Question
Eduard Lebedyuk · Jan 11, 2023

Inbound Adapter safe shutdown strategy

In Interoperability productions Inbound Adapters extract and separate retrieval logic from actual payload processing, which is left to a BS.

At a high level, an adapter looks like this:

```
Class MyAdapter Extends Ens.InboundAdapter {

Method OnTask() As %Status
{
  Set request = ..RetrieveRequest()
  Set sc = ..BusinessHost.ProcessInput(request)
  Set ..BusinessHost.%WaitForNextCallInterval=1
  Quit sc

}
}
```

However, in many cases, *RetrieveRequest* retrieves a batch payload, so our adapter looks like this instead:

```
Class MyAdapter Extends Ens.InboundAdapter {


Method OnTask() As %Status
{
  Set requests = ..RetrieveRequests()
  For i=1:1:requests.Count() {
    Set sc = ..BusinessHost.ProcessInput(requests.GetAt(i))
  }
  Set ..BusinessHost.%WaitForNextCallInterval=1
  Quit sc
}

}
```

Batch processing is popular due to a variety of reasons:

- It's usually faster
- On the external system side, it allows easier scaling
- That's the only way the external system can work

There is, however, a problem if a batch is too large or individual processing time is too long - the entire time *OnTask* runs, Adapter and the corresponding Business Service can't be shut down. The *EnableConfigItem*/*TempStopConfigItem* calls would fail unless forced, which is also harmful as it can affect forced BS and other BHs scheduled to stop.

My question: is there a cheap call I can make from inside the loop to check if there's an external BH disable request?

Of course, already retrieved but unprocessed requests must be compensated (a user needs to roll unprocessed batch items back if that's applicable at all), but assuming that is doable, how can I check for an external disable event?

My idea is something like this:

```
Class MyAdapter Extends Ens.InboundAdapter {


Method OnTask() As %Status
{
  Set requests = ..RetrieveRequests()
  For i=1:1:requests.Count() {
    If ..GotExternalDisableEvent() {
      Set ..BusinessHost.%WaitForNextCallInterval=1
      Return ..CompensateUnprocessedRequests(requests, i)
    }
    Set sc = ..BusinessHost.ProcessInput(requests.GetAt(i))
  }
  Set ..BusinessHost.%WaitForNextCallInterval=1
  Quit sc
}

}
```

Calling @James MacKeith and @Dmitry Zasypkin and @sween .

#Business Service #Interoperability #InterSystems IRIS
Product version: IRIS 2022.1
$ZV: IRIS for Windows (x86-64) 2022.1 (Build 209U) Tue May 31 2022 12:16:40 EDT