Blossom Private Data Collections

The Blossom smart contracts use Private Data Collections (PDC) to satisfy privacy requirements and AWS managed blockchain deployment requirements. Below is a small explanation of all the PDCs being used in the smart contracts and how NGAC is incorporated. Note, the use of PDCs and NGAC does not affect the process for a new organization to join the network. It's only once they've joined the network and channel, can they interact with the chaincode.

Catalog PDC

The Catalog PDC stores basic information regarding available software assets. The blossom super user (super:BlossomMSP) is the only one with permissions to Onboard new assets. They can delegate this control to other members of the BlossomMSP (not yet implemented). This is done using NGAC in the smart contract function **OnboardAsset**. All other users in the network only have read access to the PDC via NGAC in the various smart contract functions.

Licenses PDC

The Licenses PDC is only used by the blossom organization (BlossomMSP). No other organization has read or write access. This access is controlled by the collection definition (not NGAC). The purpose of this collection is to store the licenses for assets in a secure way, but also make it simple to share the licenses once an organization check them out. The checkout process is two steps:

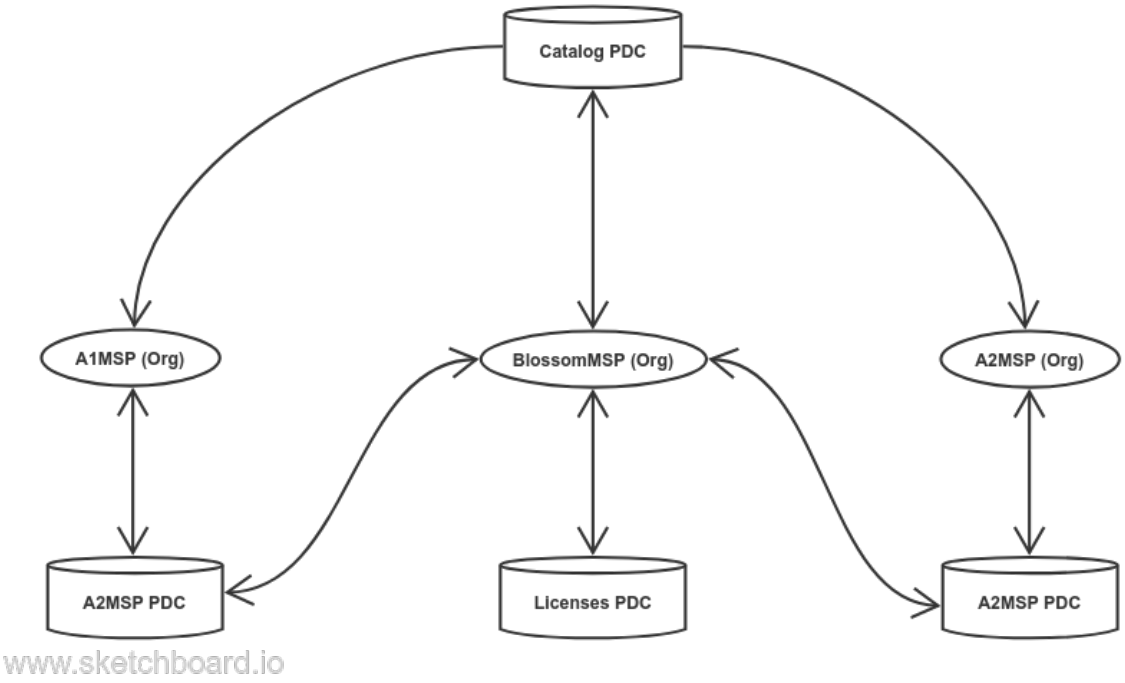
1. An organizations System Admin will call the **RequestCheckout** function to request licenses for an asset.
2. The blossom admin will call **ApproveCheckout** to share the licenses and store them in the organization's own PDC (more info below).

This process must be two steps because the requesting organization does not have access to the licenses PDC and won't be able to retrieve them. The blossom admin, having access to both the licenses PDC and the organizations PDC, allows them to retrieve the checked-out licenses and store them in the organization PDC as a result of calling ApproveCheckout.

Organization PDC

Access to PDCs is defined at the Member level in the collections config file. As such, each member will have its own PDC that stores the licenses they have checked out. Another consequence of this is there can only be one Blossom "Account" for each Member in the network. Each account is identified by the MSPID of the requesting user, since this is the same MSPID used to define access to the PDC. The BlossomMSP is the only other organization with read or write access to an Organization PDC in order to provide licenses.

Example



This is an example configuration with two organizations: A1 and A2. A1MSP and A2MSP have read/write access to their own PDC and read access to the catalog PDC. BlossomMSP has read/write access to both A1MSP and A2MSP PDCs. They also have read/write access to the catalog and licenses PDCs to manage available assets.

Adding Organizations

When an organization is added to the network, they need to also be added to the "acquisition" channel where the smart contracts are installed. They will then need to invoke the **RequestAccount** function, specifying the organization's System Owner, System Admin, and Acquisition Specialist to be used by NGAC and the organization's ATO. The account will be labeled as "Pending", until the blossom admin updates the status to "Active" using the **UpdateAccountStatus** function. Once the account is active the blossom admin must add the organization to the PDC config file and update the smart contract to reflect the new config. At this point, the new organization will be able to interact with the smart contract functions normally. Note, this workflow is only to allow an organization to call the smart contract functions. The process to join the network and channel is the same.

Collection_config.json

Below is the collection config file used to create the configuration in the above image.

```
[
  {
    "name":"catalog_coll",
    "policy":"AND('BlossomMSP.member', OR('A1MSP.member', 'A2MSP.member'))",
    "requiredPeerCount":1,
    "maxPeerCount":3,
    "blockToLive":1000000,
    "memberOnlyRead":true,
    "memberOnlyWrite":true
  },
  {
    "name":"licenses_coll",
    "policy":"OR('BlossomMSP.member')",
    "requiredPeerCount":1,
    "maxPeerCount":1,
    "blockToLive":1000000,
    "memberOnlyRead":true,
    "memberOnlyWrite":true
  },
  {
    "name":"A1MSP_account_coll",
    "policy":"OR('BlossomMSP.member', 'A1MSP.member')",
    "requiredPeerCount":1,
    "maxPeerCount":2,
    "blockToLive":1000000,
    "memberOnlyRead":true,
    "memberOnlyWrite":true
  },
  {
    "name":"A2MSP_account_coll",
    "policy":"OR('BlossomMSP.member', 'A2MSP.member')",
    "requiredPeerCount":1,
    "maxPeerCount":2,
    "blockToLive":1000000,
    "memberOnlyRead":true,
    "memberOnlyWrite":true
  }
]
```

Considerations

If an organization is updated to a status other than “Active”, the chaincode definition will need to be updated to remove them from the Catalog PDC so they can’t view available assets. They can also be removed from their own PDC. However, NGAC will ensure they can’t perform any actions and there is no information to read in their own PDC besides already checked out licenses. This will likely need to happen at the application level because the Fabric does not have a native way of doing this automatically in response to chaincode execution.

NGAC

NGAC is used in several components of the configuration to control access to smart contract functions. There are instances where a user is allowed to access a PDC but may not be able to perform a certain action given certain conditions. There are two policy definitions. The first policy is for the Catalog PDC which allows only the super user to Onboard and Offboard assets. This policy does also provide a way for the super user to delegate these permissions to other users as they see fit. The second policy is the Account policy. Each account will have the same underlying policy stored in their PDC. The policy is tailored for the specific account's users. Below is an updated diagram depicting the NGAC policy instances.