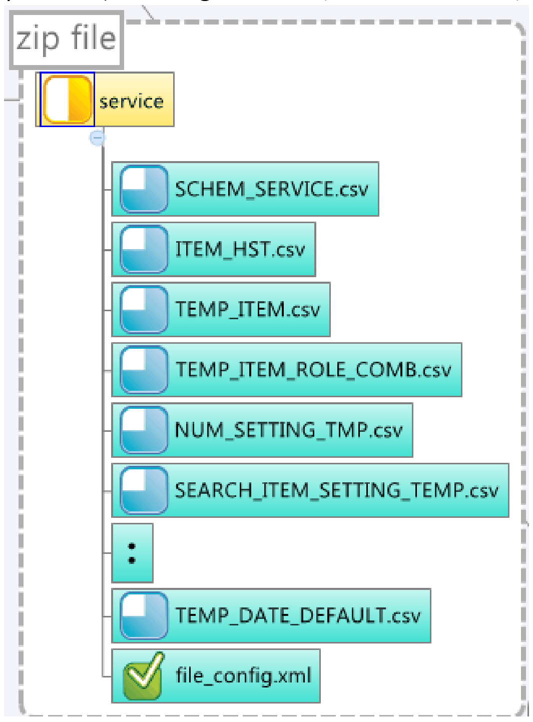


chart 5-30 output file strucutre

7.6 About the error when import

If any of processes failed while importing service, the whole transaction data should be rolled back.

The ideal way of the setting migration is that user can run the service immediately after the migration without any configuration. So the setting data should not be imported if there are some errors when importing or something user need to configure after migration.
For example, if the WORKFLOW_CD is not exist in the real environment, we should provide the information for users and do not import the data into the environment.

8. Page design for service setting data migration

8.1 Add service setting data migration page on Service Manager

Because the data structure of CWSRegistry and data migration framework are different, we can not use the current registry page and logic for export and import data. From the consultant, we know that our customers want to migrate their services most. We decided to add migrate function on Service Manager page. Reasons:
1) Usability, Service Manager is used for configuring service settings, users can confirm service settings and export/import the settings on the same page very easily.
2) Authority, users who has the authority of editing service setting also has the authority of migrate setting data, we can use the same authority control logic for migration.
3) Developing cost, developers do not need to consider about the service tree and authority control because Service Manager has already done those.
In the future, if we need to consider about the data like common condition(共通条件) which is not included by Service Manager, we will create a new catalog for that.

8.2 Data migration can be divided into two flows

1) Operation flow for to export
2) Operation flow for to import

The detail for page design is summarized in another document. Please check the attachment file. [sub refalence.ods]

9. Functions that have already been converted to CWSRegistry

9.1 Boss and subordinate setting (上司部下設定)

We can solve the problem by moving the data of boss and subordinate setting back to original tables and use new framework to implement the migration. This problem we need to be solved in another catalog.

9.2 Other functions

1) Application form external I/O [申請書外部入出力]
2) Cws log setting page [ログ設定(cwslog)]
3) System security setting page [システムセキュリティ設定]
4) External service [外部サービス]
5) Subsidiary Permisson Commute-App [認定決議書:通勤届] part settings
6) Service link setting page [サービスリンク設定]

Currently, there is no problem in using those functions. Therefore, we recommend to keep them.

9.3 Milestone of closing CWSRegistry page

In the future, we will provide the whole setting migration page. After that we move the settings in current CWSRegistry page to whole setting migration page and close the CWSregistry page. We will keep CWSRegistry page in CWS5.5.

10. Function constraints

Master data is imported by DataLink from CJK.
1) Need to synchronize master data between environments before migrating setting data
For example, in order to transport the common criteria contents of [Department of the given employee equals AAA] from test environment to real environment, customers need to make sure that there is same department AAA (and same hierarchical relation) in both test environment and real one. Also the same employees are need. If any of the conditions is different, customers could not migrate the setting perfectly.

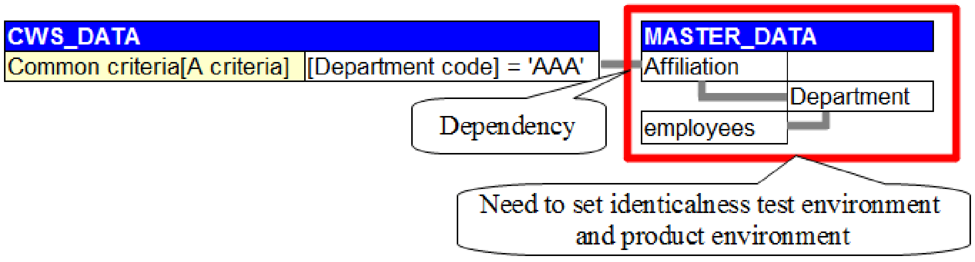

CWS_DATA
Common criteria[A criteria]
[Department code] = 'AAA'
Dependency
MASTER_DATA
Affiliation
Department
employees
Need to set identicalness test environment and product environment

chart 5-31 example of common criteria

DB copy tools can copy tables between environments. It is abled to be used in synchronizing master data.
2) When can not import setting files
Users can not import the setting files exported from a different version in which the setting table structure is changed. We will write the CWS version in config files when export the settings and check the version before importing.

Appendix

1. Data Migration Framework Guideline
This guideline is for developers when they consider migrating settings which are used by administrators.
Application data and user individual settings are not the targets of this guideline.
1.1 For new setting functions
A) When developers need to create new tables
There are three merits for using CWSRegistry:
・No SQL (do not need to write DAO)
・No need to create tables (CWSRegistry saves all data in CWS_REGISTRY table)
・No need to write code for data migration (CWSRegistry provide function for migration)
However, for CWSRegistry saves all data into one DB table, it may cause performance problem if there are too much data in registry table.
To avoid performance problem, developers should not use data which expected increase too much.
When using CWSRegistry, developers should consider about the expected data amount and get reviewed if it is appreciated to use CWSRegistry for the setting.
For example, Boss and subordinate setting (上司部下設定) is not appreicated to use CWSRegistry.
The setting is related with each employee. The amount of employees may be very large. In addition, we do not have any restrictions on the data size which means users can make endless setting data. Therefore, the boss and subordinate data will affect the whole functions using CWSRegistry.
When using CWSRegistry, developers can use the function provided by CWSRegistry to implement the data migration fucntion.
B) only add new columns to existed tables
Developers can use the new framework for data migration.

1.2 For existed functions
Developers can use new data migration framework for migration and there is no need for them to conver current funtions to CWSRegistry.
1.3 About the page for data migration
Developers need to consider the data migration page when they add new settings. Currently, there is no page for whole data migration, so it is OK to use current CWSRegistry page for migrating the settings implemented by CWSRegistry.
Developers also need to judge if it is necessary to put the migration function on the setting page.
2. Templates for CWS
2.1 What is CWS template
CWS template is a service setting file which can be imported into our customers' system to reduce setting cost when customer starts to use a new service. Also, we can push our customers to use new business process by providing templates.
2.2 For what kind of services we need to provide templates
1) For some business services used by customers with similar settings
Currently, our customers use Personal Information Application in semilar settings, so we provided template function to reduce our customers' initialize service setting cost. There are other services used by customers with semilar settings, e.g., Bussinehes Trip Application (出張申請), Commutation Expense Service(通勤届), etc.
However those services do not have template functions.
2) For some business services which have many settings
For example, Performance Review(人事考課), Business Intelligence have many settings for supporting our customers' business process. The service can be used in many kinds of business processes. If we can release template for each kind of business process, it would help our customer in understanding how to use this service and how to create better business processes.
2.3. How to release template (plan)
1) How customers can get the template
・Release with release CD
・Download from @SUPPORT
We need to manage the recommended settings for different CWS version
2) How we maintain the templates
Put the templates into SVNRepository, so that developers can manage the version and history of the templates
3) About developing cycle of the templates

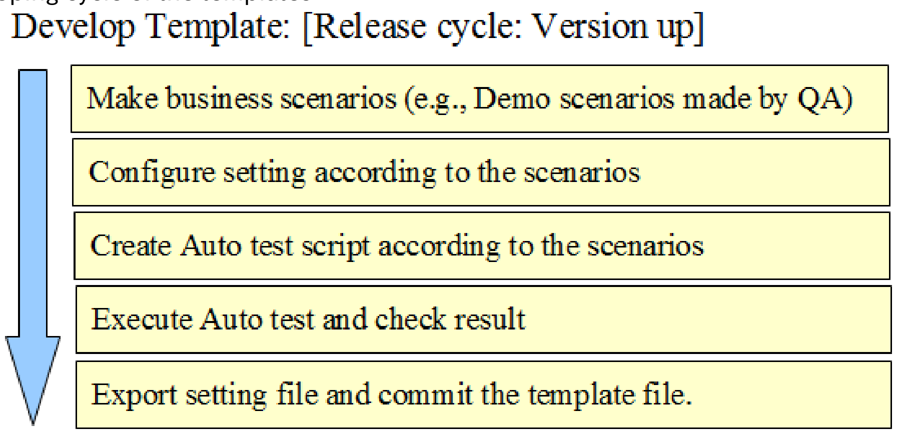

Develop Template: [Release cycle: Version up]
Make business scenarios (e.g., Demo scenarios made by QA)
Configure setting according to the scenarios
Create Auto test script according to the scenarios
Execute Auto test and check result
Export setting file and commit the template file.

chart 6-1 Develop template flow

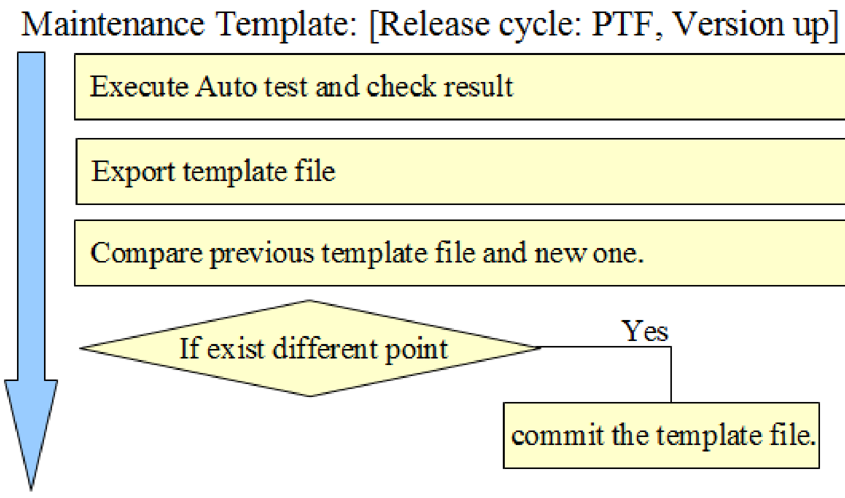

Maintenance Template: [Release cycle: PTF, Version up]
Execute Auto test and check result
Export template file
Compare previous template file and new one.
If exist different point
Yes
commit the template file.

chart 6-2 Mantenance template flow

Function List

No.	1	Milestone	CWS55VerUp,	Man Day	0.0	Priority	high
Title	Export/Import setting data function for setting page						

Brief summary:
Users can export service setting data to file and import the setting data form file to the system.
Data migration unit is each setting unit in setting page (service manager, workflow, common criteria).
Merits:
Reduce the setting cost.
1) By using service setting template, reduce the setting cost for installation
2) Users do not need to config the setting on real environment again after testing on test environment

No.	2	Milestone	CWS55VerUp,	Man Day	0.0	Priority	high
Title	Backup setting data before importing setting files						

Brief summary:
Ask users to backup the setting data which will be overwritten by the import file. When error happens during the import progress, users can recover the settings by the backup files.
Merits:
Users can recover the system when error happens.

No.	3	Milestone	CWS55VerUp,	Man Day	0.0	Priority	middle
Title	Multi units data migration function						

Brief summary:
Be able to migrate multi units in one migrate operation, such as users can choose several associated services and migrate at the same time.
Merits:
Reduce users' operation cost.

No.	4	Milestone	CWS55VerUp,	Man Day	0.0	Priority	high
Title	Confirm import data information before importing.						

Brief summary:
Before importing, be able to confirm the information of import setting file. For example, exported environment name, export time and file comment. The file comment can be written by exporting operator.
Merits:
Users can confirm the file is right or not before doing the import operation.

No.	5	Milestone	CWS55VerUp,	Man Day	0.0	Priority	middle
Title	Show import result						

Brief summary:
After importing, show import result.
Show the reasons when there are failed settings.
Merits:
Tell users what need to do after importing to improve the usability.

No.	6	Milestone	CWS55VerUp,	Man Day	0.0	Priority	middle
Title	Import progress log						

Brief summary:
Be able to get import process log.
When our customers happen problems in import, they can inquiry to us with the process log file.
Merits:
Help developers and constant to analyze customers' problem
Reduce customers' inquiry cost.

No.	7	Milestone	時期未定(案件),	Man Day	0.0	Priority	low
Title	Export/Import associated setting data						

Brief summary:
Be able to migrate the associated setting data at the same time. (Migrate the workflow, common criteria data when migrating the service setting)
Users can choose to add new setting for workflow and common criteria data or merge them with existed workflow or common criteria.
Merits:
Reduse data migration steps.

No.	8	Milestone		Man Day	0.0	Priority	middle
Title	log function (setting data changed log)						

Brief summary:
Log function to check when, who, and which settings have been change by importing.

This will be implemented as Service Manager's log function in the futrue.

Merits:
The Users will be able to make actions when unwanted changes happen to their settings and thereby unwanted events such as wrong workflows happen in their operations.