SYSPRO Rules Engine

SYSPRO 8

Reference Guide

Published: May 2021

SYSPRO™

CONTENTS

SYSPRO Rules Engine

SYSPRO Rules Engine

Exploring

Where it fits in?

The **SYSPRO Rules Engine** helps you streamline your business processes by acting as a sophisticated *if/then* statement interpreter (i.e. rule translator).

A set of services monitor your SYSPRO transactions in real time and (once a specific set of rules is defined) they analyze and determine when something you're interested in happens. Your configured actions required by each rule are then processed accordingly.

Rules are applicable to all SYSPRO transactions, regardless of where they originate (e.g. SYSPRO core product, **SYSPRO Avanti**, **SYSPRO Espresso**, etc.).

The SYSPRO Rules Engine comprises the following components:

- SYSPRO 8 Rules Data Service
- SYSPRO 8 Rules Engine Service
- Rules Administrator

Once you start using the **SYSPRO Rules Engine** and have configured your required rules, your SYSPRO transactions continue as usual and post to the SYSPRO database.

> 💡 The **SYSPRO Rules Engine** has access to a set of business objects. This means you can create dynamic rules, like setting a price alert from within the **SYSPRO Bot**.

Architecture

REFERENCE GUIDE: SYSPRO Rules Engine
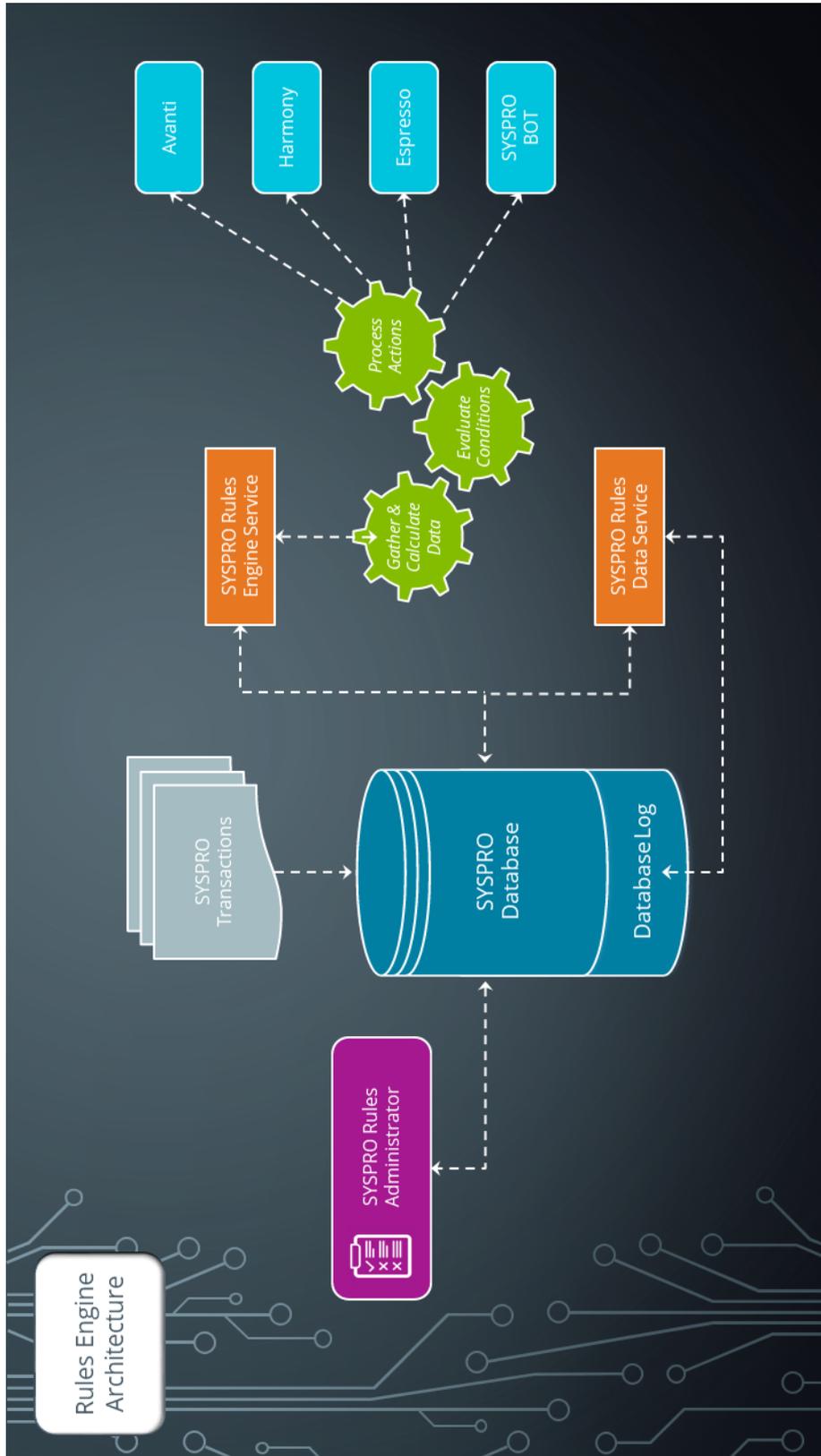
SYSPRO 8 Rules Data Service

This service monitors your SYSPRO transaction log in real time (e.g. inserts, updates or deletes) and extracts specific information required for current and active rules.

The data extracted is then persisted in the SYSPRO database, which enables the **SYSPRO 8 Rules Engine Service** to process the data without intervention so that it can execute the actions required by each configured rule.

> 💡 The information extracted includes before and after values (e.g. if a stock code's description is changed from 'Bike' to 'Mountain Bike', both values are stored).
>
> This is useful with regard to *delete-type* transactions as you have access to the whole record deleted, not just the key.

SYSPRO 8 Rules Engine Service

This service processes the data extracted by the **SYSPRO 8 Rules Data Service** after monitoring the SYSPRO transaction log.

The service is responsible for:

- Processing each log entry relayed by the **SYSPRO 8 Rules Data Service** service.
- Gathering the required data and calculating the additional values required.
- Evaluating sets of conditional statements.
- Executing all actions required per rule.

Rules Administrator

The **Rules Administrator** is a *SYSPRO Avanti* program responsible for configuring rules (by company or system-wide) that are stored in the SYSPRO database.

Terminology

Rule

Rules consist of conditions and actions.

A condition is evaluated and (if true) the **SYSPRO Rules Engine** initiates the defined actions.

Rule action

Actions are executed when the conditions of a rule are met.

> **FOR EXAMPLE:**
>
> These actions range from displaying a Harmony message, to creating a log file in a specific location, or tracking the history of a specific database column.

Rule condition

Conditions act as triggers that initiate a specific action according to how they have been configured.

They let you fine-tune a rule with the records to be excluded and when certain actions must be performed (i.e. they act as a set of conditional expressions that must be met before any rule actions are performed).

Rule target

A rule target indicates the database table to which a rule applies, as well as the operation on that table (e.g. All, Insert, Update or Delete).

Targets can be set as company-specific or system-wide.

Rule variable

Variables are used to define or manipulate any values required in the conditions or actions of a rule. The different property packets of a variable include the following:

Variable	Description
New	Variables that begin with this property packet contain the new values of a record *after* an insert or update occurs. They are only available for **Insert** or **Update** type operations.
Old	Variables that begin with this property packet contain the previous value of a record *before* an update or delete occurs. They are only available for **Update** or **Delete** type operations.

Variable	Description
Current	Variables that begin with this property packet contain the current value of a record. 💡 This is useful if the record has changed since the **SYSPRO 8 Rules Data Service** flagged the transaction. Most of the time these values are the same as the `New` variables. We recommend using the `New` property packet, as `Current` variables have a slight overhead.
Global	Variables that begin with this property packet are values related to the transaction, such as: ■ SchemaName ■ TableName ■ Operation ■ ChangedFields ■ TransactionDate ■ RuleId ■ LevelId ■ LevelValue ■ MessageId ■ BatchIndex ■ SystemWideDb
Var	Variables that begin with this property packet are custom variables that have been defined in the **Variables** pane of the **Rules Administrator** program.

Starting

Prerequisites

Technology

The following technology prerequisites are applicable to using this feature:

- SYSPRO 8 Rules Engine Service
- SYSPRO 8 Rules Data Service
- SYSPRO Avanti
- SYSPRO 8 Avanti Initialization Service

Configuring

The following options must be configured within the **Setup Options** program (*SYSPRO Ribbon bar > Setup > Setup Options > System Setup*) to use this feature:

Form name	Relevant options
Rules Engine/Harmony	This defines the companies you want the **SYSPRO 8 Rules Engine Service** to monitor. - Rules engine options - Active company list

	This defines the options that will let you access the relevant *SYSPRO Harmony* actions in the **Rules Administrator** program.
	
	- Harmony options
	  - Active company list
	  - Harmony API service address
	  - Harmony service address

SQL	This defines your SQL credentials which the service uses for its database connections.
	
	- Harmony database connection
	  - Authentication
	  - SQL Server name
	  - Login
	  - Password

| Form name	Relevant options

Connectivity	This defines the options that will let you access the relevant *SYSPRO Avanti* and *SYSPRO Espresso* actions in the **Rules Administrator** program.
	
	    ■ Avanti settings
	        □ Avanti website address

	■ Espresso settings □ Notification address

Artificial Intelligence	This defines the options that will let you access the relevant **SYSPRO Bot** actions in the **Rules Administrator** program.
	
	    ■ Bot
	        □ Azure Bot website

Restrictions and Limits

■ The **Rules Administrator** program is currently only accessible from within *SYSPRO Avanti*.

Solving

Error messages

Rules Engine is not enabled in SYSPRO
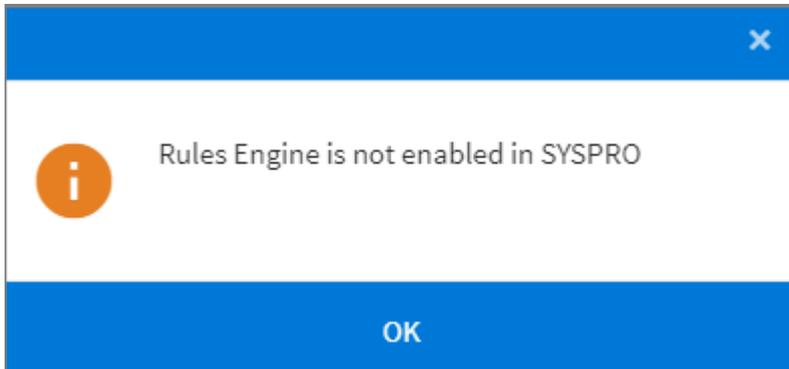
Cause

This message is displayed when loading the **Rules Administrator** program in *SYSPRO Avanti* because of a missing configuration in the SYSPRO core product.

Solution

You need to specify a list of companies that the **SYSPRO 8 Rules Engine Service** must monitor.

Configure the following options against the **Rules Engine/Harmony** form of the **Setup Options** program:

- Rules Engine options
  - Active company list

FAQs

How do I define a new rule?

1. Open the **Rules Administrator** program in *SYSPRO Avanti*.

2. Select the **Add New Rule** icon ( ).

3. From the **New Rule** window, enter the following mandatory information:

   a. The **Name** of the new rule.

   b. A **Description** for the rule.

   c. The **Table** in the database to which the rule applies.

   d. Whether the rule applies at company level or system-wide in the **Category** field.

      If at **Company** level, ensure that you enter the relevant company ID at the **Company** field.

   e. Whether the rule should result from an `Insert` (new record), `Update` (changed record) or `Delete` (removed record) in the database.

      Alternatively, you can select **All** to cover all of these transactions.

   Tick the **Enable** checkbox to enable the rule immediately, or save the rule and enable it at a later stage.

   Select the **Add Rule** icon ( ).

4. From the **Rule Information** pane, expand the **Variables** section to apply any additional fields or calculations.

   a. Select the **Add new variable** icon ( ).

   b. Indicate the **Name** of the variable so that it becomes usable later in your rule (e.g. `ContactName`).

   > The **Name** can contain letters, digits, and the underscore character (_) but it can't contain any spaces.
   >
   > The first character of the **Name** must be a letter.

   c. Indicate the **Type** and **Value** of the variable.

   d. Repeat for all variables that you want to apply to the rule.

   e. Select the **Save Rule** icon ( ).

5. From the **Rule Information** pane, expand the **Conditions** section to filter out your transactions.

    a. Select the **Add new condition** icon ( ).

    b. Indicate the details of the required condition.

    c. Repeat for all conditions that you want to apply to the rule.

    d. Select the **Save Rule** icon ( ).

6. From the **Rule Information** pane, expand the **Actions** section to assign the action the rule should perform.

    a. From the **Actions** toolbar, select the required action.

        Add Avanti Notification

        Add Bot Message

        Add Delete Rule

        Add Disable Rule

        Add Emailing Action

        Add Espresso Notification

        Add File Log

        Add Harmony Message

        Add Track History

        Add Workflow Action

    b. Repeat for all actions that you want to apply to the rule.

7. Select the **Save Rule** icon ( ).

What variable types are available when defining custom variables?

Depending on your requirements, you can select to use any of the following variables when defining a custom variable in the **Variables** pane of the **Rules Administrator** program:

String

A String variable holds zero or more characters (e.g. letters, numbers, spaces, comma, etc.).

> **FOR EXAMPLE:**
>
> `email@company.com`

Int32

An Int32 variable is a value type that represents signed integers, with values that range from negative 2,147,483,648 to positive 2,147,483,647.

> **FOR EXAMPLE:**
>
> `5000`

Double

A Double value type represents fractional or whole values and can contain up to 15 digits in total (including those before and after the decimal point).

> **FOR EXAMPLE:**
>
> `35,25`

> When selecting this variable type, ensure that it matches the **Regional Format** of the machine on which the **SYSPRO 8 Rules Engine Service** is installed.

Boolean

A Boolean variable represents either a true or false value.

Date/Time

The DateTime value type represents dates and times, with values ranging from `12:00:00 midnight, January 1, 0001 A.D. (Anno Domini)` through to `11:59:59 P.M., December 31, 9999 A.D.`

> When selecting this variable type, ensure that it matches the **Regional Format** of the machine on which the **SYSPRO 8 Rules Engine Service** is installed.

> **FOR EXAMPLE:**
>
> `1991/03/08 02:35:00` or `1991/03/08`

SQL

A SQL type variable lets you:

- select values directly from your SQL tables.
- join a foreign table.
- retrieve aggregate values.

You can use table variables and variables defined before this variable in the SQL statement.

The format to use is `{New.xxx}`, `{Old.xxx}`, `{Current.xxx}`, `{Var.xxx}` and `{Global.xxx}`.

> For system-wide database joins, ensure that you use the `{Global.SystemWideDb}` variable (e.g. `[{Global.SystemWideDb}].[dbo].[AdmOperator]`).

EXAMPLES:

select MAX(Description) as StockCodeDescription from InvMaster where StockCode = 'A100'

selectMAX(SellingPrice) as SellingPrice from InvPrice where StockCode = '{New.StockCode}' and PriceCode = '{New.ListPriceCode}'

selectSum((MOrderQty * MPrice) + (NMscChargeValue)) as OrderValue FROM [SorDetail] where SalesOrder = '{New.SalesOrder}'

selectMAX(CreditLimit) as CreditLimit, MAX(OutstOrdVal) as OutOrderVal from ArCustomer where Customer = '{New.Customer}'

In your statement, you can only return a single row, and the return column's alias (e.g. `MAX (Description) as StockCodeDescription`) is used as the variable name. We recommend that the alias name is the same as the variable name.

CSharp

> CSharp variables must return a value. The return type can be any type and remain that type for the remainder of the rule.
>
> If a Double type value is returned, it is treated as a double in the **Conditions** and **Actions** sections of the rule.

CSharp variables are very effective for performing various things, including the following:

- Calculations

  FOR EXAMPLE:

  ```
  return (New["PromotionValue"] / New["PromotionLimit"]) * 100;
  ```

- Rounding

FOR EXAMPLE:

```
return int.Parse( Math.Round((ML["PODaysLate"]), 0).ToString());
```

or

```
return Double.Parse( Math.Round((ML["ChanceOrderWillBeLate"]), 4).ToString
()) * 100;
```

- Assigning descriptions to flags

FOR EXAMPLE:

```
 switch((string)New["TrnType"])
 {
 case "R":
  return "Receipt";
 case "I":
  return "Issue";
 case "P":
  return "Physical";
 case "T":
  return "Transfer";
 case "A":
  return "Adjustment";
 case "C":
  return "CostChange";
 case "M":
  return "CostModification";
 case "B":
  return "BinTransfer";
 case "S":
  return "Sale";
 default:
  return "Other";
 }
```

- Date calculations

FOR EXAMPLE:

```
return (New["DateReceived"] - New["LineDueDate"]).TotalDays;
```

- Misc

FOR EXAMPLE:

```
if(Var["DaysLate"]="1")
        return "day";
else
        return "days";
```

or

```
return DateTime.Parse(New["EstArrivalDate"].ToString()).ToShortDateString();
```

You can use table variables and variables defined before this variable in the SQL statement.

The format to use is `New["xxx"]`, `Old["xxx"]`, `Current["xxx"]`, `Var["xxx"]` and `Global["xxx"]`.

Why should I use conditions when defining a rule?

Conditions are a set of statements that a rule needs to pass before actions can be performed.

Values of the same type can be compared to evaluate to a true or false outcome.

The combination of these statements using the AND or OR modifiers evaluates to a single true or false outcome.

If this final outcome is true, then the rule moves to processing each action.

The following scenarios demonstrate some of the different types of conditions you can apply:

- *Scenario: You want to check whether a specific value has changed for an update rule.*

FOR EXAMPLE:

⊡ Conditions ∧

≡₊ | Search

Drag a column header and drop it here to group by that column

	Bracket	Variable	Condition	Variable	Bracket	Modifier
🗑 -		New["SellingPrice"]	Not equal to	Old["SellingPrice"]	-	-

- *Scenario: You want to check a specific flag on the record (e.g. if the OnHold flag has changed and equals Y).*

FOR EXAMPLE:

⊡ Conditions ∧

≡₊ | Search

Drag a column header and drop it here to group by that column

	Bra...	Variable ↑	Condition	Variable	Bracket	Modifier
🗑 -		New["CustomerOnHold"]	Not equal to	Old["CustomerOnHold"]	-	AND
🗑 -		New["CustomerOnHold"]	Equal to	"Y"	-	-

- *Scenario: You want to know when a value has increased.*

FOR EXAMPLE:

> Using a numeric table variable (e.g. `New["SellingPrice"]`) and a fixed numeric value (e.g. `10`) in the same condition will fail.
>
> Ensure that you first convert the fixed value to a `SqlDecimal`.

How do I query a hashtag table for History Tracking rules?

You can use the COMQRL business object to query the hashtag table if you have configured a Track History rule.

FOR EXAMPLE:

```
<Query>
  <Key>
    <Function>QH</Function>
    <HistoryTableName>InvPrice</HistoryTableName>
    <HistoryColumnName>SellingPrice</HistoryColumnName>
    <HistoryColumnDataType>N</HistoryColumnDataType>
    <HistoryColumnKey1>B100</HistoryColumnKey1>
    <HistoryColumnKey2>A</HistoryColumnKey2>
    <HistoryColumnKey3 />
    <HistoryColumnKey4 />
    <HistoryColumnKey5 />
    <HistoryColumnKey6 />
    <HistoryColumnKey7 />
    <HistoryColumnKey8 />
    <HistoryColumnKey9 />
    <HistoryColumnKey10 />
    <HistoryColumnKey11 />
    <HistoryColumnKey12 />
    <HistoryColumnKey13 />
    <HistoryColumnKey14 />
    <HistoryColumnKey15 />
  </Key>
  <Options>

<IncludeAdditionalHistoryColumn1>InvMaster.Description</IncludeAdditionalHistoryColumn1>
    <IncludeAdditionalHistoryColumn2 />
    <IncludeAdditionalHistoryColumn3 />
    <IncludeAdditionalHistoryColumn4 />
    <IncludeAdditionalHistoryColumn5 />
    <IncludeAdditionalHistoryColumn6 />
    <IncludeAdditionalHistoryColumn7 />
    <IncludeAdditionalHistoryColumn8 />
    <IncludeAdditionalHistoryColumn9 />
  </Options>
</Query>
```
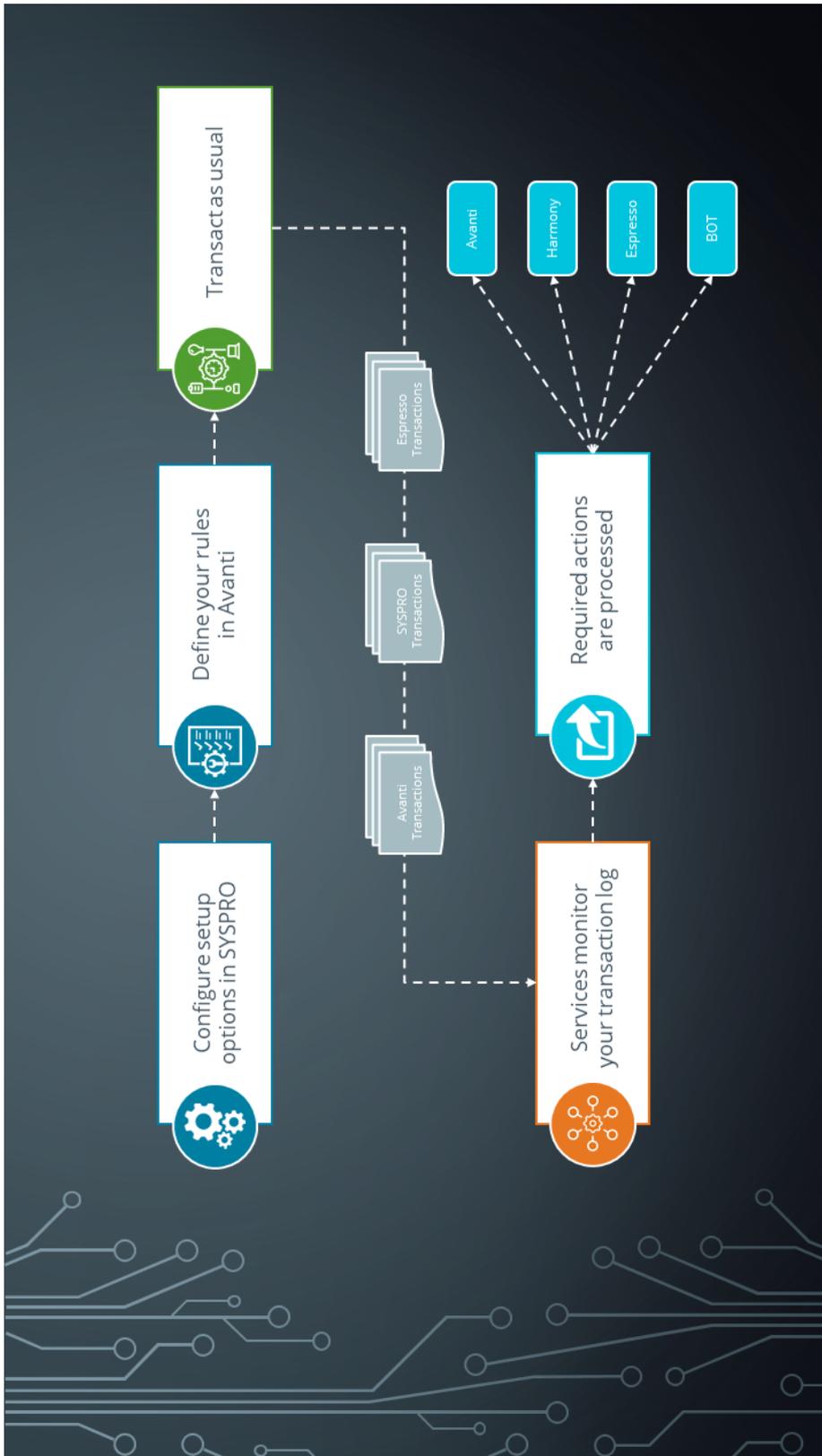
Using

Process

The following process depicts how to configure and access the **SYSPRO Rules Engine**:

1. Configure the required options against the following forms of the **Setup Options** program:

   - SQL
   - Connectivity
   - Rules Engine/Harmony
   - Artificial Intelligence

2. Define your required rules using the **Rules Administrator** program in *SYSPRO Avanti*.

3. Transact as usual using any of the SYSPRO sources.

4. The **SYSPRO 8 Rules Engine Service** and **SYSPRO 8 Rules Data Service** then monitor your transaction log and launch the defined rules where applicable.

5. The configured actions are then processed to the defined platforms.

Rule actions

There are a number of actions that you can apply to rules. These will be expanded with future SYSPRO releases.

SYSPRO Avanti Notification

Messages sent to an operator in **_SYSPRO Avanti_** will also appear in the operator's **To Do List**.

Available parameters:

- Title

- Subtitle

- Operator

- Program to call *(Optional)*

  This lets you indicate a SYSPRO program that should open automatically when the operator selects the notification.

- Program input key *(Optional)*

> **FOR EXAMPLE:**
>
> *(Title)* Stock below minimum quantity
>
> *(Subtitle)* Stock on `{New.StockCode}` has fallen below the required minimum quantity on hand
>
> *(Operator)* John Doe
>
> *(Program to call)* INVPEN
>
> *(Input key)* `{New.StockCode}`
>
> Therefore, when John Doe selects this Avanti notification, the system automatically opens the **Inventory Query** program for stock code `{New.StockCode}`.

SYSPRO Bot Message

Messages sent to an operator on the **SYSPRO Bot** use the proactive messaging feature and can also begin conversations that can flow into even further processes.

Available parameters:

- Operator

- Message

- Skill *(Optional)*

  This lets you indicate a Bot skill to invoke after the operator reads the message.

- Skill keyfields *(Optional)*

> **FOR EXAMPLE:**
>
> *(Operator)* John Doe
>
> *(Message)* A100 received into stock.
>
> *(Skill)* Create Sales Order (`SalesOrder.CreateSalesOrder`)
>
> *(Skill key)* Customer `00001` - Quantity `10` - StockCode `A100`
>
> Therefore, when John Doe reads this Bot message, the Bot automatically begins creating a sales order for customer `00001` for `10` of stock code `A100`.

SYSPRO Espresso Notification

Sending messages to an operator in ***SYSPRO Espresso*** is a great way to get hold of operators on the move.

Available parameters:

- Operator
- Message
- Application to call *(Optional)*

  This lets you indicate an Espresso application that should open automatically when the operator selects the notification.

- Application key/session variable *(Optional)*

> Espresso users must be signed in to receive these notifications

SYSPRO Harmony Beat

Beats posted to ***SYSPRO Harmony*** can include both a message (beat text) and sentiment.

Hashtags can also be used in beat text and are included in Harmony follow feeds.

Log File

This action creates a physical file that contains all the changes resulting from the SYSPRO transactions you defined to be logged.

File types available for selection:

- `TXT`
- `CSV`
- `XML`

You can also indicate how much detail must be logged by defining the **File Level**.

Track History

This action lets you store every change to a column in the database, providing detailed history tracking.

It creates a Hashtag table (e.g. `InvPrice#`) with the same key, including a `PostDateTime` entry. If you enable the **Allow Duplicates** option, it records all entries, even if the value is the same as the previous entry in the Hashtag table.

> **FOR EXAMPLE:**
>
> Track and store the selling price of your stock codes as they change.
>
> Some prices might change daily, but the **SYSPRO Rules Engine** persists every single change.

Delete Rule

This action deletes a rule after all its other actions have executed.

Disable Rule

This action disables a rule after all its actions have executed (useful for once-off rules that should be disabled after a successful single execution).

Email

This action lets you send an email.

Workflow Action

This action lets you call into a *SYSPRO Workflow Services* operation.

Sample Rules

The following Sample Rules are available from the **Rules Administrator** program in *SYSPRO Avanti* (located under **SYSPRO Templates**).

You can use these as is, or as templates to customize to your specific requirements.

AP payment cycle (Harmony)

Description	Notify when a payment cycle is run.
Actions	Post Harmony Beat
Sample AI Model	None

AR Invoice Late Payment Predicted

Description	Notify when a late invoice payment is predicted for a customer.
Actions	- Post Harmony Beat - Send Avanti Notification
Sample AI Model	CustomerInvoicePayDays

Asset added (Harmony)

Description	Notify when a new asset is added.
Actions	Post Harmony Beat
Sample AI Model	None

Asset changed (Harmony)

Description	Notify when an asset is changed.
Actions	Post Harmony Beat
Sample AI Model	None

Asset deleted (Harmony)

Description	Notify when an asset is deleted.
Actions	Post Harmony Beat
Sample AI Model	None

Cash Book deposit (Harmony)

Description	Notify when a deposit is made to cash book.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Cash Book withdrawal (Harmony)

Description	Notify when a withdrawal is made from Cash Book.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Credit limit (Harmony)

Description	Notify when a customer falls within 10% of their defined credit limit.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Customer balance store

Description	Track the value of a defined customer's balance when changes occur.
Actions	▪ Track History
Sample AI Model	None

Customer on hold (Harmony)

Description	Notify when a customer is placed on hold.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Customer order value change (Harmony)

Description	Notify when a defined customer's outstanding order value changes.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Customer value store

Description	Track the value of a defined customer's outstanding order value when changes occur.
Actions	▪ Track History
Sample AI Model	None

Delivery note printed (Harmony)

Description	Notify when a delivery note is printed.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Delivery received late (Harmony)

Description	Notify when a delivery is received late.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Detail line added to RMA (Harmony)

Description	Notify when a new detail line is added to an RMA.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Dispatch note released for invoicing (Harmony)

Description	Notify when a dispatch note is released for invoicing.
Actions	• Post Harmony Beat
Sample AI Model	None

GL account added (Harmony)

Description	Notify when a new general ledger account is added.
Actions	• Post Harmony Beat
Sample AI Model	None

GL Period Change (Harmony)

Description	Notify when a change is made to the defined GL Period.
Actions	• Post Harmony Beat
Sample AI Model	None

Harmony Beat for everything (Harmony)

Description	Notify on every transaction that occurs.
Actions	• Post Harmony Beat
Sample AI Model	None

Harmony for all changes to database (Harmony)

Description	Notify on all changes made to the database.
Actions	• Post Harmony Beat
Sample AI Model	None

Invoice 10% difference (Harmony)

Description	Notify when there is more than a 10% difference on an invoice payment.
Actions	• Post Harmony Beat
Sample AI Model	None

Job deleted for no customer (Harmony)

Description	Notify when a job with no customer defined is deleted.
Actions	■ Post Harmony Beat
Sample AI Model	None

Job deleted (Harmony)

Description	Notify when a job is deleted.
Actions	■ Post Harmony Beat
Sample AI Model	None

Job received late for no customer (Harmony)

Description	Notify when a job with no customer defined is received late.
Actions	■ Post Harmony Beat
Sample AI Model	None

Job received late (Harmony)

Description	Notify when a job is received late.
Actions	■ Post Harmony Beat
Sample AI Model	None

Job received short for no customer (Harmony)

Description	Notify when a job with no customer defined is short-received.
Actions	■ Post Harmony Beat
Sample AI Model	None

Job received short (Harmony)

Description	Notify when a job is short-received.
Actions	■ Post Harmony Beat
Sample AI Model	None

Job status rule

Description	Notify if a new job will be late or on time.
	The AI used in this rule compares the `ActCompleteDate` column against the `OrigDueDate` column in the WipMaster table in order to process this prediction.
Actions	- Send Avanti Notification
Sample AI Model	`JobStatus`.

LCT shipment might arrive late

Description	Notify if a shipment's arrival is potentially more than 10 days late.
Actions	- Post Harmony Beat - Send Avanti Notification
Sample AI Model	`LctDaysLate`.

Limited promotion 5% from expiring (Harmony)

Description	Notify if a promotion is close to expiring.
Actions	- Post Harmony Beat
Sample AI Model	None

Line received on RMA (Harmony)

Description	Notify when an RMA line is received.
Actions	- Post Harmony Beat
Sample AI Model	None

Log everything to CSV file

Description	Creates a `csv` file to record all transactions logged.
Actions	- Log File
Sample AI Model	None

Log everything to text file

Description	Creates a `TXT` file to record all transactions logged.
Actions	▪ Log File
Sample AI Model	None

Log everything to XML file

Description	Creates a `XML` file to record all transactions logged.
Actions	▪ Log File
Sample AI Model	None

New bin added (Harmony)

Description	Notify when a new bin is added.
Actions	▪ Post Harmony Beat
Sample AI Model	None

New BOM added (Harmony)

Description	Notify when a new bill of material is added.
Actions	▪ Post Harmony Beat
Sample AI Model	None

New job created for no customer (Harmony)

Description	Notify when a new job is created with no customer defined against the job.
Actions	▪ Post Harmony Beat
Sample AI Model	None

New job created (Harmony)

Description	Notify when a new job is created.
Actions	▪ Post Harmony Beat
Sample AI Model	None

New lost sale (Harmony)

Description	Notify when a sale is lost.
Actions	■ Post Harmony Beat
Sample AI Model	None

New movement (Harmony)

Description	Notify when there is a new movement.
Actions	■ Post Harmony Beat
Sample AI Model	None

New RMA created (Harmony)

Description	Notify when a new RMA is created.
Actions	■ Post Harmony Beat
Sample AI Model	None

New sales order (Harmony)

Description	Notify when a new sales order is loaded.
Actions	■ Post Harmony Beat
Sample AI Model	None

New stock code added

Description	Notify when a new stock code is added to the inventory list.
Actions	■ Post Harmony Beat ■ Send Bot Message ■ Send Avanti Notification
Sample AI Model	None

New stock code added (Harmony)

Description	Notify whenever a new stock code is added.
Actions	■ Post Harmony Beat
Sample AI Model	None

Over issue to a job (Harmony)

Description	Notify when there is an over issue to a job.
Actions	■ Post Harmony Beat
Sample AI Model	None

Predict supplier payment

Description	Predict if supplier payments will be on time or late.
Actions	■ Send Avanti Notification
Sample AI Model	`ApInvoicePayment`.

Price change on sales order (Harmony)

Description	Notify when there is a price change on a sales order.
Actions	■ Post Harmony Beat
Sample AI Model	None

Purchase order receipt rej at receiving (Harmony)

Description	Notify when a purchase order receipt is rejected during the receiving process.
Actions	■ Post Harmony Beat
Sample AI Model	None

Sales order line added (Harmony)

Description	Notify when a new sales order line is added.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Sales order return reason

Description	Notify why sales orders over a certain value might be returned.
Actions	▪ Send Avanti Notification
Sample AI Model	LostSaleReason

Stock code added warehouse allocation (Harmony)

Description	Notify when a new stock code is added to a defined warehouse.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Stock code price change (Harmony)

Description	Notify when the price of a stock code changes.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Stock issued to job (Harmony)

Description	Notify when stock is issued to a job.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Stock scrapped at receiving (Harmony)

Description	Notify if stock received from a supplier is scrapped during the receipt process.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Stock take completed (Harmony)

Description	Notify when a stock take is completed for a defined warehouse.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Store rough order value

Description	Tracks the value of the store's rough order value.
Actions	▪ Track History
Sample AI Model	None

Supplier added (Harmony)

Description	Notify when a new supplier is added.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Supplier delivery received short (Harmony)

Description	Notify if a supplier's delivery is short-received.
Actions	▪ Post Harmony Beat
Sample AI Model	None

Supplier on hold (Harmony)

Description	Notify if a supplier is placed on hold.
Actions	▪ Post Harmony Beat
Sample AI Model	None

When contact details changed (Harmony)

Description	Notify if contact details are changed and includes what the new details are.
Actions	▪ Post Harmony Beat
Sample AI Model	None

When delivery will be short delivered (Harmony)

Description	Notify if an expected delivery will be short-supplied.
Actions	▪ Post Harmony Beat
Sample AI Model	None

When new promotion added (Harmony)

Description	Notify when a new promotion is added.
Actions	▪ Post Harmony Beat
Sample AI Model	None

When POD captured for delivery (Harmony)

Description	Notify when a proof of delivery is captured.
Actions	▪ Post Harmony Beat
Sample AI Model	None

WIP after stock scrapped at inspection (Harmony)

Description	Notify if stock is scrapped during an inspection process.
Actions	■ Post Harmony Beat
Sample AI Model	None

WIP Material Cost

Description	Notify when the actual material cost exceeds the expected material cost.
Actions	■ Post Harmony Beat ■ Send Bot Message ■ Send Espresso Notification ■ Send Avanti Notification
Sample AI Model	None

Affected Programs

The following indicates areas in the product that may be affected by implementing this feature:

Rules Administrator

This is a new **SYSPRO Avanti** program that enables you to configure rules (by company or system-wide).

SYSPRO

www.syspro.com