
NIAGARA
SUMMIT
2026
SEAMLESS CONNECTIVITY,
POWERFUL INTELLIGENCE
TRIDIUM

This document is a non-binding, confidential document that contains valuable proprietary and confidential information of Tridium and must not be disclosed to any third party without our written agreement. It does not create any binding obligation for us to develop or sell any product, service, or offering. Content provided here may not be altered or modified and must remain in the format originally presented by Tridium. Any descriptions of future product direction, intended updates, or new or improved features or functions are for informational purposes only and are not binding commitments by Tridium. The sale, development, release, and timing of any such products, updates, features, or functions remain in Tridium's sole discretion.

TRIDIUM

RESTRICTING ACCESS TO FROZEN SLOTS

MELANIE COGGAN

SOFTWARE ENGINEER, SECURITY ADVOCATE

NIAGARA SUMMIT 2026 DEVELOPER

TRIDIUM

Security Coding Concept: Access

- Who can run your code
- What they can do with it
- Why this matters for Niagara

Access in Niagara

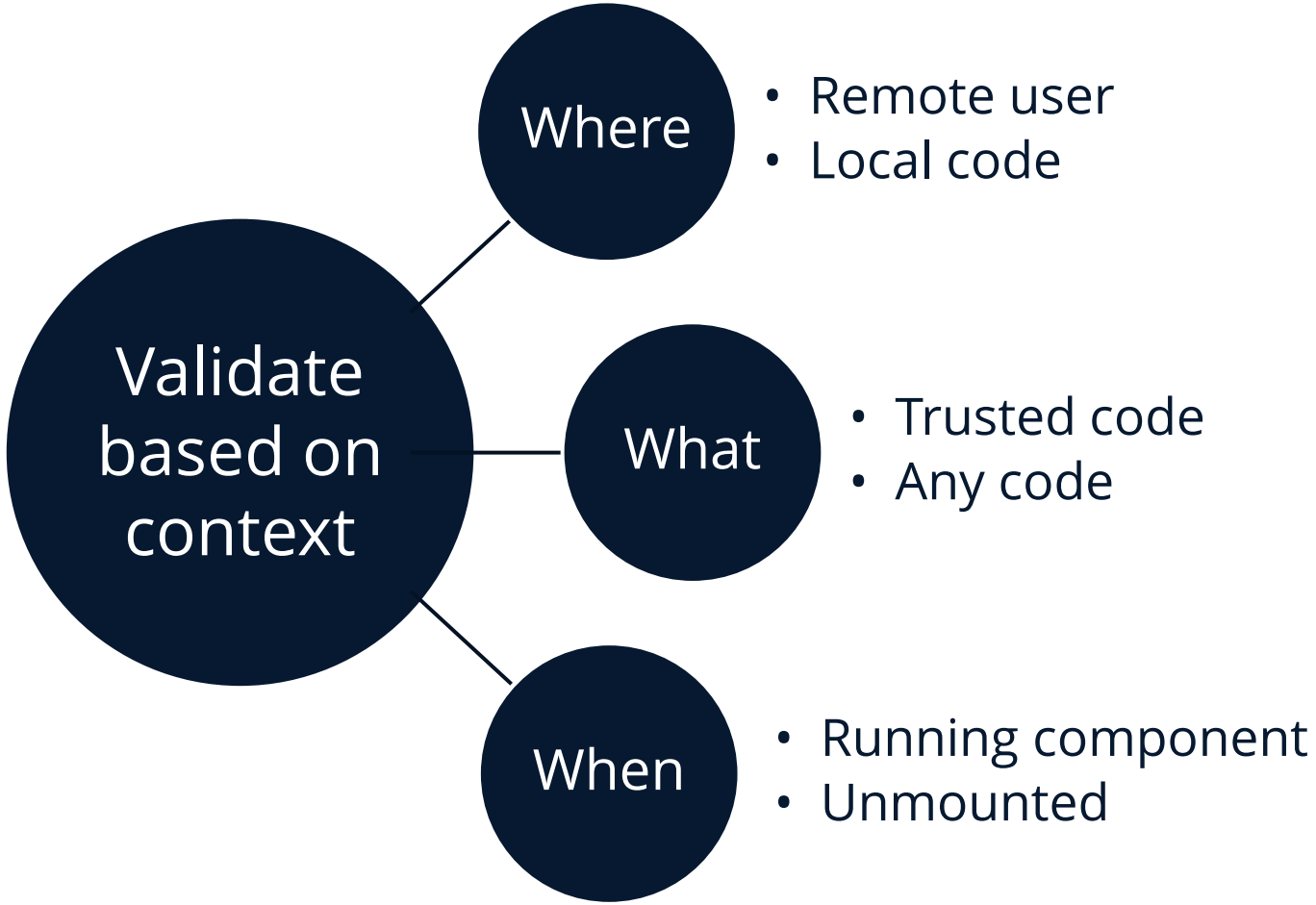
Niagara is a framework

Slots are public

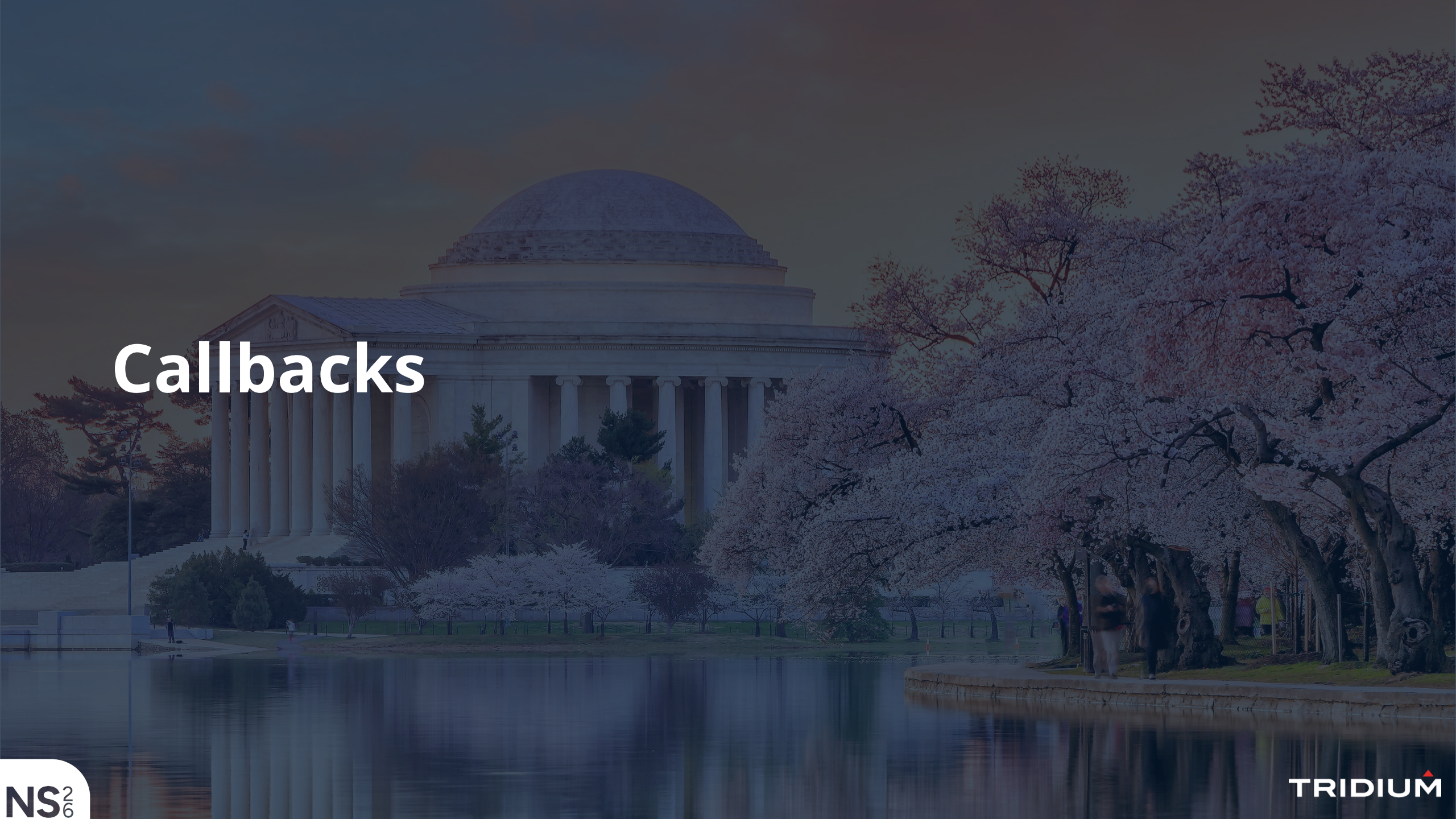
Desired Slot Behavior

- Read-only
- Allows links
- Invocation permissions
- Critical to functionality

BIFrozenSlotContextValidator



Callbacks



BIFrozenSlotContextValidator

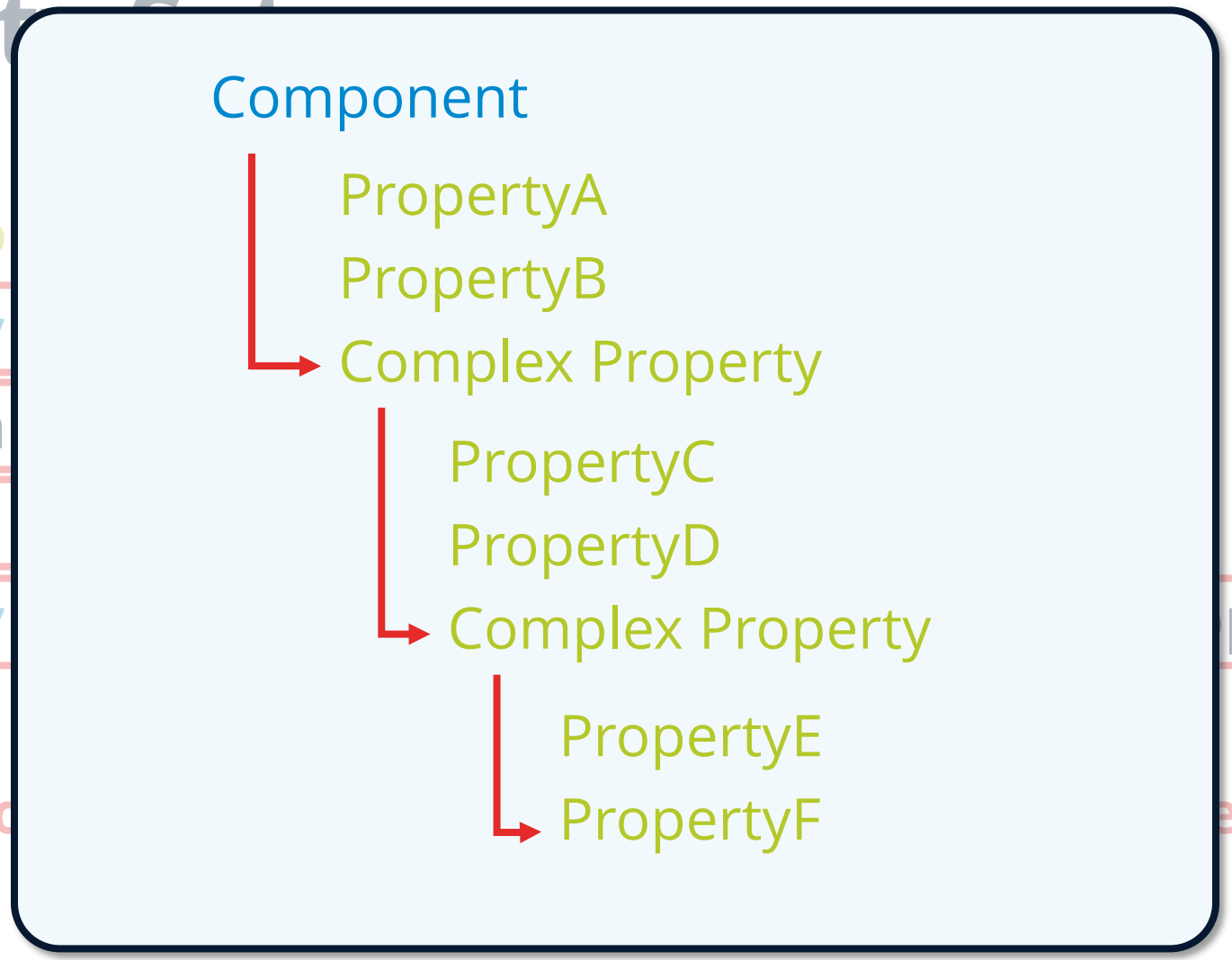
```
default void validateContextForPropertySet(…)

default void validateContextForActionInvocation(…)

default void validateContextForTopicFire(…)
```

```
default Set<String> getTrustedModulesForSlotOperations(Slot slot)

default boolean trustSubclassesForSlotOperations(Slot slot)
```


Component
PropertyA
PropertyB
Complex Property
PropertyC
PropertyD
Complex Property
PropertyE
PropertyF

Topic Fires

```
default void validateContextForTopicFire(
    Topic topic,
    BValue eventArg,
    Context context,
    Property... pathToDescendantContainingTopic)
{
}
```

Action Invokes

```
//region /*+ --------- BEGIN BAJA AUTO GENERATED CODE --------- +*/
…
@Generated
public void myAction() { invoke(myAction, null, null); }
…
//endregion /*+ --------- END BAJA AUTO GENERATED CODE ----------- +*/



…


public void doMyAction()
{
  validateContextForActionInvocation(…)
  // Do some work
}
```

Convenience Methods

NS26

TRIDIUM

Running Station

```
static void checkActiveStationOperation(
  BComplex complex,
  Slot slot,
  Context context,
  Property... pathToDescendantContainingSlot)
{
}
```

Only allows the operation if the complex is mounted in a running station

Running Station - Example

```
public void updateStatus()
{
   if (!checkLicense())
   {
      setStatus(BStatus.FAULT);
   }
}
```

Protect the ‘status’ property with `checkActiveStationOperation()`

Remote User Operations

```
static void checkRunningAndRemoteUserOperation(...)
```

Mounted in a running station
Initiated by a remote user

```
static void checkNotRunningOrRemoteUserOperation(...)
```

Not mounted in a running station

Remote User Operations - Example

```
public final void doUpdatePassword(
    char[] oldPassword,
    char[] newPassword,
    Context context
)
{
    // Do some validation
    setPassword(BPassword.make(newPassword));
}
```

Protect the 'updatePassword' action with `checkRunningAndRemoteUserOperation()`

Local Code Operations

```
static void checkRunningAndNonRemoteUserOperation(...)
```

Mounted in a running station
NOT initiated by a remote user

```
static void checkNotRunningOrNonRemoteUserOperation(...)
```

Not mounted in a running station

Local Code Operations - Example

```
public void loginFailed()
{
   auditLoginFailure()
   if (numFailures >= maxNumFailures)
      setLockout(true);
}
```

Protect the 'lockout' property with `checkNotRunningOrNonRemoteUserOperation()`

Module-Scoped Operations

```
static void checkSlotOperationCalledByOwnerOrFramework()
```

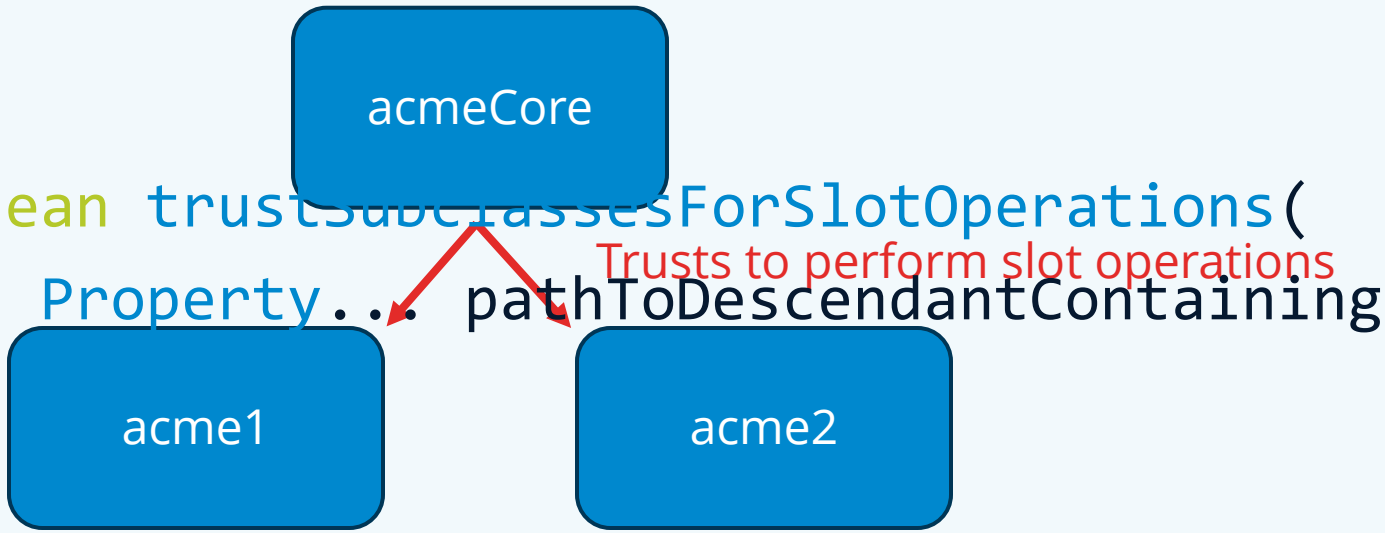
Initiated by the module that owns the slot

Initiated by other specified trusted modules

Module Scoped Operations

```
public interface BIFrozenSlotContextValidator
{
  default Set<String> getTrustedModulesForSlotOperations(
    Slot slot, Property... pathToDescendantContainingSlot
  )

  default boolean trustSubclassesForSlotOperations(
    Slot slot, Property... pathToDescendantContainingSlot
  )
}
```



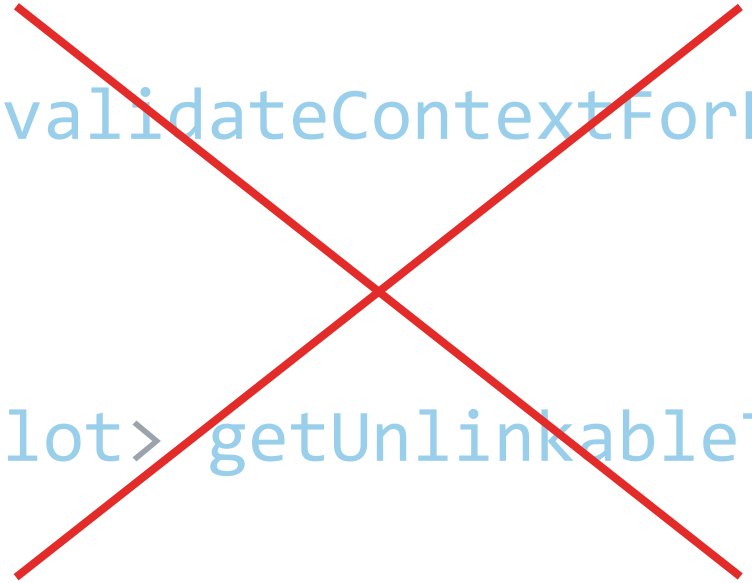
Read-Only Property Implementation

BIReadonlyProperty Container

- True read-only
- Not editable by users
- Not editable by code

BIReadonlyPropertyContainer

```
default Set<Property> getPropertiesToProtect()
```



```
default void validateContextForPropertySet(…)
```

No need to override unless you're making it final

```
default Set<Slot> getUnlinkableTargetSlots(…)
```

BIReadonlyPropertyContainer

```
Set<Property> getPropertiesToProtect()
```

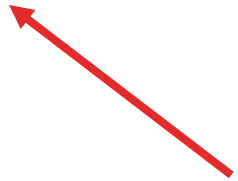
Defaults to properties with READONLY flag

Can be overridden

Make it final!

Preventing Permission Exceptions

```
public void loginFailed()
{
   auditLoginFailure()
   if (numFailures >= maxNumFailures)
      setLockout(true);
}
```



Will trigger a permission check due to `checkSlotOperationCalledByOwnerOrFramework()`

Preventing Permission Exceptions

```
public void loginFailed()
{
  auditLoginFailure()
  if (numFailures >= maxNumFailures)
  {
    SecurityUtil.doPrivileged((PrivilegedAction<Void>)() -> {
      setLockOut(true);
      return null;
    });
  }
}
```

Preventing Permission Exceptions

```
setLockout(true)
```

```
set(lockout, true)
```

```
set(“lockout”, true)
```

Thank you!