Article
Suriya Narayana... · Nov 12, 2016 5m read

Ensemble Orphaned Messages

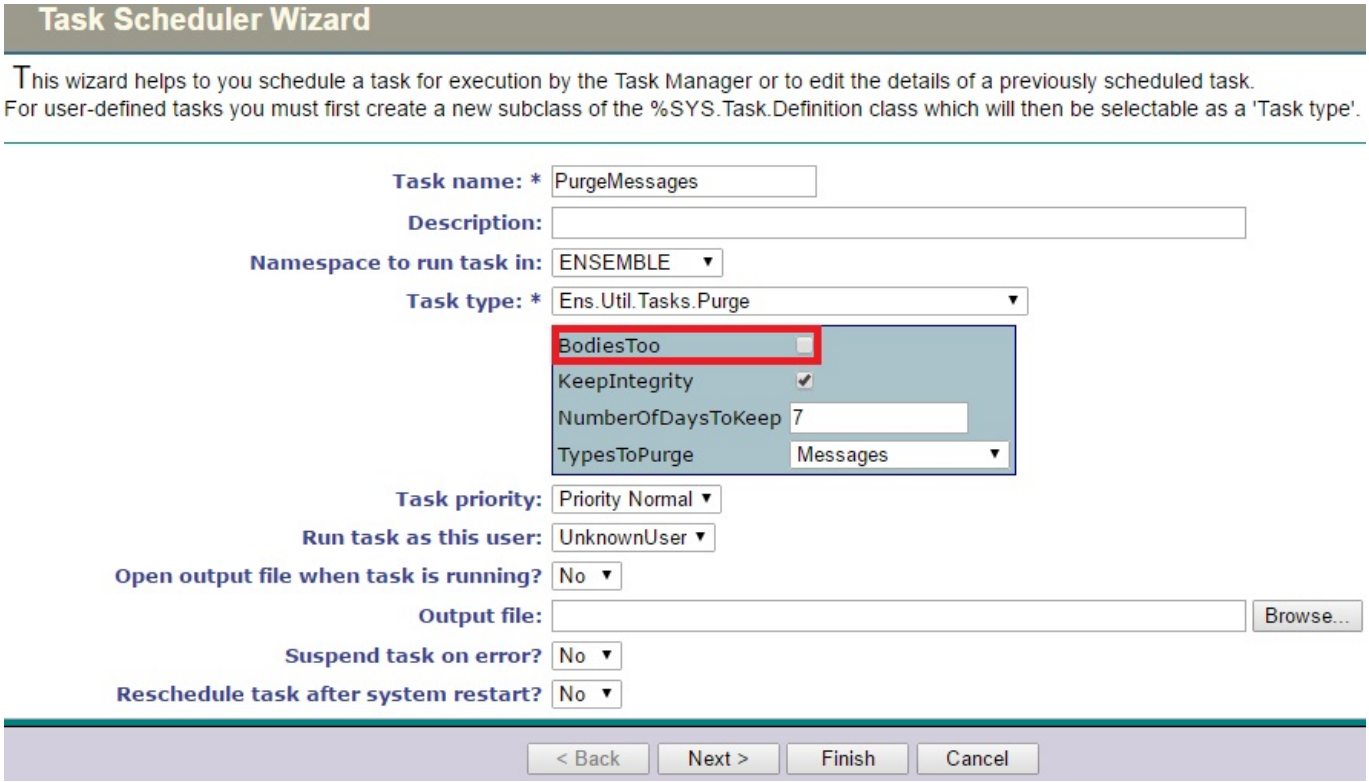
In this article, we will discuss about Orphaned Messages.

What is an Orphaned Message

Every message body is associated with a message Header which holds the metadata. The Header holds information like source configuration name, target configuration name, time created, time processed, associated message body reference, session information, message body class name, message status. When there are message body records that do not have their corresponding Header records those are called Orphan message bodies. We will discuss possible causes which could end up with orphan message bodies.

Purging only Headers

In the Purge task setup, the **BodiesToo** setting is to specify whether to purge message bodies too along with message headers. If this setting is turned **OFF** the purge task will delete only message headers retaining message bodies. These message bodies will become orphan records since the referenced header is deleted. If you purge message headers but retain the message bodies, then the Management Portal provides no way to purge the orphaned message bodies. In this case, you must purge the message bodies programmatically.



Please refer documentation about purge task

http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=EGMG_purge#EGMG_purge_basic

Complex Message Body Class (Object-valued Properties)

When Ensemble purges a message body, it does not necessarily delete object-valued properties of the message body. Specifically, it deletes other objects only if these are serial objects or are child objects (as defined by a relationship). For other objects, you must appropriately handle the deletion by defining a delete trigger or implementing the **%OnDelete()** method in the message body class.

A sample code for OnDelete implementation

```
Class Sample.Address Extends %Persistent{
/// The street address.
Property Street As %String(MAXLEN = 80);
/// The city name.
Property City As %String(MAXLEN = 80);
/// The 2-letter state abbreviation.
Property State As %String(MAXLEN = 2);
/// The 5-digit U.S. Zone Improvement Plan (ZIP) code.
Property Zip As %String(MAXLEN = 5);
}
Class Sample.Person Extends %Persistent{
/// Person's name.
Property Name As %String [ Required ];
/// Person's Social Security number. This is validated using pattern match.
Property SSN As %String(PATTERN = "3N1""-""2N1""-""4N") [ Required ];
/// Person's Date of Birth.
Property DOB As %Date;
/// Person's home address.
Property Home As Address;
/// Person's office address.
Property Office As Address;
///Callback for object deletion
ClassMethod %OnDelete(oid As %ObjectIdentity) As %Status [ Private ]{
      // Delete the property object references.
      Set tSC = $$$OK, tThis = ##class(Sample.Person).%Open(oid)
      If $ISOBJECT(tThis.Home) Set tSC = ##class(Sample.Address).%DeleteId(tThis.Home
.%Id())
      If $ISOBJECT(tThis.Office) Set tSC = ##class(Sample.Address).%DeleteId(tThis.Of
fice.%Id())
      Quit tSC
}
///Callback/Trigger for SQL delete
Trigger OnDelete [ Event = DELETE ]{
      // Delete the property object references. {%%ID} holds the id of the record bei
ng deleted.
      Set tID={%%ID}
      Set tThis = ##class(Sample.Person).%OpenId(tID)
      If $ISOBJECT(tThis.Home) Do ##class(Sample.Address).%DeleteId(tThis.Home.%Id())
      If $ISOBJECT(tThis.Office) Do ##class(Sample.Address).%DeleteId(tThis.Office.%I
d())
      Quit
}
}
```

Message objects created but never sent to another host

When a message is sent/forwarded to another host Ensemble creates a new Message Header and associates the corresponding message body. If the message body/object instance created in a Business Service/Process is saved to disk/database but never sent to another host in the production it will have no associated Header and will remain as an Orphan Message Body. The best practice is not to create a message body unless it will be forwarded or do not call the %Save() on the object instance (The SendRequestSync/SendRequestAsync API will save it before putting the message into target config queue). This way the message body object will not be persisted unless it is sent to another host.

The most common reason for this problem is where developers

a. Clone the message body and never forward the cloned message body
b. Message body objects created in Context variables (BPL) and never forwarded.

Effects of Orphan Messages

Orphan messages will not be purged by the purge task. These messages will consume disk space and as they grow in number proportionately the disk usage will increase. The disk space usage will not be only from the message body data, but also from any indexes/search table entries for each of these orphan message body records.

Identifying Orphan Messages

The existence of orphan messages can be identified by querying the message header and body. Each message header references the corresponding message body. The message body object Id reference is stored in the MessageBodyId property of the header and the message body class name is stored in MessageBodyClassName property of the header.

An example query to find orphan messages in HL7 Message table:

```
SELECT HL7.Id  FROM EnsLib_HL7.Message HL7

  LEFT JOIN Ens.MessageHeader hdr

  ON HL7.Id=hdr.MessageBodyId

  WHERE hdr.MessageBodyId IS NULL
```

The above query will return all the HL7 messages which don't have their corresponding headers. The query can be modified to query for any other message types simply replacing the message body table name.

Purging Orphan Messages

The Management Portal provides no way to purge the orphaned message bodies. In this case, we must purge the message bodies programmatically. In the **ENSDEMO** database, the class **Demo.Util.CleanupSet** provides an example of how you might do this. This routine does a deep purge too taking care of the object property references as well.

There is another routine written for reference to help purge orphan messages, but this routine doesn't do deep purge and only helps delete message bodies. I have included the link for github below to download the source:

https://gist.github.com/suriyasv/2ed7f2dbcfd8c79f3b9938762c17c0b5

The best practice is to always

1. Avoid programming errors (as discussed earlier) to prevent orphan messages

2. Setup purge task with BodiesToo setting turned OFF only if bodies are required knowing that these message bodies can only be purged programmatically.
3. Implementing Relationships or OnDelete implementations for Persistent Object properties.

I hope that this article would be useful when building your production. Please let us know if you have any questions or concerns. Thank you.

#Interoperability #Monitoring #System Administration #Tips & Tricks #Ensemble

Source URL: https://community.intersystems.com/post/ensemble-orphaned-messages