Focused Run for SAP Solution Manager 2.0
Starting Feature Pack 00
## Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Textual cross-references to other documents.</td>
</tr>
<tr>
<td><strong>EXAMPLE</strong></td>
<td>Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, <code>SELECT</code> and <code>INCLUDE</code>.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><code>&lt;Example&gt;</code></td>
<td>Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><strong>EXAMPLE</strong></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE</strong></td>
<td>Keys on the keyboard, for example, <code>F2</code> or <code>ENTER</code>.</td>
</tr>
</tbody>
</table>
Caution

Before you start the implementation, make sure that you have the latest version of this document that is available at https://help.sap.com/viewer/p/FOCUSED RUN.

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2019-06-26</td>
<td>Initial version</td>
</tr>
</tbody>
</table>

Document History
# Table of Contents

## Contents

1. Introduction .............................................................................................................. 5
2. Process of Model Creation ....................................................................................... 6
   2.1 Data Collection...................................................................................................... 6
   2.2 Analysis of Metrics and Feature Selection .......................................................... 6
   2.3 Data Preparation and Training a Model ............................................................... 6
   2.4 Deployment of Models in Runtime ...................................................................... 7
3. Creating Custom Models ......................................................................................... 8
   2.1 Create Model Definition File ............................................................................... 8
   2.2 Create BAdI Implementation ............................................................................... 9
   2.3 Import Model Definition ...................................................................................... 12
   2.4 Check Configuration ......................................................................................... 12
1 Introduction

With regards to artificial intelligence, a model is the result of a training process that produces predictions. This means that a model can be trained with specific data from past occurrences to produce predictions on future behavior. For example, you can train a model by providing various metrics of different categories like performance, exception, and configuration. These metrics form the model’s parameters from which it can produce a prediction. When testing a model, the same parameters are provided. The model evaluates the new data based on the processing of data acquired from the training phase, and the model produces a prediction for the new data. The document is valid from Focused Run 2.0 SP00 of SAP Solution Manager.
2 Process of Model Creation

2.1 Data Collection

You collect data of all monitoring metrics for a specific timeframe from the system which experienced an unplanned downtime. For example, a HANA system went down on 25th December 00:00, you must collect data of all HANA metrics from 36 hours before the downtime occurred and 36 hours after the downtime occurred i.e. from 23rd December 12:00 to 26th December 12:00. The collection frequency of each metric is 5 minutes.

2.2 Analysis of Metrics and Feature Selection

You must analyze time series plot of each metric and select only those metrics which showed any interesting behavior before the downtime. For example, in Figure 1 Average Ping Time from HANA System you observe that it showed variation before the downtime (which occurred on 07th December 11:30). You collect downtime data from many systems and select a set of metrics which explain downtime after manually analyzing time series plots. You also remove metrics which show the same behavior as you want to keep your metric set for training small for efficiency purposes.

![Figure 1: Time series plot of Average Ping Time](image)

2.3 Data Preparation and Training a Model

You remove rows which are in downtime period and 15 minutes before the downtime. For example, a downtime which occurred on 25th December between 00:00 to 00:30, you must remove data from 24th December 23:45 to 25th December 00:30. Then you go back in time and see when your metric set showed interesting behavior leading up to the downtime, you label those rows of data as outage. You use the rest of data which we didn’t label as outage as non-outage (Figure 2). Therefore, we have a binary classification problem for which we used LSTM to build a model. There are few hyperparameters like sequence length, number of layers and neurons in each layer which we tune to get better results. The reason you are using LSTM is, it is a multivariate time series classification method which looks back few minutes of data and then decides whether this pattern resembles an outage sequence or not as opposed to row based classification which considers only one measurement and then classifies as an outage or a non-outage.
2.4 Deployment of Models in Runtime

You need to use tensorflow library to train models which are deployed in a R server. At runtime, you have to collect data of the metrics which you selected during the training and send it to R server where the models are loaded, and predictions happen.
3 Creating Custom Models

Steps for creating Custom Models in System Anomaly Prediction of Focused Run for SAP Solution Manager 2.0 (FRUN 2.0)

1. Create the model definition file
2. Create BAdI Implementation
3. Import Model Definition
4. Check Configuration
5. Check Monitoring

3.1 Create Model Definition File

Create a model definition file in the following format.

```xml
<?xml version="1.0" encoding="utf-8"?>
<SHIPMENT>
  <SHIPMENT_VERSION>2</SHIPMENT_VERSION>
  <SHIPMENT_DESCRIPTION>This shipment contains OOM model and DB Hung Model for SAP HANA</SHIPMENT_DESCRIPTION>
  <SHIPMENT_DATE>20082018</SHIPMENT_DATE>
  <MODELS>
    <MODEL>
      <MODEL_HEADER>
        <MODEL_DESCRIPTION>Description of the model(model_text)</MODEL_DESCRIPTION>
        <MODEL_ID>Model_ID</MODEL_ID>
        <VERSION>Version</VERSION>
        <IS_SAP_MODEL></IS_SAP_MODEL>
        <CATEGORY>Category for grouping models</CATEGORY>
        <THRESHOLD>Numeric threshold</THRESHOLD>
      </MODEL_HEADER>
      <SYSTEMS>
        <SYSTEM>
          <TYPE>HANADB</TYPE>
        </SYSTEM>
      </SYSTEMS>
      <METRICS>
        <METRIC>
          <METRIC_NAME>TECHNICAL_NAME_OF_METRIC_1</METRIC_NAME>
          <METRIC_ID>TYPE_ID_OF_METRIC_1</METRIC_ID>
        </METRIC>
        <METRIC>
          <METRIC_NAME>TECHNICAL_NAME_OF_METRIC_2</METRIC_NAME>
          <METRIC_ID>TYPE_ID_OF_METRIC_2</METRIC_ID>
        </METRIC>
      </METRICS>
    </MODEL>
  </MODELS>
</SHIPMENT>
```
Important points to note:

1. You can upload multiple models at once using multiple `<MODEL>` tags. When you are uploading a new custom model, definition of all the existing custom models should also be present in this file. When you upload models, all the existing custom model definitions in the FRUN system are deleted and all definitions in this xml files are uploaded.

2. `MODEL_DESCRIPTION` is the name of model as should be visible in System Anomaly Configuration.

3. `MODEL_ID` should be the unique ID for your model. `MODEL_ID` should not start with string “SAP” as all the models which will be delivered by SAP will have ID starting with “SAP”.

4. `VERSION` is used for identifying version of a model. When uploading definition for a new version of existing model, separate `<MODEL>` tag should be used for new version along with the existing version definition.

5. All the metrics for a model can be listed using separate `<METRIC>` tag. This list of metrics will be checked while activating the custom model for a system. If one of the metrics is not active for a system, the model will not be activated for that system. This is the list of metrics that you will see in results UI metric list for a model. You can also use this list for aggregating data to apply your model. All the metrics will be present in PAS_SA_MODEL_MET along with the model ID.

6. `<SYSTEM>` tag is used to define the system type for which this model is applicable. The system type mentioned here will be used to filter the models in configuration and in monitoring UI. For example, if JAVA is mentioned here, then for systems of type JAVA, you would be able to see and configure this model.

7. Keep `<SHIPMENT>` tag as is. This tag should be present in XML. This tag is for tracking shipments from SAP, but it should be present even in case of custom models.

3.2 Create BAdI Implementation

1. RUN the transaction se80. Open package AI_PAS in the object navigation.
3. In the BAdI definitions column, right click “Implementations”. Click on “Create BAdI implementation. Press F8.
4. Give a name for Enhancement implementation

5. Press Enter. Continue and then save this implementation as Local object. Continue.

7. Give a name for the BAdI Implementation, enter a description, and name of the implementing class.
8. Continue and save as local object. Newly created BAdI Implementation would open.
9. Double click on “Filter Val.”.

10. Go to edit mode. Double click cell under Value 1.

11. Fill value 1 as <MODELID_VERSION>. For example, if model id is CUST_JAVA and version is 1. Then filter should be CUST_JAVA_1. And choose comparator 1 as “=”. Press Enter.
12. Double-click “Implementing Class”.

13. Double-click the method IF_SYSTEM_ANOMALY_SCORE~CALCULATE_ANOMALY_SCORE. This is the method where you must implement the scoring logic for your model. You get the list of context IDs as input and you must return the calculated scores for different context IDs as defined by output structure.

3.3 Import Model Definition

1. Save the xml in a text file name “model_property.xml”.
2. Zip the file.

3.4 Check Configuration

1. Go to System Monitoring UI.
2. In the scope selector, select the system type as the system type of the model. Enter.

3. In the configuration UI, under System Anomaly Configuration, click on a system.
4. You should be able to see the imported model and its associated metrics.

5. Switch on the activate button.

3.5 Check Monitoring

1. After you have implemented the method for scoring systems based on your model in the method `IF_SYSTEM_ANOMALY_SCORE~CALCULATE_ANOMALY_SCORE`, you should be able to see the result of the calculation on result UI after anomaly engine job run.
2. Open System Monitoring UI. Select system type as the system type of model in the score selector.
3. Click on “System Anomaly Prediction” page on the left side menu.
4. You should be able the see the result for the system configured.