www.bettercodingtools.com

Contents

Message Broker Coding style

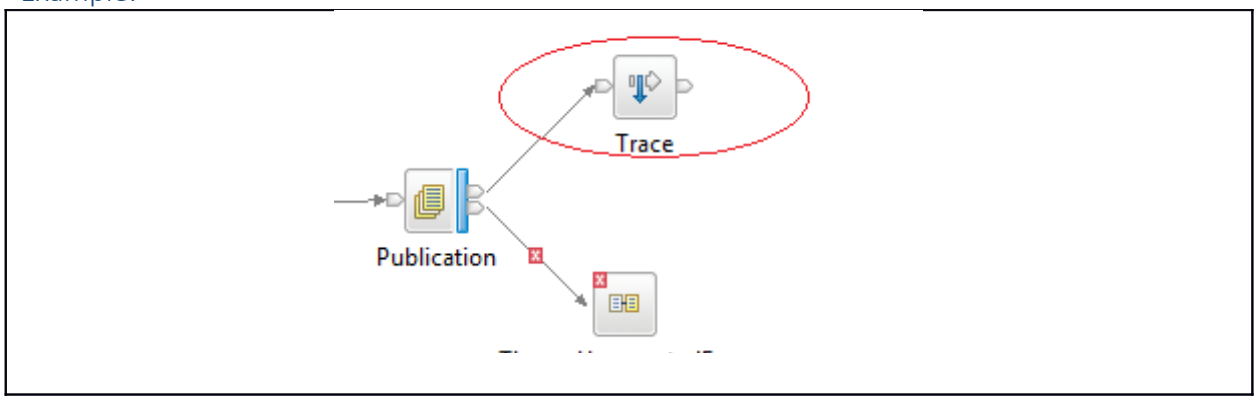
Rule: Trace nodes should not be used

Sonar Rule: R108

Rational:

Trace nodes should not be used in production code.

Example:



Preferred:

NA

References:

NA

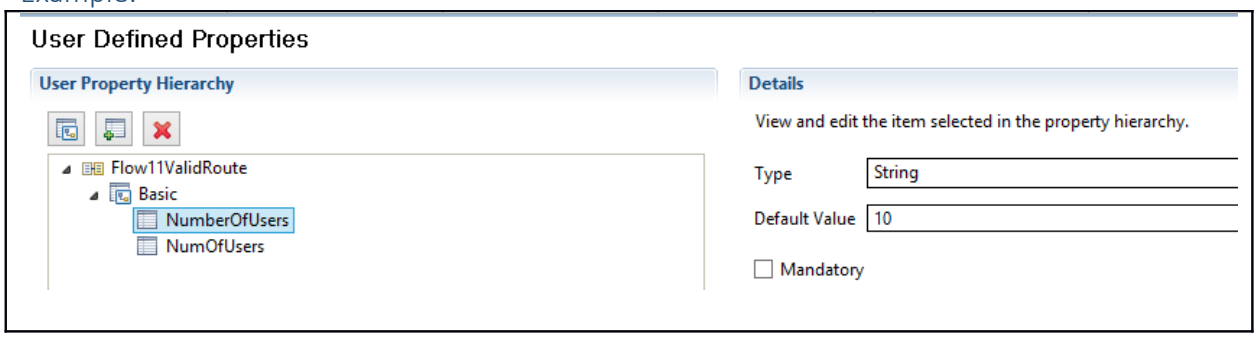
Rule: The flow contains a duplicate UDP/property default value

Sonar Rule: R90

Rational:

UDP's (user defined properties) in the same flow with the same value could be duplicates.

Example:



Preferred:

Check that UDP's with the same values are required.

References:

NA

Rule: It is good programming practice to give an EXTERNAL variable an initial value

Sonar Rule: R61

Rational:

By giving an EXTERNAL variable a default value it makes understanding the code easier and can simplify the default deployment process.

Example:

```
DECLARE deployEnvironment EXTERNAL CHARACTER;
```

Preferred:

```
DECLARE deployEnvironment EXTERNAL CHARACTER 'Dev';
```

References:

http://www-01.ibm.com/support/knowledgecenter/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ak04980_.htm

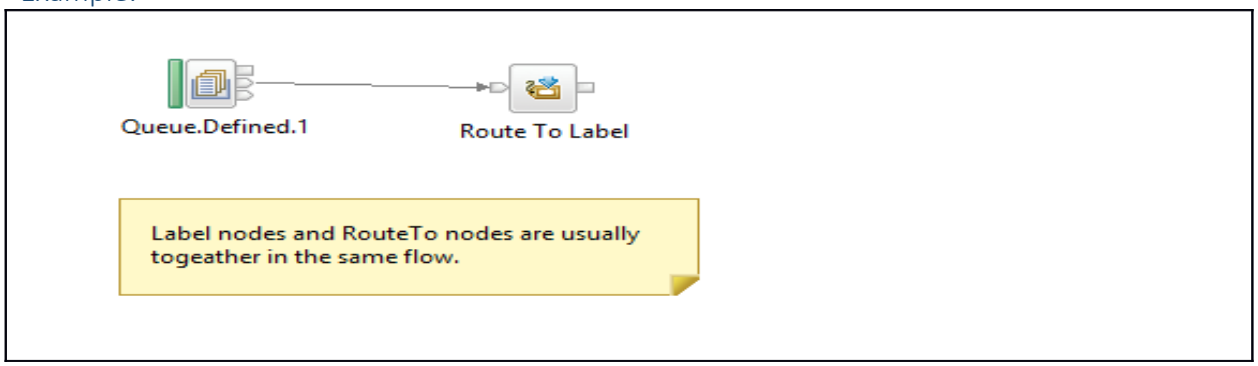
Rule: Usually the RouteTo and Label are in the same flow as to make things more readable

Sonar Rule: R60

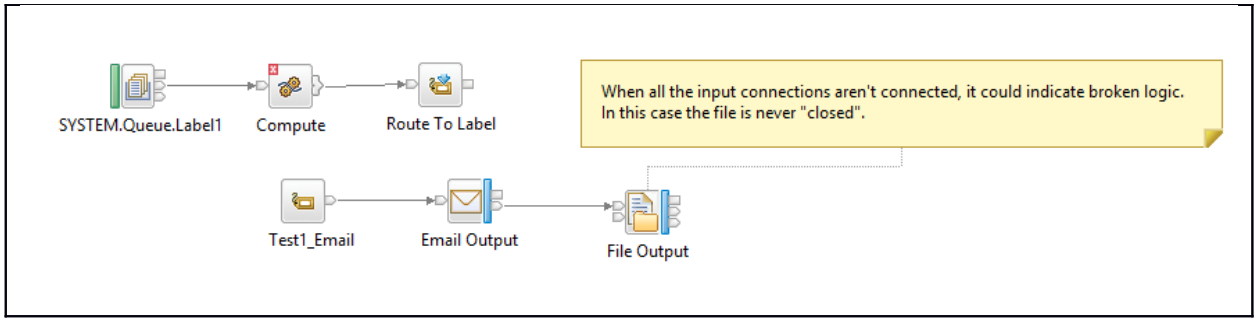
Rational:

RouteTo and a Label nodes can be split across different flows, but they have to be in the same execution group and both running. Keeping them in the same flow makes the logic easier to understand.

Example:



Preferred:



References:

NA

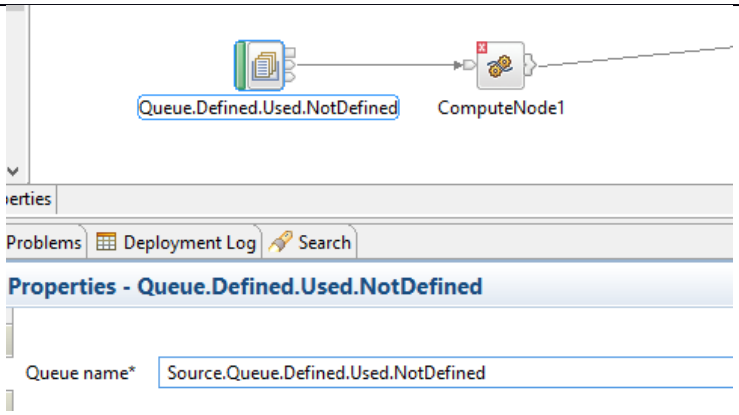
Rule: MQNode name within the flow doesn't match the Queue name

Sonar Rule: R32

Rational:

By having the name of the Input/Output MQ nodes match to the queue that they reference it makes it easier to understand a flow within the context.

Example:



Preferred:

Change the node name to reflect the queue that it reads or writes to.

References:

NA

Rule: Environment values should be under the Variables subtree

Sonar Rule: R22

Rational:

Having environment values in a standard place helps orgainze the code.

Example:

```
SET Environment.MQMD = InputRoot.MQMD;
```

Preferred:

```
SET Environment.Variables.MQMD = InputRoot.MQMD;
```

References:

NA

General Coding Style

Rule: The line is extra long and may cause issues being viewed

Sonar Rule: R19

Rational:

Long lines are harder to understand and may be inherently complex.

Example:

```
IF (XMLNSC.Data.Content.Value = LocalEnvironment.ThePersonNameWeAreProcessing.Name) THEN
```

Preferred:

Split lines to make them readable

```
IF (XMLNSC.Data.Content.Value =
        LocalEnvironment.ThePersonNameWeAreProcessing.Name) THEN
```

Or

```
BOOLEAN equal = FALSE;
SET equals = XMLNSC.Data.Content.Value =
LocalEnvironment.ThePersonNameWeAreProcessing.Name;

IF (equal = TRUE)  THEN
```

References:

NA

Rule: Multiple statements on the same line

Sonar Rule: R47

Rational:

Multiple statements on the same line make the code difficult to read and understand.

Example:

```
SET description = 'Fred'; SET age='21'; SET height = 20;
```

Preferred:

Split lines to make them readable

```
SET description = 'Fred';
SET age='21';
SET height = 20;
```

References:

NA

Rule: Case has no default ELSE statement

Sonar Rule: R46

Rational:

A case statement with no default path could indicate a logic error. It can also be confusing.

Example:

```
SET description =
	CASE age
		WHEN '0' THEN 'really young'
		WHEN '100' THEN 'really old'
	END;
```

Preferred:

```
SET description =
	CASE age
		WHEN '0' THEN 'really young'
		WHEN '100' THEN 'really old'
		ELSE 'Somewhere inbetween'
	END;
```

References:

http://coding.tocea.com/java/sf_switch_no_default/

http://checkstyle.sourceforge.net/config_coding.html#MissingSwitchDefault

Rule: Case statement has single WHEN. Could be replaced by an IF statement

Sonar Rule: R45

Rational:

Case statements with only 2 conditions are essentially “IF” conditions. Use an “IF” condition as it is easier to read.

Example:

```
CASE Environment.Variables.PersonType
	WHEN 1 THEN
		SET Environment.Variables.ItsABoy = 'TRUE';
		SET Environment.Variables.LowerName = lowerType;
END CASE;
```

Preferred:

```
Is equivalent to:
```

```
IF (Environment.Variables.PersonType = 1) THEN
       SET Environment.Variables.ItsABoy = 'TRUE';
       SET Environment.Variables.LowerName = lowerType;
END IF;
```

References:

NA

Rule: The line is extra long and may cause issues being viewed

Sonar Rule: R19

Rational:

Longer lines may require scrolling to be seen by developers on their screens. Also, files with longer lines are more difficult to print.

The default value is 130, but can be over-ridden by setting the property

```
sonar.mb.esql.maxlinesize=size
```

In the sonar.properties file for the project.

Example:

NA

Preferred:

Reformat longer lines to be more readable.

References:

NA

Rule: Keywords should be in upper case

Sonar Rule: R1

Rational:

For highlighting keyworcds in ESQL, they should be in uppercase.

Example:

```
declare ptrException reference to InputTree.*[1];
```

Preferred:

```
DECLARE ptrException REFERENCE TO InputTree.*[1];
```

References:

http://www.ibm.com/developerworks/websphere/library/techarticles/0803_shen/0803_shen.html

http://pic.dhe.ibm.com/infocenter/ratdevz/v8r0/index.jsp?topic=%2Fcom.ibm.etools.est.doc%2Fref%2Frsfsql027.html

Complexity

Rule: The parameter on a method/procedure has a short name (and is likely to be meaningless)

Sonar Rule: R42

Rational:

Parameters passed to functions and procedures should be meaningful and ideally reveal their intent where possible.

The minimum length for each parameter that the check will use is controlled by the "sonar.mb.esql.parameterlength" property in the sonar.properties file. The default minimum length is "2".

```
sonar.mb.esql.parameterlength=4
```

Example:

```
CREATE PROCEDURE SetEmailParms (IN Er REFERENCE)
        BEGIN
```

Preferred:

Give procedure/function parameter definitions meaningful names.

```
CREATE PROCEDURE SetEmailParms (IN EmailReference REFERENCE)
        BEGIN
```

References:

NA

Rule: The method/procedure has a higher number of parameters then the threshold

Sonar Rule: R41

Rational:

Lots of parameters for a function or procedure could indicate that the function/procedure is more complicated then it needs to be. This check is controlled by the "sonar.mb.esql.parametercount" property in the sonar.properties file. The default value is "10".

For example

```
sonar.mb.esql.parametercount=5
```

States that if a function/procedure has more then 5 parameters a violation will be issued.

Example:

```
CREATE PROCEDURE SetEmailParms (IN Environment REFERENCE,
                                IN EventCode CHARACTER,
                                IN ReplaceValue CHARACTER,
                                IN RecipientTo CHARACTER,
                                IN Retries INTEGER)
BEGIN
```

Preferred:

Look (in-conjuction with the unused parameters/variables check) at whether all the parameters are required, or whether the procedure could be simplifed.

References:

NA

Rule: Negative IF / ELSE condition

Sonar Rule: R2

Rational:

Negative conditions are harder to understand conceptually then positive conditions.

Example:

```
IF FIELDNAME(CreditorRole.ns:NextParty) is not null THEN
   SET NextPartyCreditorRole = NextPartyCreditorRole + 1;
   MOVE CreditorRole FIRSTCHILD;
ELSE
   LEAVE X;
END IF;
```

Preferred:

```
IF FIELDNAME(CreditorRole.ns:NextParty) is null THEN
   LEAVE X;
ELSE
```

```
   SET NextPartyCreditorRole = NextPartyCreditorRole + 1;

   MOVE CreditorRole FIRSTCHILD;

END IF;
```

These two pieces of code are logically the same but the preferred is more readable.

References:

NA

Database checks

Rule: JDBC has not been configured

Sonar Rule: R4

Rational:

The MB-Precise plugin will attempt to validate your ESQL and MsgFlow code against any databases that the code is using. If the plugin detects database access code (a SELECT, DELETE, INSERT, UPDATE), it will attempt to validate that the tables and columns that the SQL is referencing are valid and consistent.

Example:

NA

Preferred:

Configure in the sonar.properties file the connection information of your database.

```
sonar.mb.jdbc.driver=org.h2.Driver
sonar.mb.jdbc.url=jdbc:h2:mem:test;
sonar.mb.jdbc.user=sa
sonar.mb.jdbc.password=
```

References:

NA

Rule: A table being referenced has not been found in the DB schema

Sonar Rule: R6

Rational:

The MB-Precise plugin will attempt to validate your ESQL and MsgFlow code against any databases that the code is using. If the plugin detects database access code (a SELECT, DELETE, INSERT, UPDATE), it will attempt to validate that the tables and columns that the SQL is referencing are valid and consistent. Inconsistent code could indicate that a table or column name has been misspelled or is missing from the DB environment.

Example:

NA

Preferred:

Check that the table exists in the database and the spelling matches.

References:

NA

Rule: A column being referenced has not been found in the DB schema

Sonar Rule: R7

Rational:

The MB-Precise plugin will attempt to validate your ESQL and MsgFlow code against any databases that the code is using. If the plugin detects database access code (a SELECT, DELETE, INSERT, UPDATE), it will attempt to validate that the tables and columns that the SQL is referencing are valid and consistent. Inconsistent code could indicate that a column name has been misspelled or is missing from the DB environment.

Example:

```
SET LocalEnvironment.Variables.SelectData[] = PASSTHRU('SELECT Name, Age, Height ' ||
                                                       'FROM THEDATA.PersonTable');
```

Preferred:

Check that the column exists in the database in the expected table and the spelling matches.

References:

NA

Rule: A column being referenced has not been indexed. This may be a performance issue

Sonar Rule: R8

Rational:

The MB-Precise plugin will attempt to validate your ESQL and MsgFlow code against any databases that the code is using. If the plugin detects database access code (a SELECT, DELETE, INSERT, UPDATE), it will attempt to validate that the tables and columns that the SQL is referencing are valid and consistent. This warning indicates that columns that a table are being accessed by are not indexed.

Example:

```
SET LocalEnvironment.Variables.SelectData[] = PASSTHRU('SELECT Name, Age, Height ' ||
                                                       'FROM THEDATA.PersonTable' ||
                                                       'WHERE Wieght > ?', weight);
```

Preferred:

This may or not be an issue depending upon the table being accessed size and the frequency of access. Analysis of the amount of expected data in the tables affected should be undertaken with the DBA to see if this will be an issue in a production system.

References:

NA

MQ Configuration

Rule: MQ Definition file has not been configured or doesn't refer to a valid file

Sonar Rule: R10

Rational:

The MB-Precise plugin will attempt to validate your MQ queues and topics against your ESQL and Msgflow code to make sure that all queues being accessed have been defined via a confiuration script, and that all queues defined are being used (that there are no redundant queues in the application).
To do this analysis the tools needs to access the “mqsc configuration file that the queues are defined in.
This error indicates that the queue file path has not been defined or is not valid.

Example:

For example a file might contain queue definitions:

```
********************************************
*  Define the queues for the application
********************************************

DEFINE QL('Unused.Queue.Defined.in.file') REPLACE
* DEFINE QLOCAL ('XML_PASSENGERQUERY_IN') REPLACE
```

Preferred:

Keep the “.mqsc” files up to date with the code to avoid issues when promoting through environments.

References:

NA

Rule: MQ Queue defined but not used in the code

Sonar Rule: R11

Rational:

The MB-Precise plugin will attempt to match queue configuration to queue used. This warning indicates that a queue is defined in the definition but not in the code.
It could mean that

- the queue is used in a different application
- the queue access is dynamic (ie dependant on logic in the code)
- the queue is never used and is redundant

Example:

```
********************************************
*  Define the queues for the application
********************************************
DEFINE QL('Unused.Queue.Defined.in.file') REPLACE
```

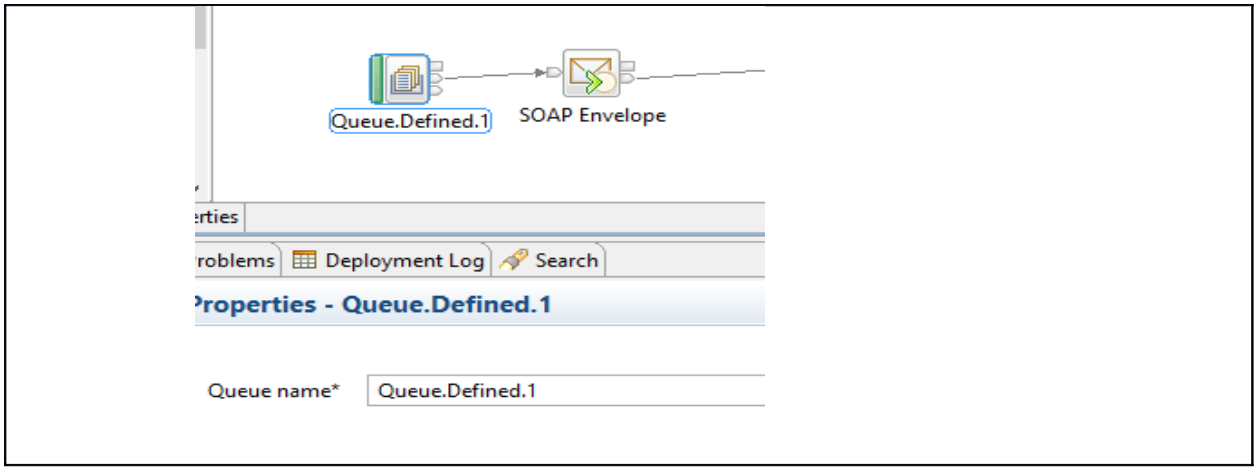
Preferred:
Check that the queue is accessed.

References:
NA

Rule: MQ Queue used in the code but not not listed in the definition file
Sonar Rule: R12
Rational:
The MB-Precise plugin will attempt to match queue configuration to queue used. This warning indicates that is referenced in the code but not defined.
It could mean that

- the queue was created in a different application
- the queue access is dynamic (ie dependent on logic in the code)
- the queue was manually created and should be part of the configuration file

Example:



Preferred:
Check whether the queue accessed should be in the queue configuration file.

References:
NA

Message Broker Performance

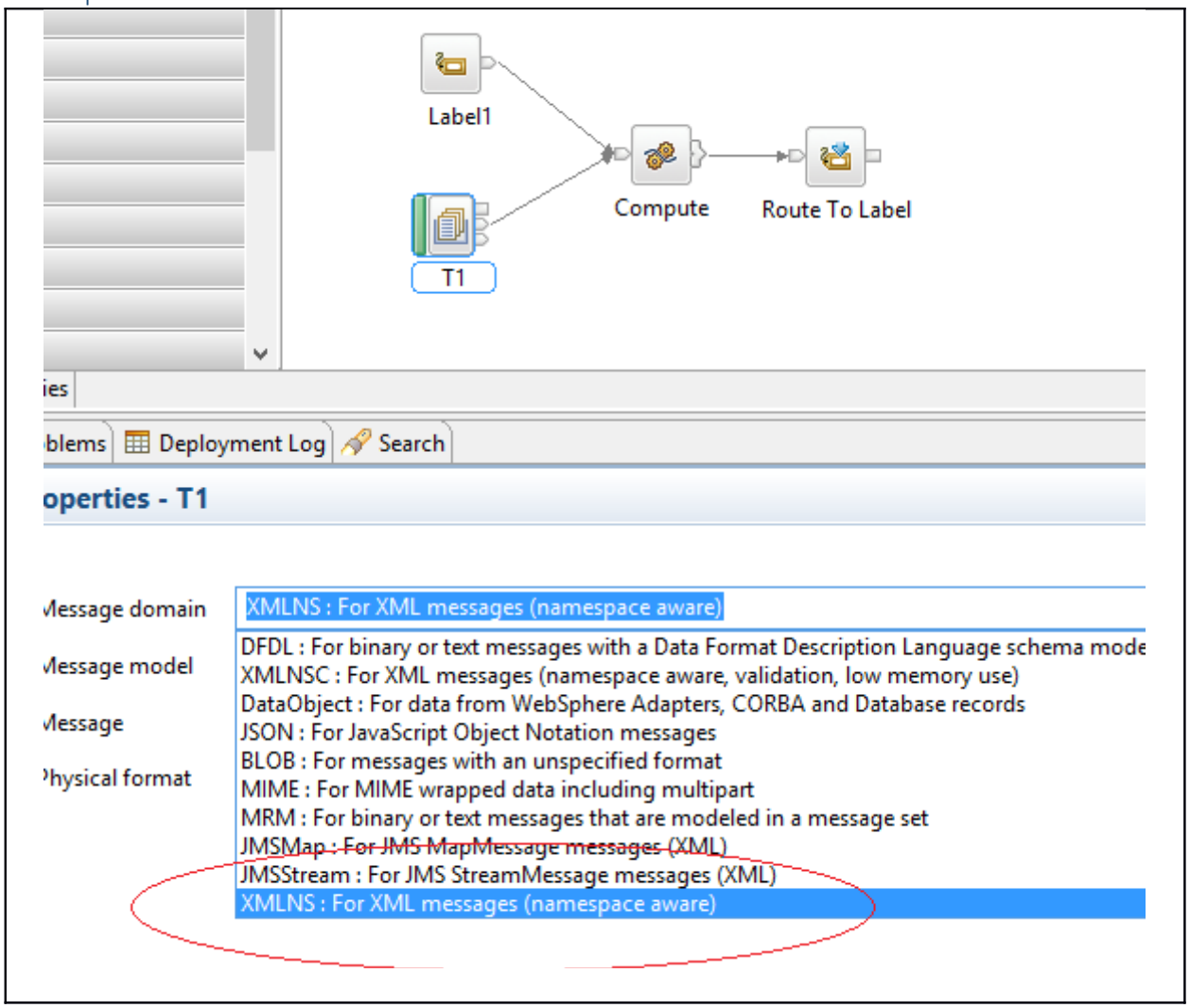
Rule: Use XMLNSC over XMLNS

Sonar Rule: R3

Rational:

Use XMLNSC over XMLNS where possible. XMLNSC is more efficient.

Example:



Preferred:

NA

References:

NA

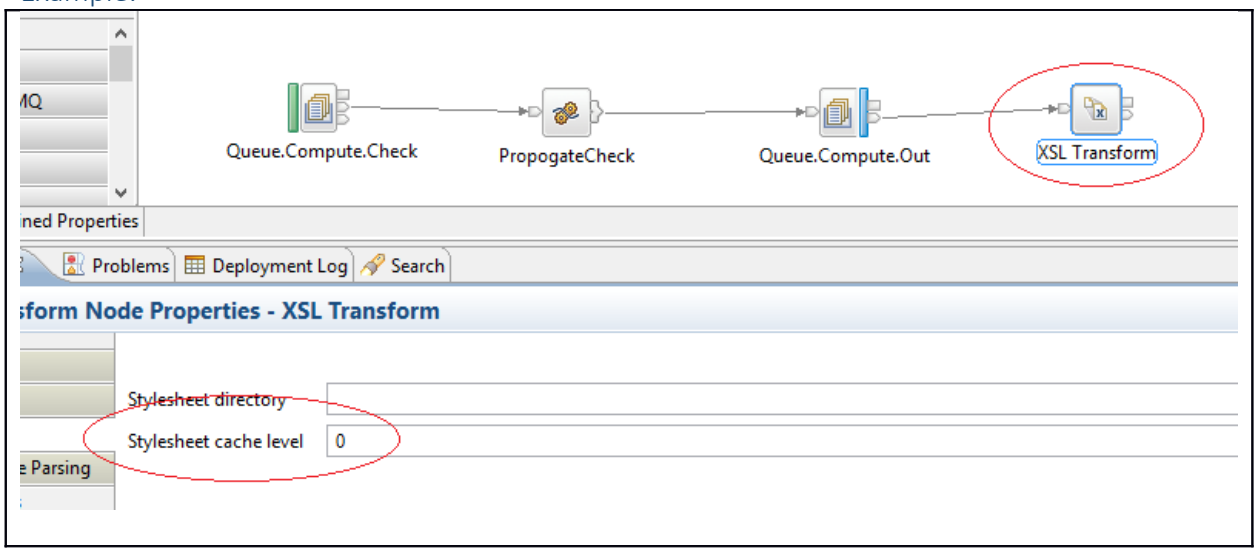
Rule: The XSL cache is set to 0, so style sheets will be compiled each time the node runs

Sonar Rule: R100

Rational:

Caching of style sheets can improve performance.

Example:



Preferred:

NA

References:

NA

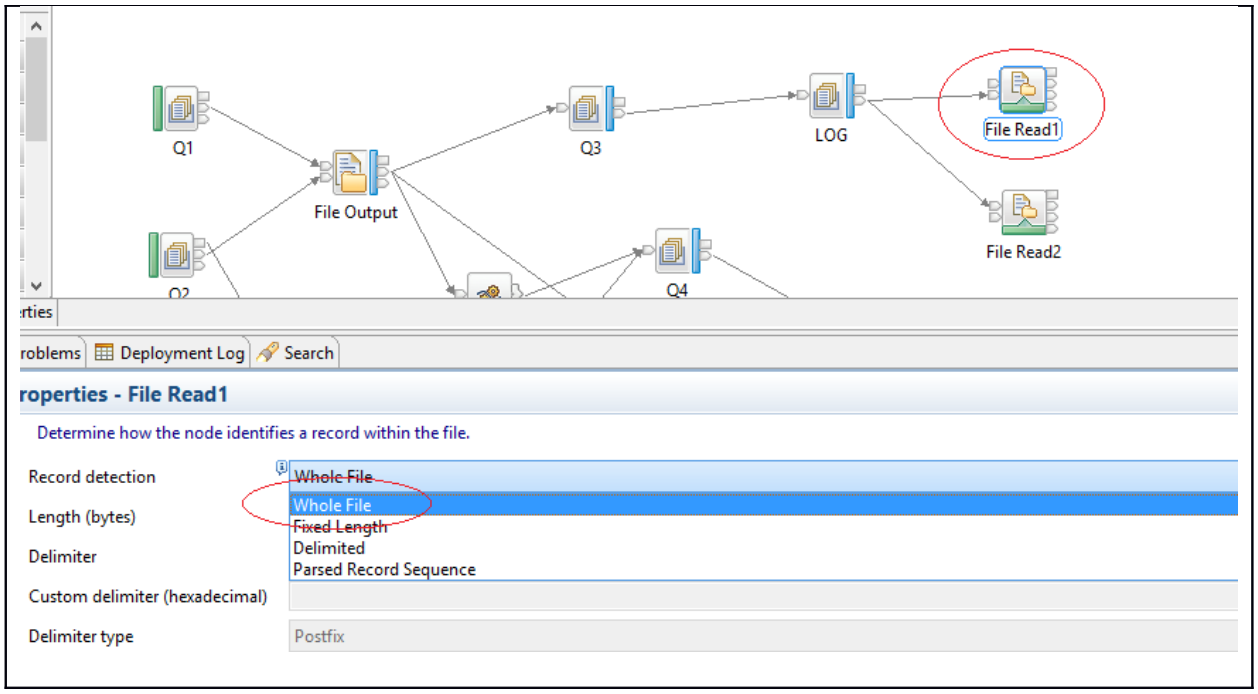
Rule: Reading whole file may cause issues with performance. Split into batches where possible

Sonar Rule: R106

Rational:

Reading the whole file can affect performance.

Example:



Preferred:

NA

References:

NA

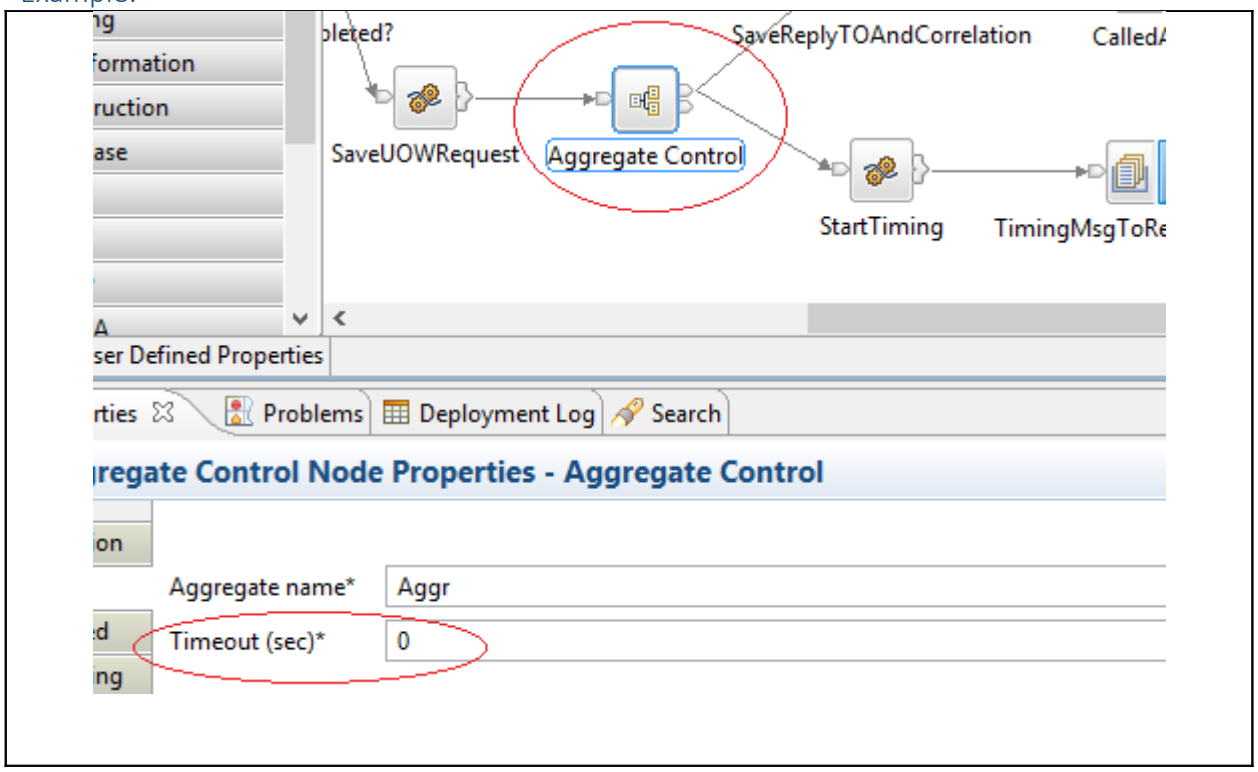
Rule: The AggregateControl Node has an infinite timeout set. This may cause flows to never complete if all replies do not arrive

Sonar Rule: R97

Rational:

An aggregate that waits indefinitely may block the execution group.

Example:



Preferred:

NA

References:

NA

Rule: BITSTREAM is deprecated. Use ASBITSTREAM instead

Sonar Rule: R58

Rational:

BITSTREAM has been deprecated.

Example:

```
SET Env.Person.PersonId =BITSTREAM(Env.Person.Image);
```

Preferred:

```
SET Env.Person.PersonId = ASBITSTREAM(Env.Person.Image, InputRoot.Properties.Encoding,
InputRoot.Properties.CodedCharSetId);
```

References:
NA

Rule: SLEEP() has been called. Calling SLEEP blocks the flow in the execution group

Sonar Rule: R44

Rational:
Calling SLEEP within an ESQL file causes the thread to pause, which prevents the execution group form processing any other messages for that flow. Calling SLEEP could indicate an issue with the architecture that may be able to be addressed in a non-blocking fashion.

Example:
```
CALL SLEEP(1000);
```

Preferred:
Limit the calls to SLEEP and investigate alternatives.

References:
NA

Rule: Using a SELECT * will affect the resources used (memory) if not all the fields are required

Sonar Rule: R39

Rational:
When Message Broker brings back the records from a database query, they take resources (CPU and memory). If the a wider select is used then what is required by the logic, then more CPU and memory is required for the query to run.

Example:
```
SET LocalEnvironment.Variables.SelectData[] = PASSTHRU('SELECT * ' ||
                                                       'FROM THEDATA.PersonTable' ||
                                                       'WHERE Wieght > ?', weight);
```

Preferred:
Use a narrower list of fields/columns that match what is required by the logic where possible. An alternative is to make use of views that only provide the necessary data when a “SELECT *” is used.

References:
NA

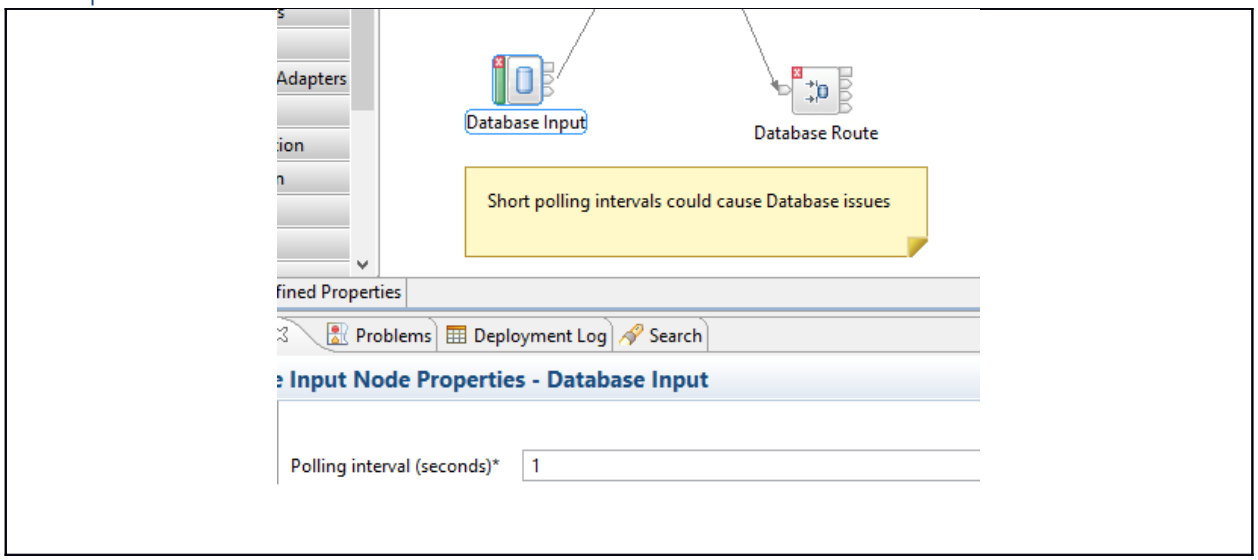
Rule: Database access with low polling interval could cause database contention issues for other applications/code

Sonar Rule: R38

Rational:

The DatabaseInput node polls the database at a set interval. If that interval is low, this could cause issues with other users running queries against the database.

Example:



Preferred:

Analyse the database usage and load to make sure that multiple Message Broker DatabaseInput nodes aren't causing database contention.

References:

NA

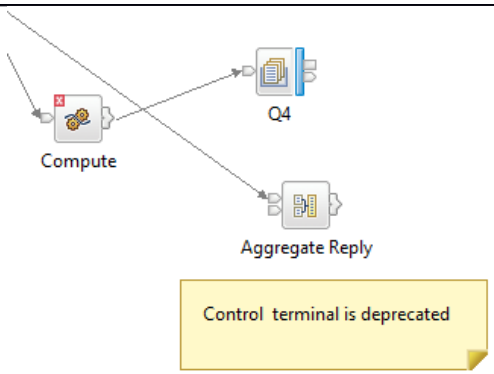
Rule: A terminal that has been deprecated is being used

Sonar Rule: R88

Rational:

The AggregateReply node 'control' terminal has been deprecated.

Example:



Preferred:

Make use of the AggregateControl component.

References:

http://www-01.ibm.com/support/knowledgecenter/SSKM8N_8.0.0/com.ibm.etools.mft.doc/ac04750_.htm

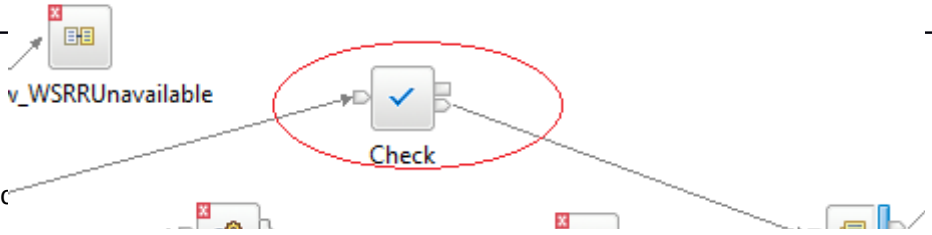
Rule: Check node found in the flow. Check node has deprecated by the validation node

Sonar Rule: R37

Rational:

The check node is deprecated and been replaced by a better performing validation node.

Example:



Preferred:

Replace chec

References:

NA

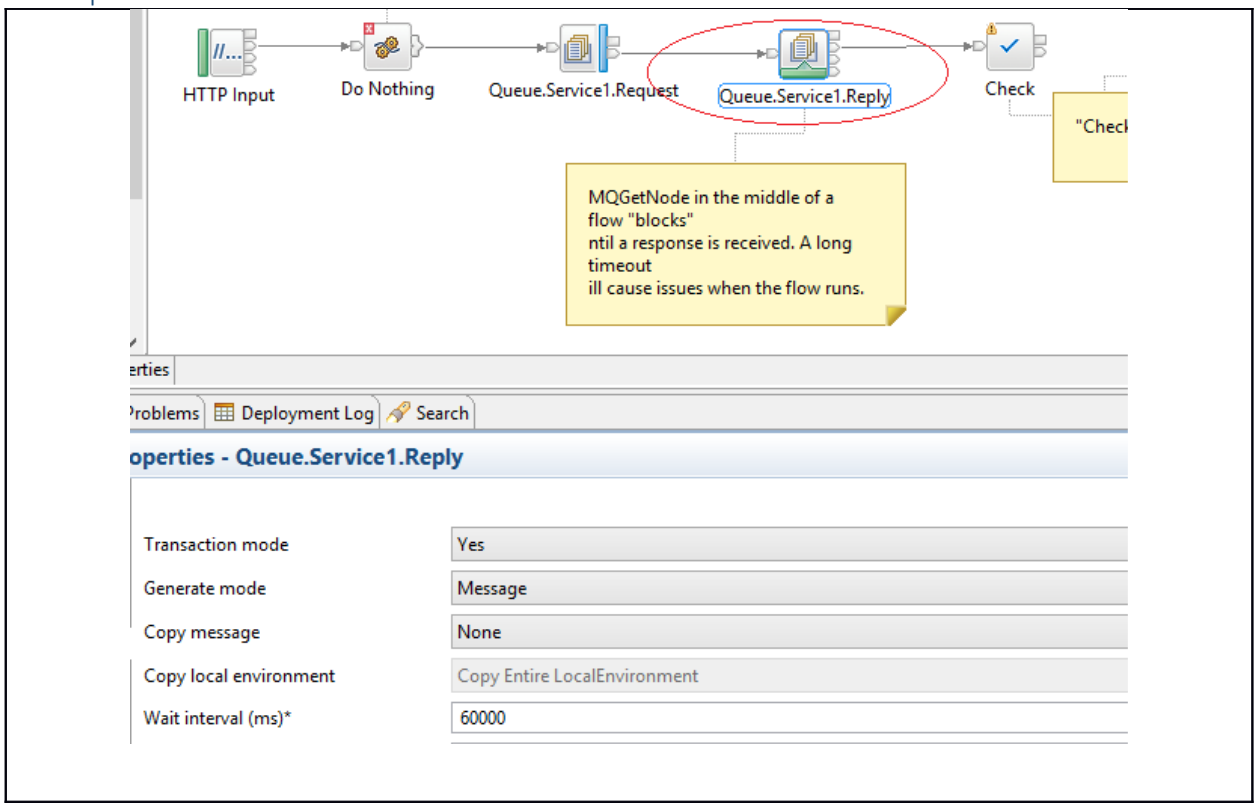
Rule: The node has a very long delay waiting for a response. This will cause blocking of the runtime and could suggest issue with the design/architecture

Sonar Rule: R34

Rational:

When waiting for a response from a queue (via an MQGet), the flow is essentially blocked. A long waiting time will affect through-put and the performance of broker.

Example:



Preferred:

Alternative design patterns are available that may allow long waits to be avoided.

References:

NA

Rule: Use LocalEnvironment over Environment

Sonar Rule: R23

Rational:

The different environment trees have different scopes at runtime. The LocalEnvironment only lives as long as the compute node, so it is preferred to the Environment that lives as long as the flow.

Example:

```
WHILE bLoop <= Cardinality(Environment.Variables.AListOfStuff[]) DO
        SET Environment.Variables.StuffLocation[bLoop+1] =
        Environment.Variables.AListOfStuff[bLoop].LocationValue1;
        SET bLoop = bLoop + 1;
END WHILE;
```

Preferred:

```
SET LocalEnvironment.P1.Name = OutputRoot.XMLNSC.Request.Person.Name;
```

References:

NA

Rule: Avoid using CARDINALITY within loops

Sonar Rule: R21

Rational:

A CARDINALITY check is costly in terms of CPU. Having the check as a loop condition or within a loop should be avoided if possible.

Example:

```
WHILE bLoop <= Cardinality(Environment.Variables.AListOfStuff[]) DO
        SET Environment.Variables.StuffLocation[bLoop+1] =
        Environment.Variables.AListOfStuff[bLoop].LocationValue1;
        SET bLoop = bLoop + 1;
END WHILE;
```

Preferred:

```
DECLARE endLoop INTEGER;
SET  endLoop = Cardinality(Environment.Variables.AListOfStuff[]);
WHILE bLoop <= endLoop DO
        SET Environment.Variables.StuffLocation[bLoop+1] =
Environment.Variables.AListOfStuff[bLoop].LocationValue1;
        SET bLoop = bLoop + 1;
END WHILE;
```

References:

http://www-01.ibm.com/support/knowledgecenter/SSMKHH_9.0.0/com.ibm.etools.mft.doc/bj28653_.htm

Rule: Use XMLNSC over XMLNS

Sonar Rule: R101

Rational:

The XMLNS parser has been deprecated. The XMLNSC parser is more efficient and uses less resources. It also allows for better levels of validation.

Example:

```
SET Environment.Variables.P1.State = OutputRoot.XMLNS.Request.Person.State;
```

Preferred:

```
SET Environment.Variables.P1.State = OutputRoot.XMLNSC.Request.Person.State;
```

References:

http://www-01.ibm.com/support/knowledgecenter/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ad70530_.htm

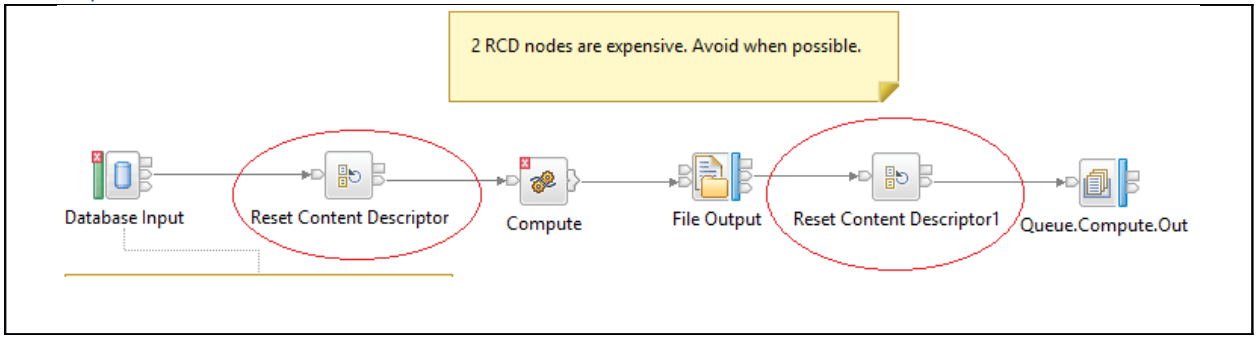
Rule: Two or more RCD nodes in the same flow path

Sonar Rule: R26

Rational:

RCD nodes are expensive, having 2 or more in the same flow path could cause performance issues. “Avoid using Reset Content Descriptor nodes. An RCD node is intended to change the message domain which actually parses the complete message tree. This is both memory and CPU intensive activity.”

Example:



Preferred:

Look at whether an RCD could be replaced with an ASBISTREAM or some alternate approach.

References:

http://www.ibm.com/developerworks/websphere/library/techarticles/0809_kudikala/0809_kudikala.html

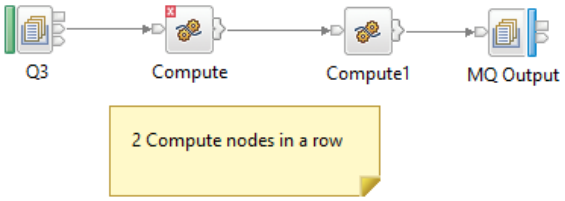
Rule: The flow has two or more compute nodes in a row

Sonar Rule: R14

Rational:

Compute nodes need to do resource intensive parsing of messages. Two nodes in a row will be slower and take more resources then one node performing the same logic.

Example:



Preferred:

Rationalize the logic to have as few compute nodes as possible in each flow.

References:

http://linderalex.blogspot.com.au/2010/07/developing-in-websphere-message-broker.html

Rule: Navigating message tree could be replaced by a reference

Sonar Rule: R15

Rational:

Navigating the message tree can cause re-parsing of the message. By making use of a reference the code runs faster.

Example:

```
SET personName = OutputRoot.XMLNSC.Request.Person.Name;
SET Environment.Variables.P1.Name = OutputRoot.XMLNSC.Request.Person.Name;
SET Environment.Variables.P1.Age = OutputRoot.XMLNSC.Request.Person.Age;
SET Environment.Variables.P1.PostCode = OutputRoot.XMLNSC.Request.Person.PostCode;
SET Environment.Variables.P1.FirstName = OutputRoot.XMLNSC.Request.Person.FirstName;
SET Environment.Variables.P1.LastName = OutputRoot.XMLNSC.Request.Person.LastName;
```

Preferred:

```
DECLARE reqRef REFERENCE TO OutputRoot.XMLNSC.Request.Person;
SET personName = reqRef.Name;
SET Environment.Variables.P1.Name = reqRef.Name;
SET Environment.Variables.P1.Age = reqRef.Age;
SET Environment.Variables.P1.PostCode = reqRef.PostCode;
SET Environment.Variables.P1.FirstName = reqRef.FirstName;
SET Environment.Variables.P1.LastName = reqRef.LastName;
```

References:

http://www-01.ibm.com/support/knowledgecenter/SSMKHH_9.0.0/com.ibm.etools.mft.doc/bj28653_.htm

Rule: CopyEntireMessage makes calling CopyMessageHeaders redundant

Sonar Rule: R111

Rational:

The code generated by message broker may need to tuned.

Example:

```
CREATE COMPUTE MODULE Backlog10_SoapNodeRepliesInvalid_PathTwo
    CREATE FUNCTION Main() RETURNS BOOLEAN
    BEGIN
        CALL CopyMessageHeaders();
        CALL CopyEntireMessage();
        RETURN TRUE;
    END;

    CREATE PROCEDURE CopyMessageHeaders() BEGIN
        DECLARE I INTEGER 1;
        DECLARE J INTEGER;
        SET J = CARDINALITY(InputRoot.*[]);
        WHILE I < J DO
            SET OutputRoot.*[I] = InputRoot.*[I];
            SET I = I + 1;
        END WHILE;
    END;

    CREATE PROCEDURE CopyEntireMessage() BEGIN
        SET OutputRoot = InputRoot;
    END;
END MODULE;
```

Preferred:

NA

References:

NA

Message Broker Logic failures

Rule: Should check that the last MOVE completed

Sonar Rule: R126

Rational:

When using 'MOVE', the variable or reference can be set to undefined and cause logic errors or exceptions. It is good defensive programming practice to check the 'MOVE' completed successfully.

Example:

```
*******************************************************************************
EGIN
    -- Create a reference to the first child of the exception list
    declare ptrException reference to InputTree.*[1];
    -- keep looping while the moves to the child of exception list work
    WHILE lastmove(ptrException) DO
        -- store the current values for the error text
        IF ptrException.Number is not null THEN
            SET messageNumber = ptrException.Number;
            SET messageText = ptrException.Text;
            IF (messageText = 'User exception thrown by throw node') THEN
                SET messageText = ptrException.Insert[1].Text;
            END IF;
        END IF;
        -- now move to the last child which should be the next exceptionlist
        move ptrException lastchild;
    END WHILE;
ND;
```

Preferred:

NA

References:

NA

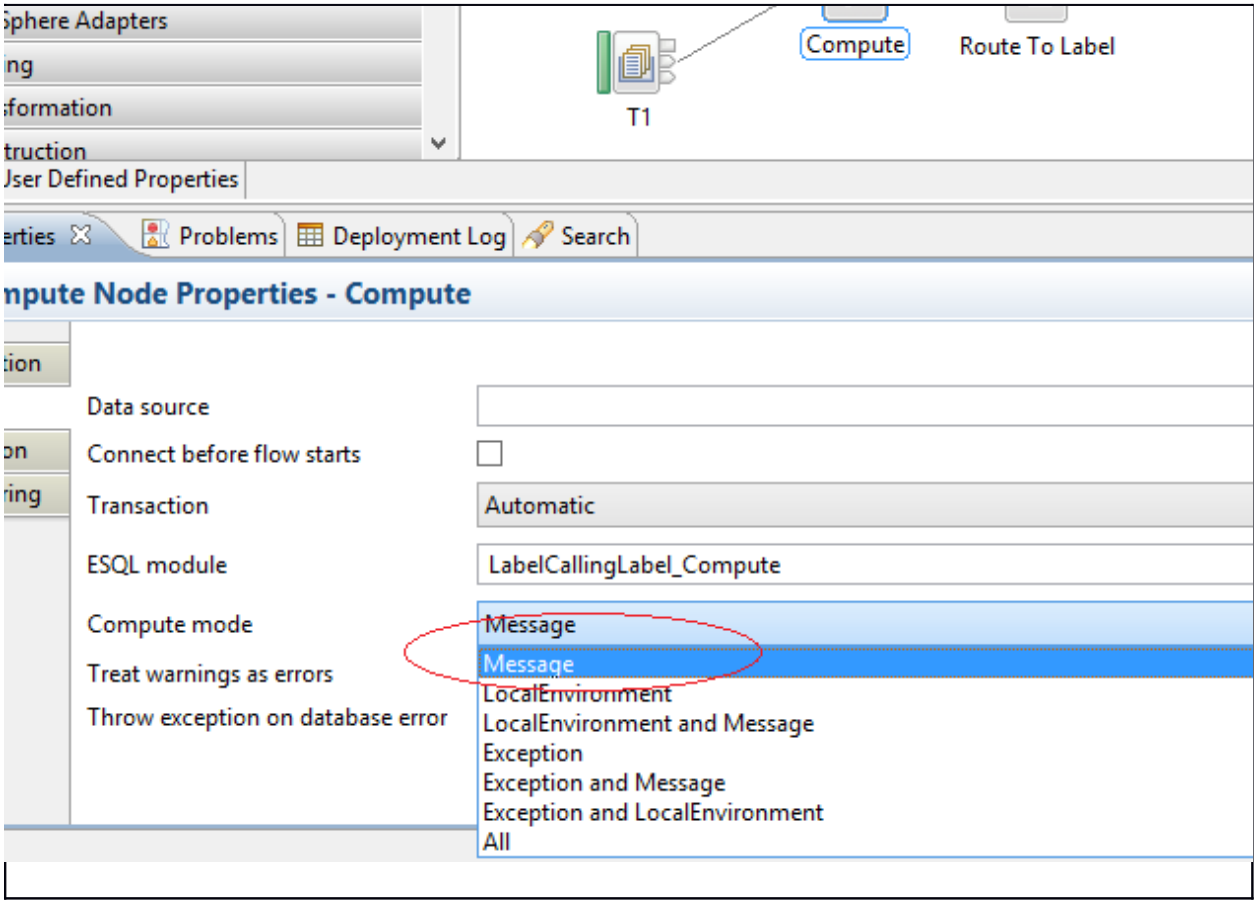
Rule: The compute mode is message but the message is never read or written

Sonar Rule: R93

Rational:

If the message is not being used then either the 'Compute Node' can be changed to be one of the other settings as to be more efficient. Otherwise if the message should be changed then this could indicate a logic error in the associated ESQL.

Example:



Preferred:
NA

References:
NA

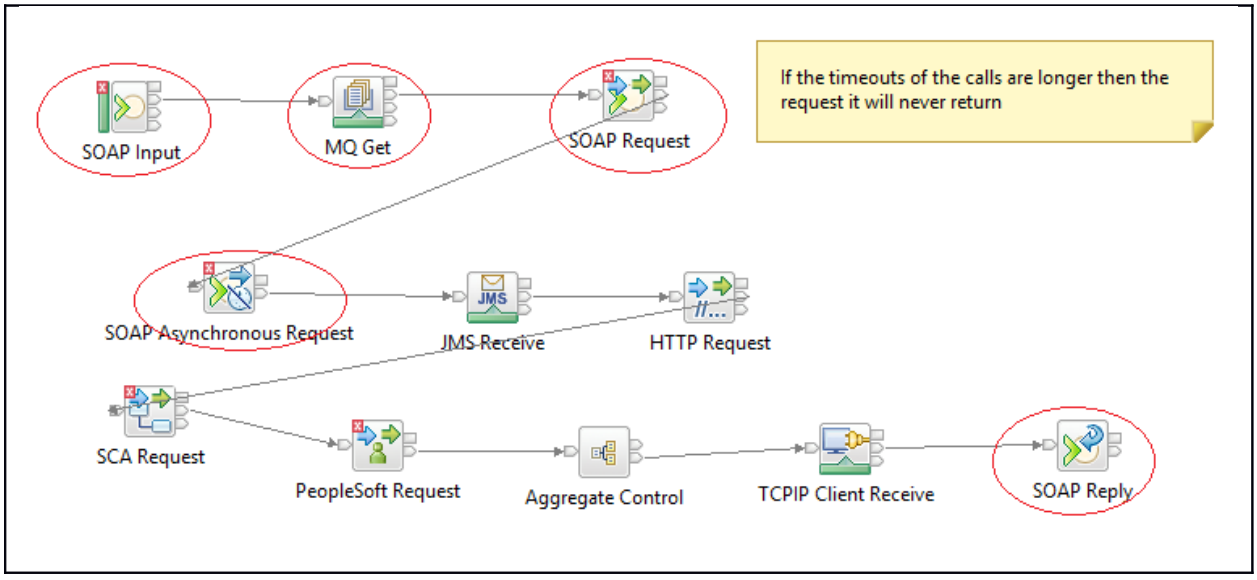
Rule: The timeouts on the nodes in the flow are potentially longer than the allowed delay on the input node

Sonar Rule: R98

Rational:

If the input timeout and the maximum elapsed time that the nodes within the flow can take are not aligned then the request can timeout before the response has been created.

Example:



Preferred:

Adjust the timeout of the calls or the timeout of the request. As an alternative, you could look to make the requests idempotent to allow requests to be resent.

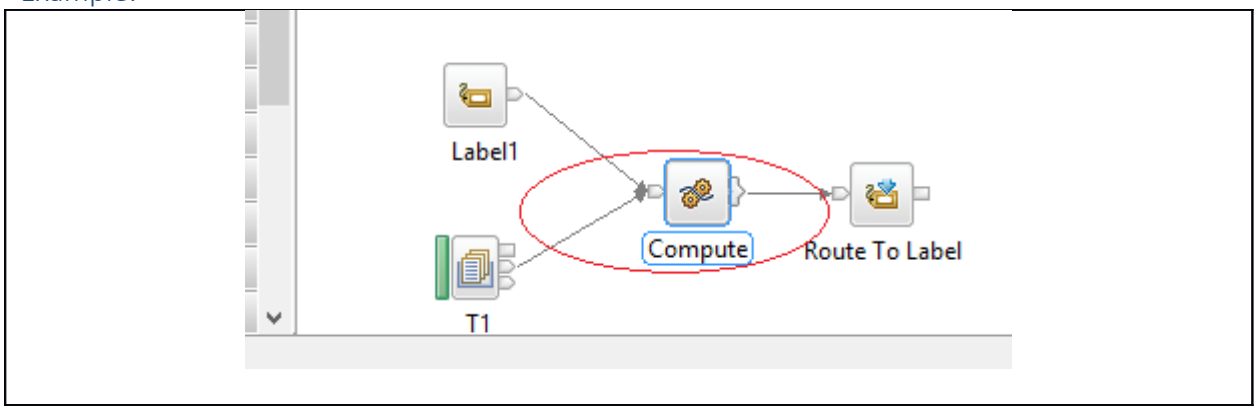
References:
NA

Rule: The compute node never creates an output message

Sonar Rule: R94

Rational:
If the compute node is not creating an output message then this could indicate a logic failure.

Example:



Preferred:
NA

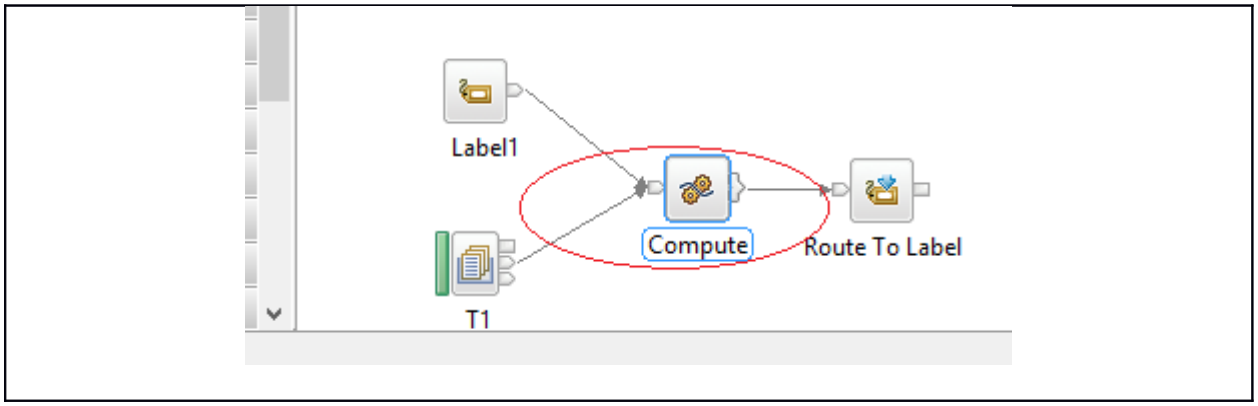
References:
NA

Rule: The main method is referred to by more then 1 compute node

Sonar Rule: R95

Rational:
If the compute nodes share an ESQL file, then they cannot be changed without altering the logic of the other flow. It will also become confusing to maintain the logic.

Example:



Preferred:

If common logic is required then it may be preferable to create a common function/procedure.

References:

NA

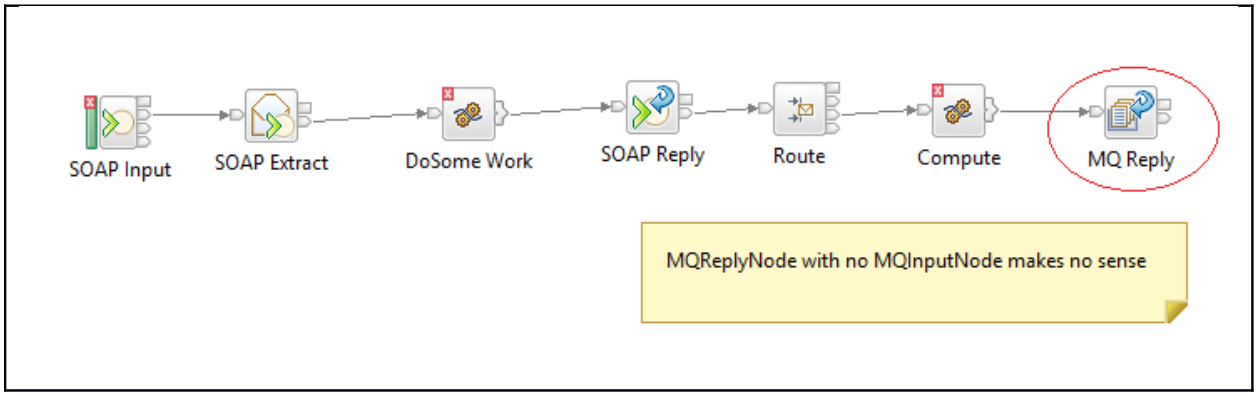
Rule: Flow contains an MQReplyNode without an MQInputNode

Sonar Rule: R96

Rational:

MQReplyNode does not match to an MQInputNode, you can only reply to an incoming message.

Example:



Preferred:

NA

References:

NA

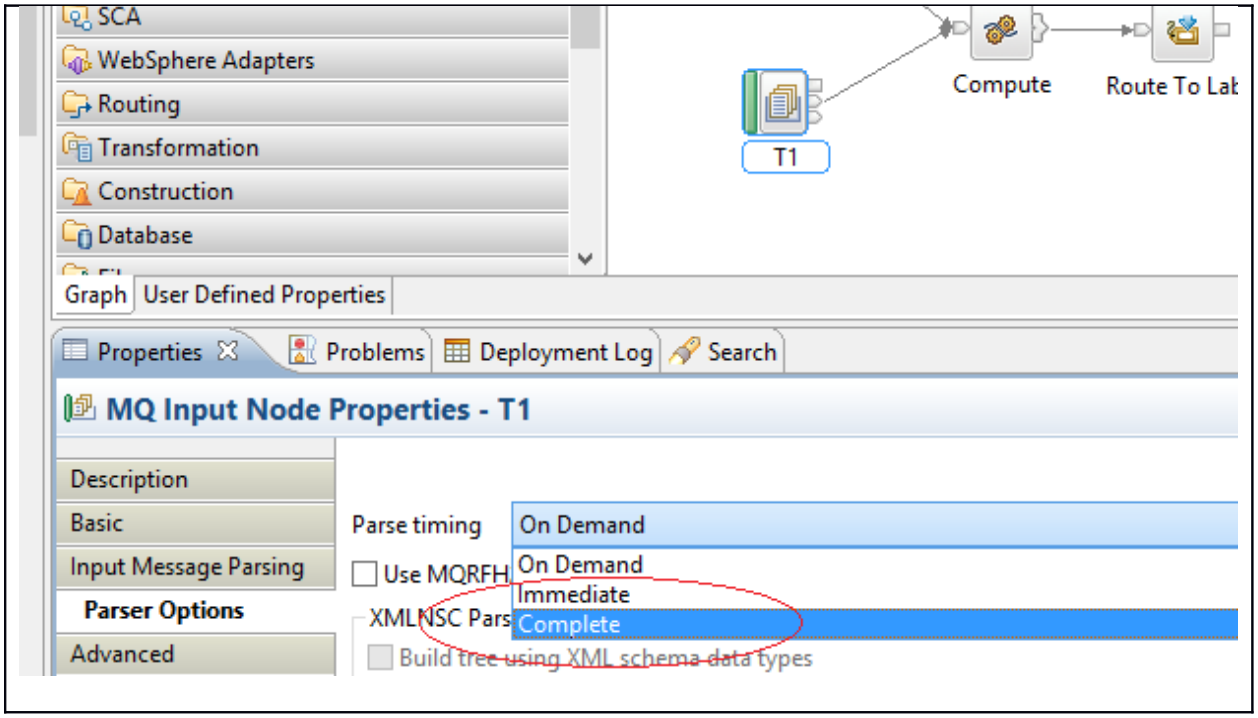
Rule: InputNode parse timing is not set to 'complete'

Sonar Rule: R67

Rational:

Enabling 'Complete' input parsing allows the whole message to processed/validated at the start, so failure can happen as early as possible.

Example:



Preferred:

References:
NA

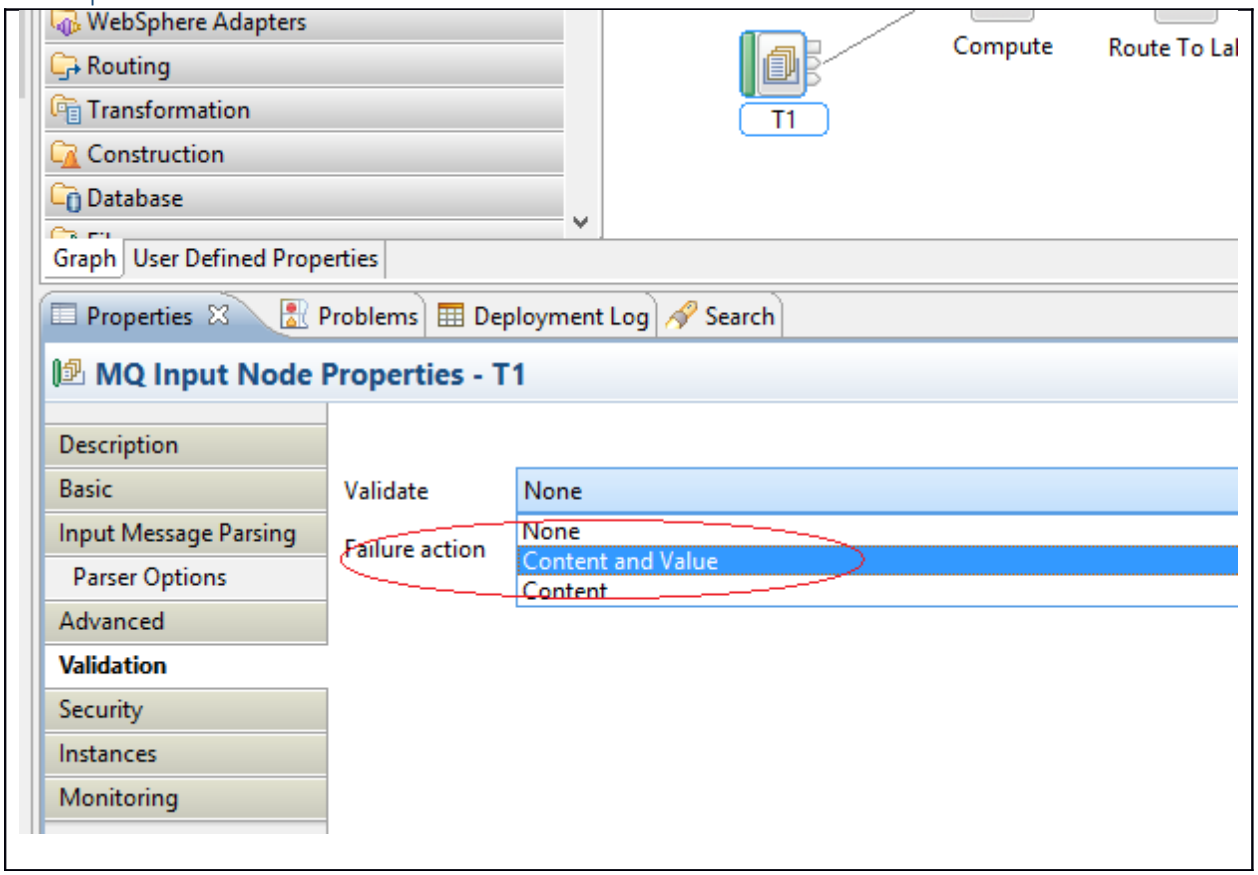
Rule: InputNode validation is not set to 'content and value'

Sonar Rule: R68

Rational:

Enabling 'Content and Value' validation allows the whole message to processed/validated at the start, so failure can happen as early as possible.

Example:



Preferred:

NA

References:

NA

Rule: The filter node may only have one return value

Sonar Rule: R91

Rational:

Filter nodes that only have 1 return are not providing filtering to more then one available path.
A filter node with a single return could be either a logic error or could be redundant.

Example:

```
CREATE FILTER MODULE Flow2_Filter
	CREATE FUNCTION Main() RETURNS BOOLEAN
	BEGIN
		SET OutputRoot.XMLNSC.Response.details.person.age = '20';
		RETURN FALSE;
	END;
END MODULE;
```

Preferred:

Check that a filter node is required.

References:

NA

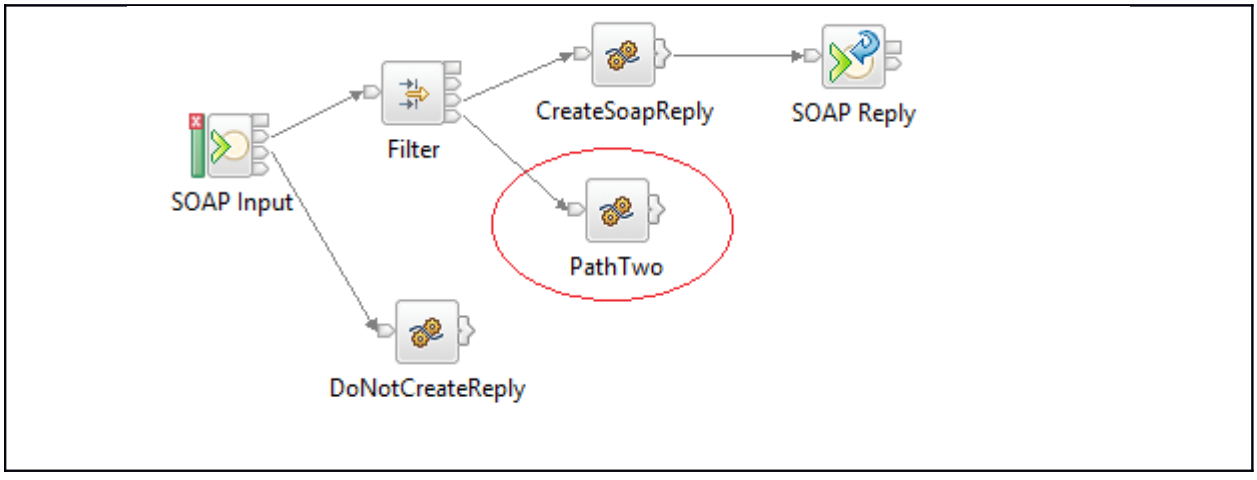
Rule: The message flow does not consistently reply to messages/requests

Sonar Rule: R65

Rational:

This rule checks that when using a request reply pattern, either with SOAP, HTTP or MQ, if one path through the code replies, then all paths through the code reply.
In the case of MQ, if a client is expecting a response and the flow doesn't always return one, then the consumer is blocking/waiting potentially indefinately. SOAP and HTTP responses will time out, but that also may cause issues for service consumers.

Example:



Preferred:

Check that each path, even the error paths have a valid reply.

References:

http://www.ibm.com/developerworks/websphere/library/techarticles/0910_phillips/0910_phillips.html

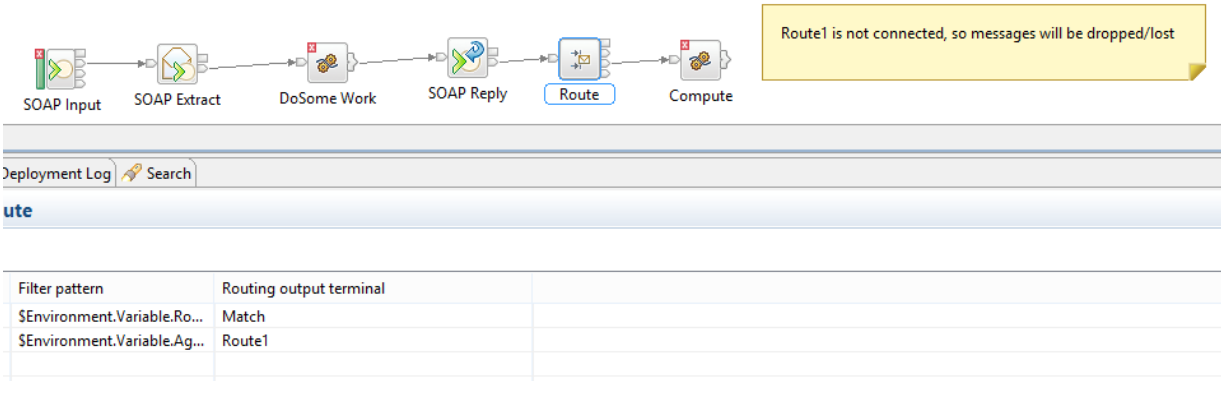
Rule: The routing nodes connections and filters may not be consistant

Sonar Rule: R62

Rational:

Routing nodes have a table of allowed exits (routes), if the routes and the connected outputs don't match, this could indicate a logical error.

Example:



Preferred:

Check the route table against the connections attached.

References:

NA

Rule: The queue name defined may not be compliant (length, case, underscores, starts with SYSTEM., blanks, short names)

Sonar Rule: R59

Rational:

Queue names can be created that function successfully in one environment configuration but fail or work differently in another. This check suggests when a queue name isn't following a consistent naming practice, or has a name that may be problematic in different environments (OS/version issues)

Example:

Queue names such as :

Queue Name	Reason
A	should be discouraged as they are short and meaningless
SYSTEM.xxx	could conflict with broker runtime queues

Preferred:

Work towards a consistent queue naming convention.

References:

NA

Rule: The code may be referring a to field that is not part of the MQMD header definition (and may be ignored)

Sonar Rule: R57

Rational:

When ESQL interacts with a message, there is no tyype safe checking that you would get with a struct in C or an Object in Java.
This check indicates that a field is being accessed that doesn't exist and will be ignored.

Example:

```
SET OutputRoot.MQMD.StrucIdx = 'abc';
```

Preferred:

Check against valid/allowed MQMD header fields.

```
SET OutputRoot.MQMD.StrucId = 'abc';
```

References:

NA

Rule: The LOOP may not have a valid LEAVE statement (and may not exit validly)

Sonar Rule: R56

Rational:

An infinite loop within ESQL code will cause the execution group (EG) to stop responding/hang.
This violation indicates that an infinite loop can occur.

Example:

```
DETAILS_LOOP : LOOP
        SET details = Environment.Variables.Details;
```

```
        IF COALESCE(details,'') = '' THEN
                -- use in testing
        END IF;
END LOOP DETAILS_LOOP;
```

Preferred:

Check the exit conditions:

```
DETAILS_LOOP : LOOP
        SET details = Environment.Variables.Details;
        IF COALESCE(details,'') = '' THEN
                -- use in testing
                LEAVE DETAILS_LOOP;
        END IF;
END LOOP DETAILS_LOOP;
```

References:

NA

Rule: The compute nodes connections are inconsistent

Sonar Rule: R55

Rational:

There is a logical coupling of a compute node and its connections to the ESQL code that is executed when it runs. This violation indicates that there is an inconsistent state between the ESQL logic and compute node configuration.

Example:

In this case, the PROPOGATE from the node logic is inconsistant with the conections.
Messages would be "lost" or dissapear, which should probably be handled with a return "FALSE" then a PROPGATE.

```
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
	CALL CopyMessageHeaders();
	CALL CopyEntireMessage();

	IF (Environment.Variables.Person.Sex = 'Male') THEN
		PROPAGATE TO TERMINAL 'out1';
	ELSE
		PROPAGATE TO TERMINAL 'out2' DELETE NONE;
	END IF;
	DECLARE details CHARACTER;
	RETURN TRUE;
END;
```

Preferred:

Check that the paths through the node match the ESQL PROPOGATE statements.

References:

NA

Rule: The date format may not be correct

Sonar Rule: R54

Rational:

The plugin attempts to scan the date formatting used in the ESQL and determine whether the format is valid.

Example:

```
RETURN CAST(dateAsChar AS DATE FORMAT 'dd/MMMMM/yy');
```

Preferred:

Check the date format is valid.

References:

NA

Rule: The filter node may not have its connections connected correctly

Sonar Rule: R52

Rational:

There is a logical coupling of a filter node and its connections to the ESQL code that is executed when it runs. This violation indicates that there is an inconsistent state between the ESQL logic and filter node.

Example:

```
CREATE FILTER MODULE Flow2_Filter
	CREATE FUNCTION Main() RETURNS BOOLEAN
	BEGIN
		RETURN TRUE;
	END;
END MODULE;
```

The ESQL logic above for the filter node never returns unknown or true, so doesn't make logical sense.

Preferred:

Check that filter node and the ESQL are consistent.

References:

NA

Rule: The filter node cannot modify the message

Sonar Rule: R53

Rational:

The message tree is immutable when a filter node runs. Any attempts to write to the message tree will be ignored and are treated as a logic error.

Example:

```
CREATE FILTER MODULE Flow2_Filter
	CREATE FUNCTION Main() RETURNS BOOLEAN
	BEGIN
		SET OutputRoot.XMLNSC.Response.details.person.age = '20';
		RETURN FALSE;
```

```
        END;
END MODULE;
```

Preferred:

Check that filter node and the ESQL are consistent.

References:

NA

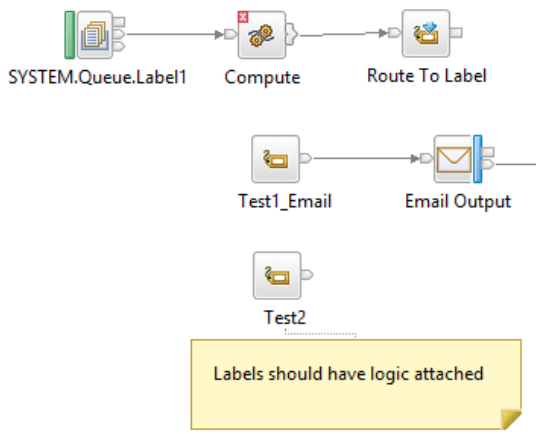
Rule: Label has no associated processing logic attached

Sonar Rule: R50

Rational:

When Message Broker jumps to an empty label, the processing stops at a dead end. This could indicate a logic issue. For example, if this is done as part of a SOAP Request/Reply pattern, then the flow will never return a response.

Example:



Preferred:

Check that labels with no processing attached are logically valid.

References:

NA

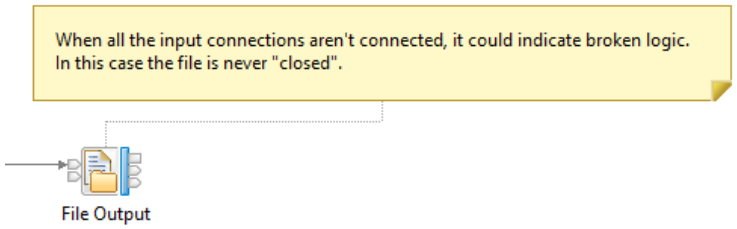
Rule: Not all input nodes connected. Resources may not be processed correctly

Sonar Rule: R51

Rational:

Some nodes require multiple inputs to function correctly. These include the FileOutput node (where the using flow needs to close the file), the Aggregation node that requires a message to indicate that aggregation can be completed.
When these terminals aren't connected, it could be an indication of a logic error.

Example:



In this case, the file is never closed.

Preferred:

Check that flows in question are configured according to how they are being used.

References:

NA

Rule: TODO has been left in the comments

Sonar Rule: R43

Rational:

“TODO” that has been left in the code could either be an issue around code maturity, or could indicate business logic or an algorithim that hasn't been completed.

Example:

```
// TODO complete the last name check with the special case of 66 year olds
IF (Env.Person.LastName IS NULL) THEN
        IF (Env.Person.FirstName IS NOT NULL) THEN
                SET Env.Message.Out = 'Wow, you have done well';
                SET Env.Message.NextValue = '10';
        ELSEIF (Env.Person.Age > 99) THEN
                SET Env.Message.Out = 'Wow, you are almost there';
                SET Env.Message.NextValue = '9';
        END IF;
END IF;
```

Preferred:

Check the logic and either complete the logic or remove the comment.

References:

http://checkstyle.sourceforge.net/config_misc.html#TodoComment

Rule: Code is unreachable following a RETURN or THROW statement

Sonar Rule: R40

Rational:

Code that follows a RETURN or THROW cannot be running. This can either suggest dead code or a logic error.

Example:

The login in the line highlighted below will not be executed.

```
IF (Env.Person.LastName IS NULL) THEN
        IF (Env.Person.FirstName IS NOT NULL) THEN
                SET Env.Message.Out = 'Wow, you have done well';
                SET Env.Message.NextValue = '10';
        ELSEIF (Env.Person.Age > 99) THEN
                SET Env.Message.Out = 'Wow, you are almost there';
                SET Env.Message.NextValue = '9';
        END IF;
END IF;
RETURN;
SET Env.Message.Flag.Ignored = 'TRUE';
```

Preferred:

Check the logic and either move or remove the affected code.

References:

NA

Rule: The PASSTHRU statement parameters and values don't match

Sonar Rule: R33

Rational:

The plugin matches the parameters to the structure passed into the PASSTHRU function. This allows the plugin to make sure that the SQL code is consistent and with its parameters.

Example:

```
SET LocalEnvironment.Variables.SelectData[] = PASSTHRU('SELECT * ' ||
                                        'FROM THEDATA.PersonTable WHERE Age = ?');
```

Preferred:

Analyse the SQL and the parameters to make sure that they are consistent.

References:

NA

Rule: Node refers to an empty main method. Either code has been left out or the node can be removed from the flow

Sonar Rule: R30

Rational:

When a compute node is added to a message flow, it is created with a matching ESQL procedure. This violation indicates that a node has been added but no ESQL code has been attached. This node is essentially not performing any function and can be removed. It could also indicate that the node should have logic added, which has been missed by the developer.

Example:

```
CREATE COMPUTE MODULE Flow5_Compute
	CREATE FUNCTION Main() RETURNS BOOLEAN
	BEGIN
		RETURN TRUE;
	END;

	CREATE PROCEDURE CopyMessageHeaders() BEGIN
		DECLARE I INTEGER 1;
		DECLARE J INTEGER;
		SET J = CARDINALITY(InputRoot.*[]);
		WHILE I < J DO
			SET OutputRoot.*[I] = InputRoot.*[I];
			SET I = I + 1;
		END WHILE;
	END;

	CREATE PROCEDURE CopyEntireMessage() BEGIN
		SET OutputRoot = InputRoot;
	END;
END MODULE;
```

Preferred:

Either add code the ESQL procedure or delete the node.

References:

NA

Rule: The input node has no failure handler connected. Errors may not be able to be tracked or may be lost

Sonar Rule: R48

And

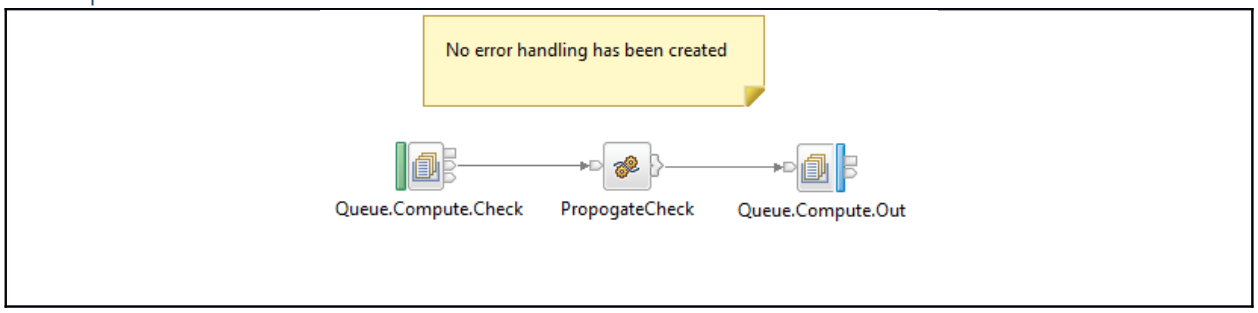
Rule: The input node has no catch handler connected. Errors may not be able to be tracked or may be lost

Sonar Rule: R71

Rational:

If the first node in a flow has no failure handler, then errors may be lost depending upon how the input node is configured. Many MB users have default Error/Failure flows that they make use of.

Example:



Preferred:

Check that possible error conditions are taken into account in the design and are handled appropriately.

References:

http://blogs.msdn.com/b/dotnet/archive/2009/02/19/why-catch-exception-empty-catch-is-bad.aspx

http://www.ibm.com/developerworks/library/j-jtp06294/

Rule: Try/Catch no functional catch connected. May cause errors to be lost

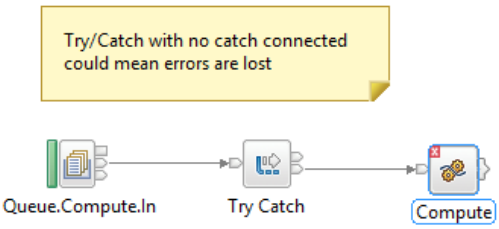
Sonar Rule: R25

Rational:

Exception handlers prevent the default error handling for flows. For flows that consume messages, the default error handling to send the message to the appropriate dead letter queue, for flows consuming SOAP messages, the default error handling to return a fault.

Try/catch handlers with no catch result in the error being silently swallowed, which in most cases can cause the loss of information, such as the contents of a business message.

Example:



Preferred:

Look to add an exception handler or remove the Try/Catch node.

References:

http://blogs.msdn.com/b/dotnet/archive/2009/02/19/why-catch-exception-empty-catch-is-bad.aspx

http://www.ibm.com/developerworks/library/j-jtp06294/

Rule: Atomic references atomic

Sonar Rule: R17

Rational:

ATOMIC sections of code mark critical sections that only one thread can enter at a time.
Nested calls to ATOMIC blocks of code could cause a dead lock where 2 independent threads are have created a dead lock situation.

Example:

```
ATOMICROUTING : BEGIN ATOMIC -- beginning of atomic block
  CALL AtomicProcedure();
END ATOMICROUTING ; -- end of the atomic block

CREATE PROCEDURE AtomicProcedure() BEGIN
        EMBEDDEDATOMIC : BEGIN ATOMIC -- beginning of atomic block. Processing is single
threaded until the END; is reached
                SET OutputRoot.XMLNSC.SoapMessage.SoapBody.Person.Name = 'Fred';
        END EMBEDDEDATOMIC ; -- end of the ROUTING atomic block
END;
```

Preferred:

Analyse and refactor code to prevent possible dead lock situations.

References:

NA

General coding best practices

Rule: The condition is more complicated then the threshold

Sonar Rule: R92

Rational:

Complicated IF and WHILE conditions are difficult to read and understand.
The current threshold is 3.

Example:

```
IF (Person.Age = 20) AND (Person.Sex = 'M') AND (Person.Heighht > 1029) OR (Person.Weight = 50)
THEN
```

Preferred:

Refactor conditions to be more readable.

References:

NA

Rule: The function or procedure is longer than the threshold

Sonar Rule: R29

Rational:

Long methods are generally harder to understand and maintain then shorted methods.
The threshold for when a violation occurs is controlled by the property “sonar.mb.esql.functionsize” in the sonar.properties file. The default value is “50”.

Example:

NA

Preferred:

Look at whether some of the code can be refactored into logical blocks. Analyse in conjuction with the CPD (copy paste detection) to see if some of the code can be provided by a common procedures.

References:

http://sourcemaking.com/refactoring/long-method

Rule: Cyclomatic Complexity is higher then the threshold

Sonar Rule: R28

Rational:

Cyclometric complexity is one measure of how complex a procedure/function is.

The threshold for when a violation occurs is controlled by the property “sonar.mb.esql.complexity” in the sonar.properties file. The default value is “10”.

Example:

```
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
	CALL CopyMessageHeaders();
	-- CALL CopyEntireMessage();
	SET Environment.Variables.P1.Name = OutputRoot.XMLNSC.Request.Person.Name;
	SET Environment.Variables.P1.Age = OutputRoot.XMLNSC.Request.Person.Age;
	SET Environment.Variables.P1.PostCode = OutputRoot.XMLNSC.Request.Person.PostCode;
	SET Environment.Variables.P1.FirstName = OutputRoot.XMLNSC.Request.Person.FirstName;
	SET Environment.Variables.P1.LastName = OutputRoot.XMLNSC.Request.Person.LastName;

	IF (Environment.Variables.P1.Age <> 20) then
		SET Environment.Variables.P1.StudentCard = 'TRUE';
```

```
	ELSE
		IF (Environment.Variables.P1.Age = 65) THEN
			SET Environment.Variables.P1.PenionCard = 'FALSE';
		END IF;

		IF (LENGTH(OutputRoot.XMLNSC.Request.Person.PostCode) > 4) THEN
			IF (NOT CONTAINS(OutputRoot.XMLNSC.Request.Person.PostCode,'2612')
AND NOT ENDSWITH(OutputRoot.XMLNSC.Request.Person.PostCode,'2613')) THEN
				SET OutputRoot.Test6.Local =
LEFT( OutputRoot.XMLNSC.Request.Person.PostCode, 4 );
			ELSE
				SET Environment.Variables.P1.PenionCard = 'UNKNOWN';
			END IF;
		END IF;
	ELSE

		SET Environment.Variables.P1.PenionCard = 'UNKNOWN';
	END IF;
	CALL ComplicatedFunction(Environment.Variables.SomeValue);

	CALL Test_Duplications();
	SET LocalEnvironment.Variables.SelectData[] = PASSTHRU('SELECT * ' || 'FROM
THEDATA.PersonTable');
	RETURN TRUE;
END;
```

Preferred:

Procedures and functions should be simplified where possible.

References:

http://en.wikipedia.org/wiki/Cyclomatic_complexity

http://blogs.msdn.com/b/zainnab/archive/2011/05/17/code-metrics-cyclomatic-complexity.aspx

http://weblambdazero.blogspot.com.au/2013/08/cyclomatic-complexity-why-long-methods.html

Rule: Unused variable

Sonar Rule: R5

Rational:

Unused variables add complexity to your code and with no value. They waste developers time understanding code which is essentially dead code. They can also indicate a failure in an algorithim.

Example:

NA

Preferred:

Comment out unused variables and check that the code can still be build. Then delete them from the code so that there is less useless commenting.

References:

NA

Rule: Unused method

Sonar Rule: R16

Rational:

Unused methods add complexity to your code and with no value. They waste developers time understanding code which is essentially dead code. They can also indicate a failure in an algorithim.

Example:

NA

Preferred:

Comment out unused methods and check that the code can still be built and executed. Then delete the code so that there is less useless commenting.

References:

NA

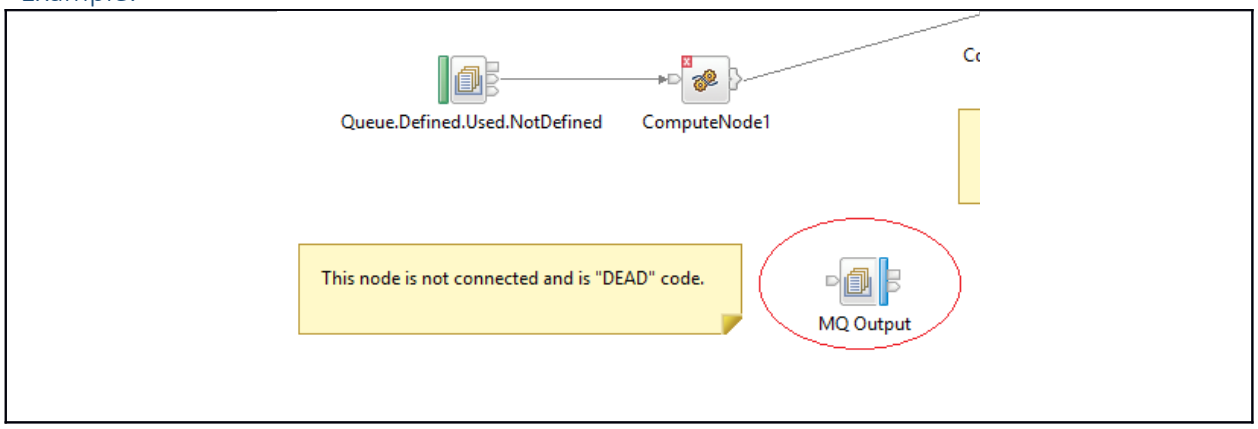
Rule: There is no input connection to this node. The code may not be reachable or functioning

Sonar Rule: R49

Rational:

A flow may not be connected, and hence may not be invoked by the logic. This could mean that the whole flow is either incorrectly configured, which could cause logic errors, or is not required and is dead code, and can be removed.

Example:



Preferred:
Delete any flows that are not required.

References:
NA

Rule: A subflow has been created but is not being referenced. It may be able to be removed

Sonar Rule: R36

Rational:
Subflows are the logical equivalent of common procedures but for msgflow's. Subflows that are not referenced and are not required can be deleted to reduce confusion and improve clarity of the code.

Example:
NA

Preferred:
Delete the subflow that is not required.

References:
NA

Rule: Source file is empty

Sonar Rule: R24

Rational:
Empty files have no function and should be cleaned up.

Example:

NA

Preferred:

Delete empty files.

References:

NA

Other

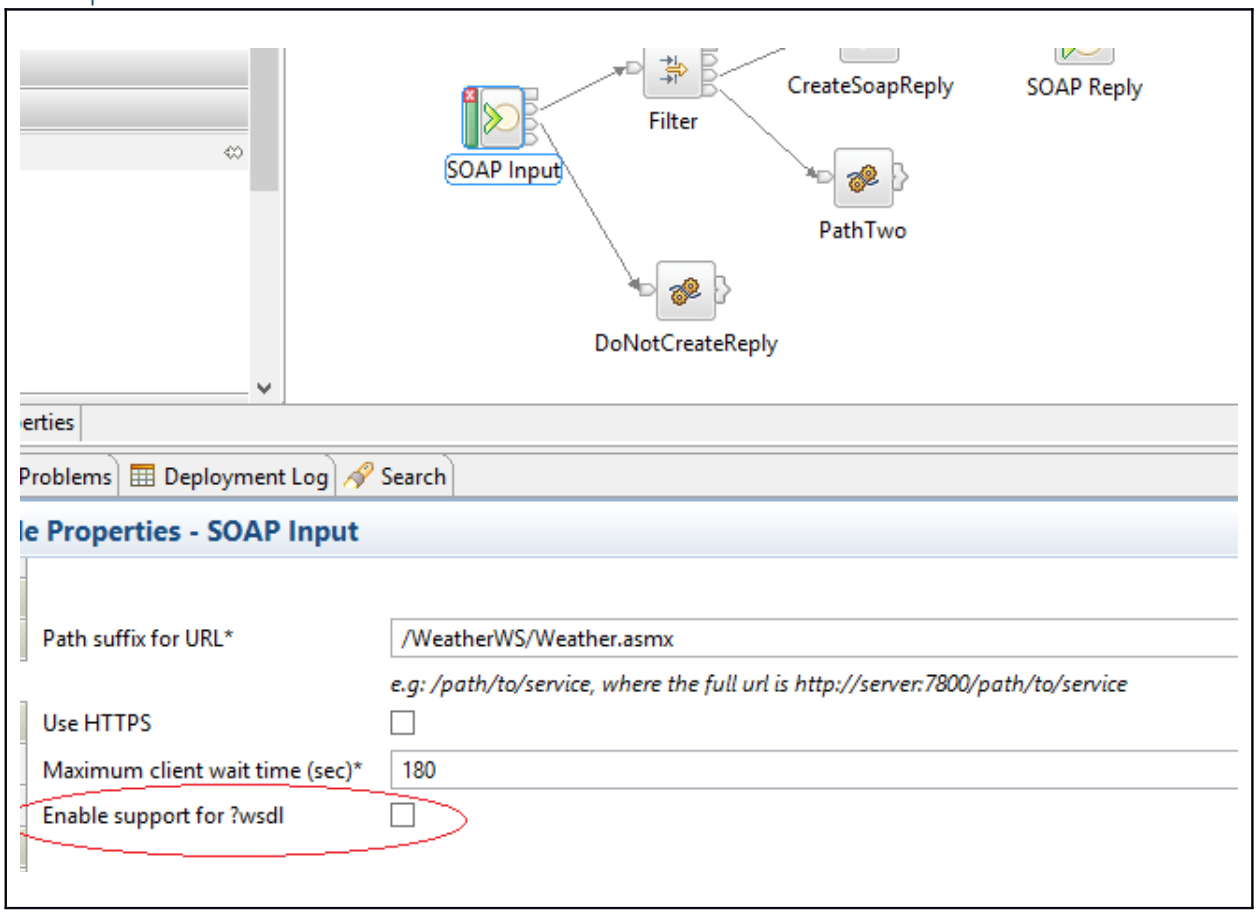
Rule: SOAPInputNode does not have 'Enable support for ?wsdl checked'

Sonar Rule: R66

Rational:

Enabling 'wsdl' support allows developers to extract information from deployed services to assist them in developing consuming services. For production systems that face the general public this may need to be turned off.

Example:



Preferred:

NA

References:

NA

Rule: The message flow may not have been included in the deployment build scripts

Sonar Rule: R63

Rational:

The plugin matches message flows against ant deployment/build scripts when available. If a flow is not in a deployment script, this could indicate that the script is incomplete or that the flow is dead/unused code and can be deleted.

Example:

Extract from build.xml

```
<exec executable="${create.bar.home}/mqsicreatebar" spawn="false">
        <arg value="-cleanBuild" />
        <arg value="-skipWSErrorCheck" />
        <arg value="-data" />
        <arg value="${env.WORKSPACE}" />
        <arg value="-b" />
        <arg value="${bar.name}" />
        <arg value="-p" />
        <arg value="Flow1a" />
        <arg value="TestServices" />
        <arg value="TestSchema" />
        <arg value="TestMessageSet" />
        <arg value="UtilityServices" />
        <arg value="-o" />
        <arg value="Flow1a.msgflow" />
        <arg value="TestProjectMessageSet/TestMessageSet/messageSet.mset" />
</exec>
```

Preferred:

Check the deployment scripts and flows match.

References:

NA

Rule: Credentials are in plain text

Sonar Rule: R18

Rational:

This indicates that credentials for authorisation/authentication have been setup in the code in plain text.

Example:

```
SET OutputRoot.Properties.IdentityMappedType = 'usernameAndPassword';
SET OutputRoot.Properties.IdentityMappedToken = 'xxx';
SET OutputRoot.Properties.IdentityMappedPassword = 'plaintextpassword';
```

```
SET OutputRoot.Properties.IdentityMappedIssuedBy = 'zz';
```

Preferred:

They ideally should be replaced by a constant defined as an ‘EXTERNAL’ so that they can be replaced at deployment time using a broker override.

References:

NA

Documentation

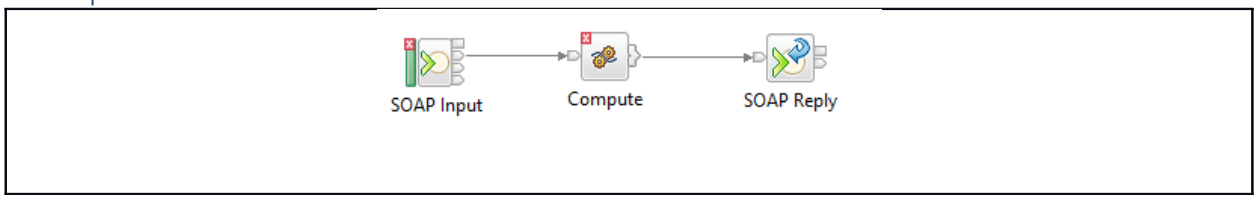
Rule: Message flow does not contain a note

Sonar Rule: R31

Rational:

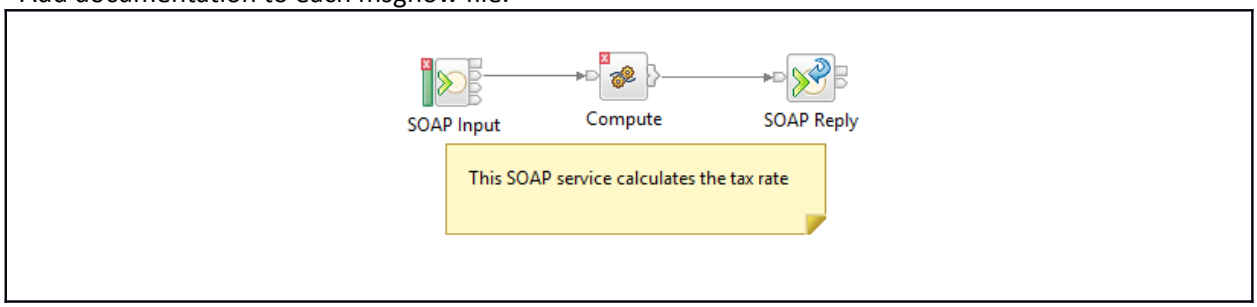
One option for documenting flows is to add a note to the msgflow. This rule when enabled indicates which flows don't have a documentation note.

Example:



Preferred:

Add documentation to each msgflow file.



References:

http://code.mycila.com/license-maven-plugin/

http://java.dzone.com/announcements/maven-2-license-plugin

Rule: File does not contain header comments

Sonar Rule: R20

Rational:

The plugin can check that a “standard” header has been added to the top of each ESQL file. This may be for ensuring SVN placeholders are in each file, a standard copyright message or any other standard documentation that might be useful. It mimics the Maven Licence plugin
It can be configured by setting the property *sonar.mb.header.company* in the sonar.properties file.

Example:

```
sonar.mb.header.company=Copyright Abc.co, 1997
```

Preferred:

NA

References:

http://code.mycila.com/license-maven-plugin/

http://java.dzone.com/announcements/maven-2-license-plugin

Rule: File does not contain header comments

Sonar Rule: R20

Rational:

The plugin can check that a “standard” header has been added to the top of each ESQL file. This may be for ensuring SVN placeholders are in each file, a standard copyright message or any other standard documentation that might be useful. It mimics the Maven Licence plugin
It can be configured by setting the property *sonar.mb.header.company* in the sonar.properties file.

Example:

```
sonar.mb.header.company=Copyright Abc.co, 1997
```

Preferred:

NA

References:

http://code.mycila.com/license-maven-plugin/

http://java.dzone.com/announcements/maven-2-license-plugin

Rule: Header files should contain author, version and date

Sonar Rule: R121

Rational:

Having a standard place holder for author, version and date allows for improved documentation and better integration with tools such as SVN.

Example:

```
/*
        Author: Test Coder
        Version: 1.0.1
        Date: 1/12/2015
*/
```

Preferred:

NA

References:

NA

Installation guide

The MB-Precise plugin runs on SonarQube (Sonar). There are a number of prerequsites to installing the plugin.

1. Java

Download and install a Java runtime and configure the appropriate JAVA_HOME environment variable so that java can run.

At the command prompt, type in

```
java –version
```

the command prompt output should be like the following:

```
java version "1.8.0_11"
Java(TM) SE Runtime Environment (build 1.8.0_11-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.11-b03, mixed mode)
```

2. SonarQube

Download and install SonarQube

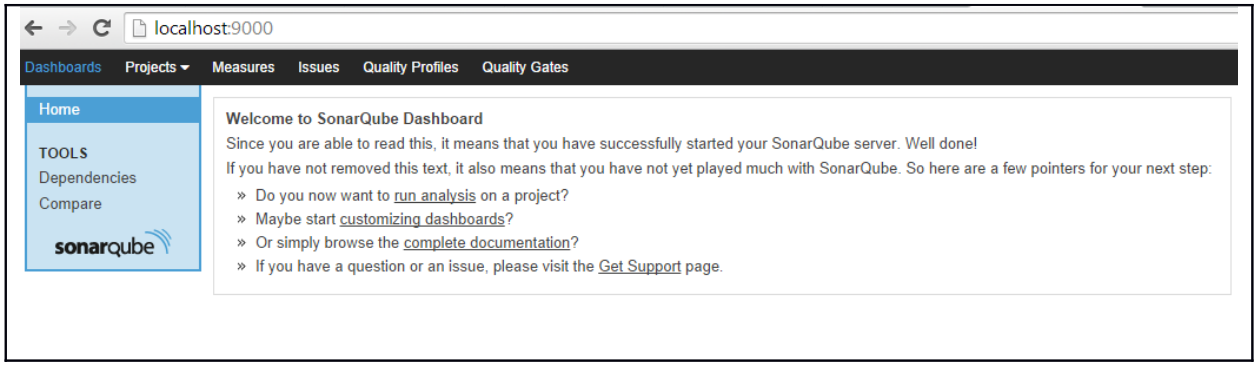
http://www.sonarqube.org/downloads/

Currently the MB-Precise plugin works on Sonar version 4.3.1+.

Start the server. Opening a browser, navigate to

http:\\localhost:9000

A page like the following should be displayed in Sonar is functioning correctly:



3. Sonar runner

Download and install sonar runner.

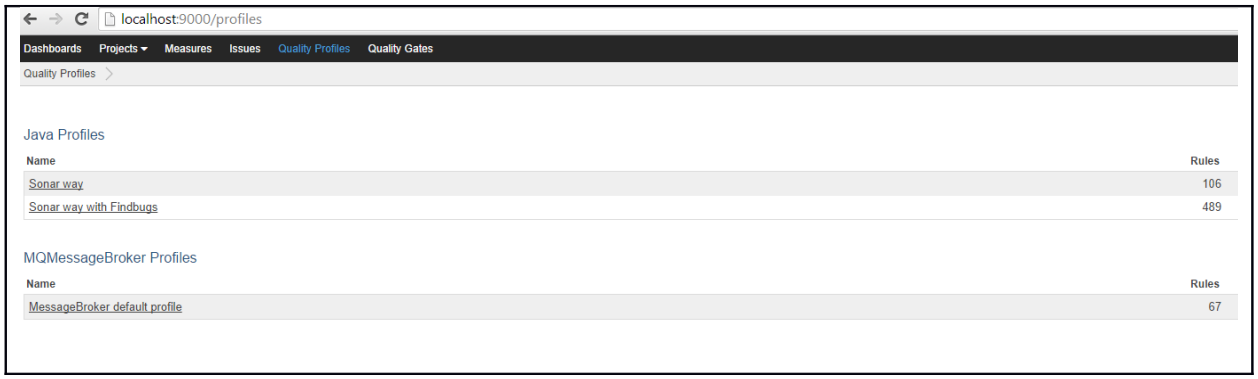
SonarRunner launches Sonar analysis on the client side.

http://www.sonarqube.org/downloads/

4. Install plugin

Copy the Mb_precise-x.jar into the ‘extensions\plugins’ folder on the Sonarqube server and restart the server.

Navigate to "profiles". The MB-Precise default profile should appear:



5. Configure sonar.properties

There are a number of additional properties that have been added that are used by the plugin:

This is threshold for how long the MQGet and other nodes with timeouts should wait. This check tries to prevent long processes that can block the Execution Group.

```
sonar.mb.flow.timeout.seconds=15
```

This property tells the plugin where to generate diagrams and documentation to. It is an absolute path. The following would send all diagrams to the “C:\test\demos\generated” directory.

```
sonar.mb.flow.diagram.output=C:\\test\\demos\\generated\\
```

This optional property tells the plugin what the common code heading block for ESQL should contain. This could be the company name and some copyright details. Is similar to what the maven plugin does:

http://mojo.codehaus.org/license-maven-plugin/check-file-header-mojo.html

```
sonar.mb.header.company=Richards Test Company
```

This is threshold for when a particular ESQL file is flagged as being complex. Its a numerical whole value and can be tuned as required.

```
sonar.mb.esql.complexity=16
```

This is threshold for when a particular ESQL function is flagged as being too long. Its a numerical whole value and can be tuned as required.

```
sonar.mb.esql.functionsize=32
```

This is threshold for when a particular line of ESQL is too long. Usually it has to able to be read without excessive scrolling.
Its a numerical whole value and can be tuned as required.

```
sonar.mb.esql.maxlinesize=99
sonar.mb.jdbc.*
```

This allows the plugin to connect to an active SQL/Relational datasource and validate that the tables and fields referenced in the code exist in the database. It also checks that queries are accessing tables against valid indexes.

```
sonar.mb.jdbc.driver=org.postgresql.Driver
sonar.mb.jdbc.url=http://localhost:9000/DemoDB
sonar.mb.jdbc.user=sa
sonar.mb.jdbc.password=password123
```

There are additional resources available:

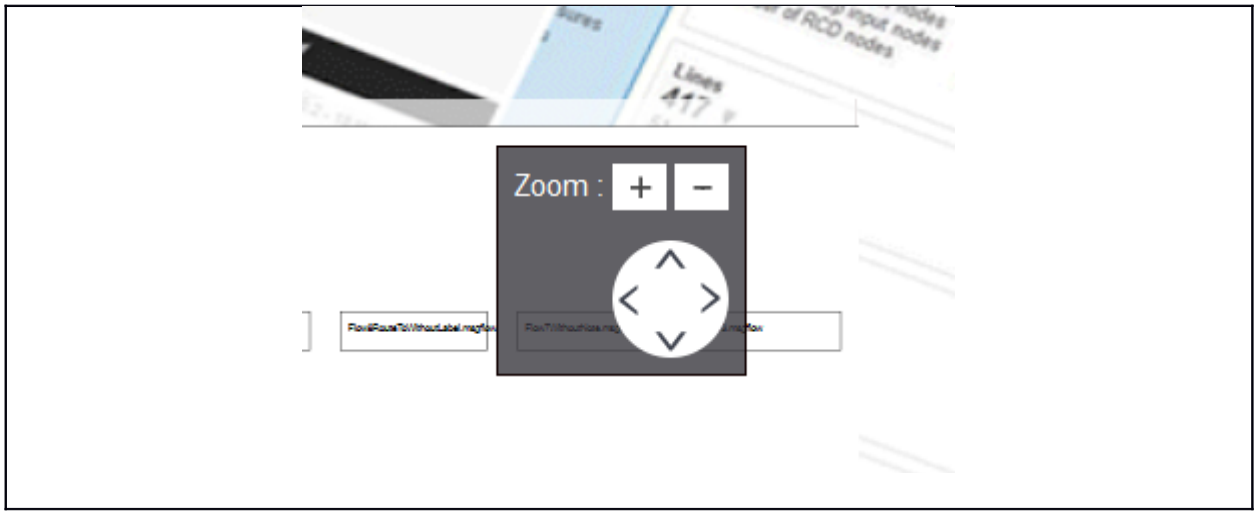
http://docs.codehaus.org/display/SONAR/Installing
http://docs.codehaus.org/display/SONAR/Analyzing+with+SonarQube+Runner
http://docs.codehaus.org/display/SONAR/Requirements#

Document Generation

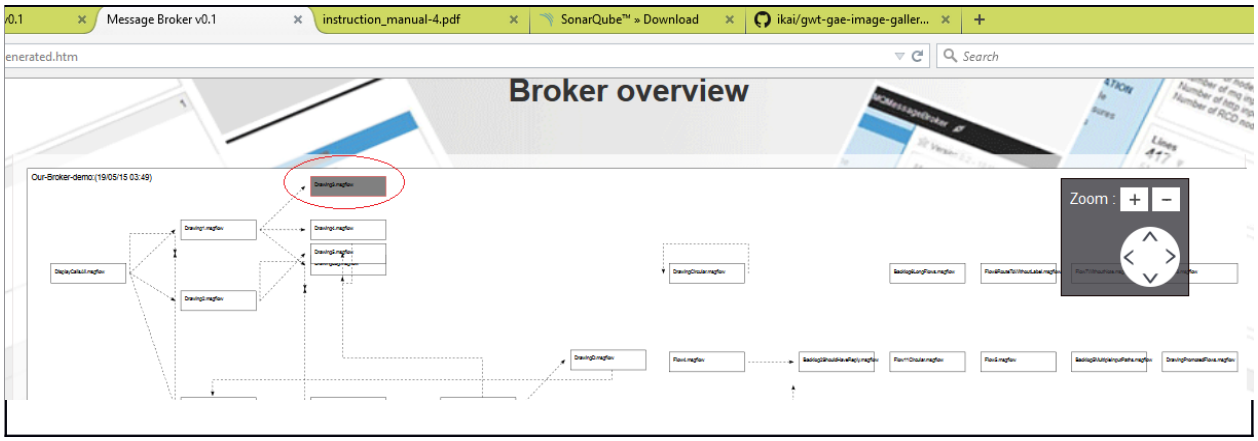
The plugin generates an overall diagram of the system from a flow perspective. It also loks to document each flows input/outputs and dependencies.
The document and diagram generation is controlled by the path set in the sonar.properties file

```
sonar.mb.flow.diagram.output=C:\\test\\demos5\\generated\\
```

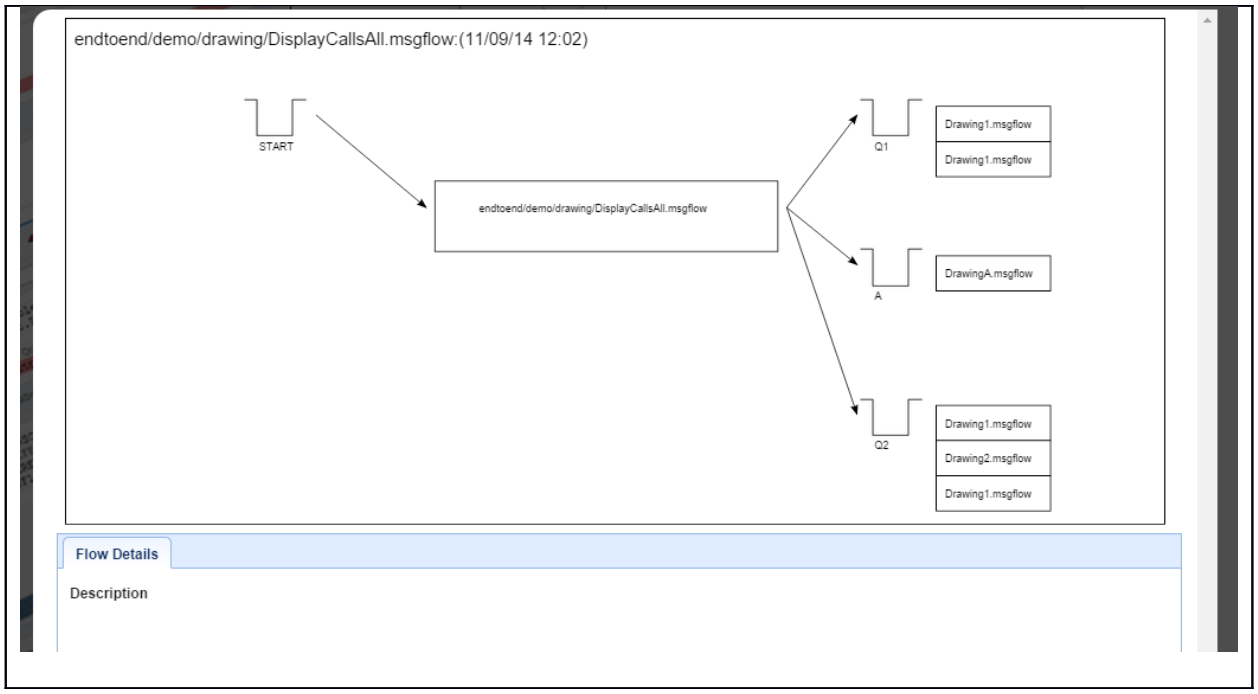
This is the directory in which the diagram will be generated into. This could be a local directory, an NFS folder, webDav or some other form of sharing/publication.
The diagram can be navigated using the arrows that provide panning and zoom functionality.



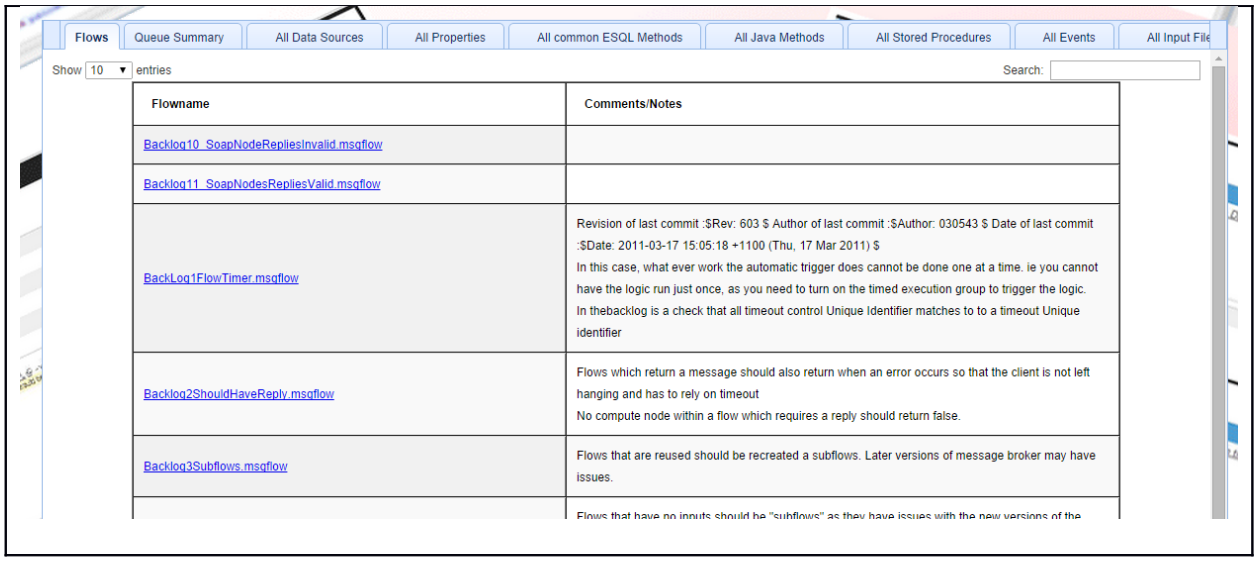
Clicking on a flow (a highlighted red rectangle) allows you to view the "flow" level.



Should open:



Along the bottom of the main diagram are tabs.
The tabs are a collection of information from the the project.



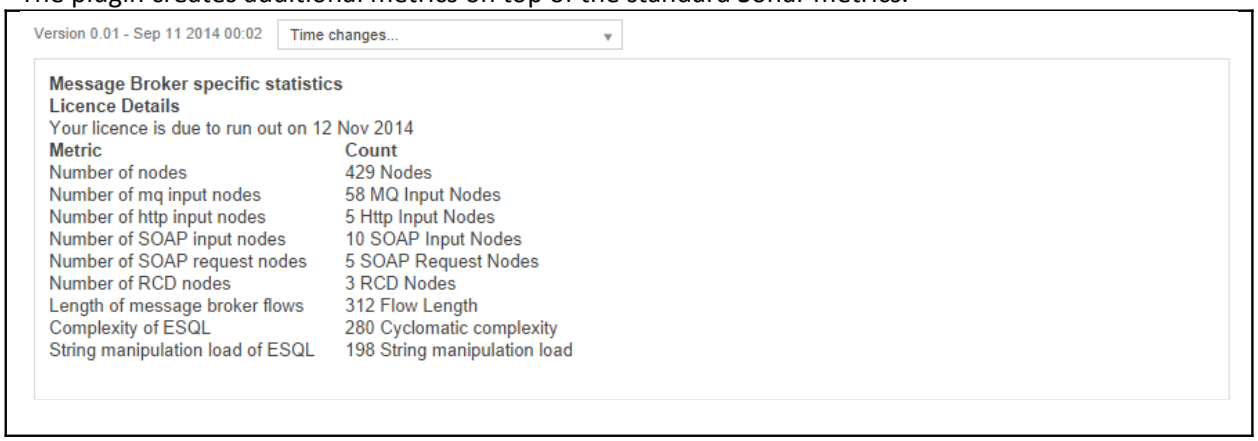
The tabs are:

Tab	**Description**
Flows	a list of all flows and their notes/comments.

Queue Summary	a list of all queues and the flow they are used in.
All Data Sources	a list of all data sources used and in which flows.
All Properties	a list of all UDP's the flows they are used in.
All common ESQL methods	A list of all ESQL procedures/functions not associated with a Filter/Compute node. i.e. all "re-usable" procedures/functions.
All java methods	All java methods and the ESQL module they belong to.
All stored procedures	All stored procedures and the ESQL they belong to.
All events	All monitoring events created.
All input files	All file names used from flows
All output files	All file names used from flows

Metrics

The plugin creates additional metrics on top of the standard Sonar metrics.



These can be selected and sorted by file.

Metric	Description
Number of nodes	all nodes, which tells you which flows are the most complicated from the point of view of pure size.
Number of http input nodes	
Number of SOAP input nodes	
Number of SOAP request nodes	
Number of RCD nodes	which is useful for looking at performance
Length of message broker flows	which is useful for looking at performance
Complexity of ESQL	
String manipulation load of ESQL	which measures how many string manipulation functions a peice of ESQL calls. Which is useful for looking at performance.