Time slices of contractual relationships

contract, slice

On many projects, we encounter a source of data about users, employees or org. structures that work with so-called time slices. For better works with this time slots, agenda of contract's time slices was created.

The basic idea is that time slices are stored in a self-contained agenda. This agenda only contains time slices for identity contracts. If a given slice is currently valid, its values will be **copied into the linked identity contract**. **Every day** a scheduled task is performed, which calculates whether another slice is valid. Such a slice becomes currently used as a contract (its values are copied into the contract).

In one day can exists only **one slice** for one contract. Every slice must contains all data of the contract. Slice is **snapshot** of the contract!

Task for update contract by current slice

For recalculate current using slices as contract task `SelectCurrentContractSliceTaskExecutor` was created. This task ensures: Find all slices which should be for actual date using as contract and copy their values to parent contracts.

When will be found more then once slice (for same contract) which should be set as current, then the contract will be not recalculated! **Error will be logged in this LRT**.

By default is that task triggered every day at **0:30 AM**.

Slices recalculation

During synchronization is for each processed item set dirty state. Newly created event by synchronization contains flag **SET_DIRTY_STATE_CONTRACT_SLICE** = true. This flag is catches by processor **ContractSliceSaveRecalculateProcessor**. In processor is for these slices set entity state with dirty flag.

After synchronization correctly ended is started HR processes if is these option enabled. Together with HR processes is also started task **ClearDirtyStateForContractSliceTaskExecutor**. These task process all slices that has dirty state and recalculated their status.

When you disable HR process is required created the task **ClearDirtyStateForContractSliceTaskExecutor** as dependent on synchronization. Otherwise slices will not be recalculated! LRT ClearDirtyStateForContractSliceTaskExecutor is very important part of slice sync, this LRT ensures create/delete of contracts and delete slices (marked during sync to as to delete). This is reason, why after every sync of slice, **must be this LRT executed**.

Protection of the contract validity

Sometimes there may be a situation where one of the time slices **ends** the contract, and at the same time there is a next time slice that **restarts** this contract. If there is no gap between termination and restart, then the contract will not terminate (no accounts will be deleted). If the dates do not follow, then (by default) will be **contract terminated** and all connected **accounts will be removed** from the target systems.

However, in some situations (projects), it is required to use the **protection period** for which the contract will **not be terminated**, provided that there is a next slice in the contract, which restarts the contract. Furthermore must be ensured that the gap between the termination and the beginning of the contract is less than or equal to the protection interval.

The protected interval can be set using the property `idm.sec.core.contract-slice.protection-interval`, where the value is the **number of days**. If the number of days between the termination of the contract and its renewal in the following time slice is **less than or equal** to the number of days set in the protection interval, then the date of the contract validity **from** the following slice will be used instead of the date of **termination** of the contract from the currently valid slice.

By default, the property `idm.sec.core.contract-slice.protection-interval` is set to value '**0**'. That means protection interval is **disabled**.

If you **change** the property `idm.sec.core.contract-slice.protection-interval`, it doesn't immediately affect already existing contracts and slices. You must ensure that **all** affected contracts will be recalculated, so the new value of the protection interval is used. The easiest way: run the reconcilation of all contract slices.

How to customize the method for creating/updating contract by slice?

Method `updateContractBySlice` in the `ContractSliceManager` ensures creating or updating a contract by slice (marked with '**Is using as a contract**'). This method supports protection mode for contract validity (described above).

If you need to implement **more specific behavior**, you can **override** that method. For this you can create new service whitch will extend the default implementation of the contract slice manager `DefaultContractSliceManager`.

This new service must have an annotation `@Primary` (ensures that new service will have a priority).

```
/**
 * Your manager for contract slices
 *
 * @author svandav
 *
 */
@Primary
@Service("yourContractSliceManager")
public class YourContractSliceManager extends DefaultContractSliceManager
implements ContractSliceManager {

    @Override
    @Transactional
    public IdmIdentityContractDto
updateContractBySlice(IdmIdentityContractDto contract, IdmContractSliceDto
slice,
            Map<String, Serializable> eventProperties) {

        /**
         * Your code ...
         */
        return super.updateContractBySlice(contract, slice,
eventProperties);
    }
}
```

Sync

Synchronization of time slices is very similar as sync of contracts.

Sync of slices adds new attributes:

- **Contract code** - Code of the parent contract. This `String` value represents relation between all slices for the same contract. It means all slices for one contract must have this value same. During slice recalculation is created final relation on the contract (`parentContract`) in dependency on this contract code.
- **Valid from of slice** - Defines time from that is slice valid. Valid till of slice is computes automatically (by validity of next slice) after save.

Sync of contract slice using specific configuration from sync of contract. **Valid till**

date is set only if slice is last (in sync)!

More informations about sync of time slices are here.

Processors

Core module

- **ContractSliceSaveProcessor**
  - Basic processor for create and update entity `IdmContractSlice` in the database.
- **ContractSliceSaveRecalculateProcessor**
  - Processor executes after `ContractSliceSaveProcessor`. Ensures order of slices (generates `validTill` on the slice) and creates or updates the parent contract (copy values from current slice to the contract).
- **ContractSliceDeleteProcessor**
  - Basic processor for delete of the entity `IdmContractSlice`. Ensure referential integrity (deletes connected `IdmContractSliceGuarantee`). If is slice last (for same parent contract), then is parent contract deleted too.
- **ContractSliceGuaranteeSaveProcessor**
  - Basic processor for create and update entity `IdmContractSliceGuarantee` in the database.
- **ContractSliceGuaranteeDeleteProcessor**
  - Basic processor for delete of the entity `IdmContractSliceGuarantee`.

Acc module

- **ContractSliceDeleteProcessor**
  - Ensures referential integrity - Before delete of the slice, deletes all contract-slice-account relations.
- **ContractSliceAccountSaveProcessor**
  - Basic processor for create and update entity `IdmContractSliceAccount` in the database.
- **ContractSliceAccountDeleteProcessor**
  - Basic processor for delete of the entity `IdmContractSliceAccount`. Ensure referential integrity. If is relation last (for same account), then is account (`AccAccount`) deleted too.

Important properties

Slices are stored in the separate entity `IdmContractSlice`. Basic structure of the slices are same as identity contract.

Relations between slices are realized over parent contract. It means every slice has relation on the contract. If none valid slice exists, then is created unvalid contract.

For expired slice will existed contract (set as expired).

One from the slices (under one contract) is **marked as actual** (`IdmContractSlice.isUsingAsContract()`).

EAV definition for IdmIdentityContract is reusing in slices. EAV values are persisted in separete table (`IdmContractSliceFormValue`).

If is slice last, then his valid till field are sets to **null**.

Contract cannot be modified when has some slices (is controlled by slices). **Check is only on FE**. Contract cannot be removed when has some slices (is controlled by slices). **Check is only on BE**. If the **last slice** of the contract is deleted, then contract will deleted too.