If a **Process Definition Document (PDD)** is the “as-is” capture of the process, then a Solution Design Document (SDD) is the “to-be”. That is, a diagrammatic representation of what the process will look like once it’s automated.

When the process has been approved for automation and the PDD has been signed off by the process owner & technical team, then creating the SDD is the next step in the cycle. The SDD should be a bridge between the PDD and the automation which will be built. It should serve as a technical guide for the developer, outlining key process steps and how they will be automated, the applications involved, and even how exceptions should be handled.

**Walkthrough of a Typical SDD**

An RPA Solution Architect or a senior RPA Developer/Engineer is the ideal candidate for owning and writing an SDD. However, before the pen hits the paper. - err fingers hit the keyboard - it’s important to remember that the audience for the document will be an RPA Developer or Engineer who knows Automation 360, but doesn’t know the process. Therefore it’s really important to make sure the process and key outputs from the PDD are accurately captured in the SDD. So let’s walk through a typical SDD.

### 1. Overview of the process

The first section in the SDD should provide an introduction to the process to be automated and a link to the PDD. Be mindful not to duplicate the PDD but to include a high-level view of the process (something which should already be in the PDD) to give the reader an understanding of what the process does. Focusing on what outputs are achieved by the process can also help focus the reader on what’s important.

### 2. Applications

Up front, the SDD should list the core applications on which the process is dependent. Alongside this, the following details should be provided:

* Name of the application
* Description - What it’s used for in the application
* Version - The version of the application required to run the process
* Tech/Dependencies - Does the application make use of any plugins which will also need to be installed?
* Environment - The operating system or virtualization software (e.g. Citrix) that the application will need to run in
* Authentication - Does the application require credentials or an SSO login
* Bot role - What is the minimum level of access the Bot will need e.g. read-only or write

Any pre-requisites required by the applications should also be contained within this section of the SDD:

* The device(s)/device pool the Bot will run on
* Browser and any associated plugin requirements
* Browser zoom level
* Operating System resolution

### 3. To-be process

The next and most critical section of the SDD is the outline and design of how the process will function in its automated form. This section will consume the most time and requires the most thought as it ultimately directs the development work on the Bot. A good indicator of a poorly written SDD is the amount of time required to explain or clarify the “to-be” process from the author to the developer.

### 3.1 Solution pattern

The solution pattern for the process (Run Remote via Schedule, Run Remote on Demand, Run Local on Demand and Run Local through AARI, etc) should be identified before the development steps are determined. More on solution patterns can be found [**here**](https://community.automationanywhere.com/developers-blog-85009/solution-pattern-modeling-framework-85151). The SDD you use could be generic (for all Solution Patterns) or could be specific to the Solution Pattern you’re following.

### 3.2. Schedule

If the solution pattern requires a schedule then this should be broken out in this section with the following information:

* Start time
* Start time zone
* Repetition - How often should the Bot be run e.g. every 1 hour
* Repetition end time - At what time should the Bot stop running
* Repetition end time zone
* Frequency - How often should this schedule be run

#### 

### 3.3 Inputs and Outputs

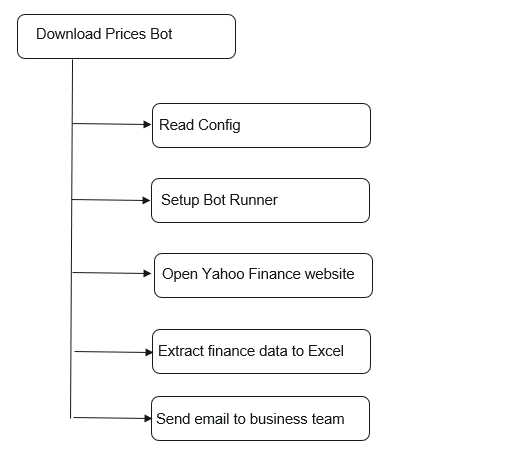
This section of the SDD should explicitly focus on any inputs that the automation requires and the outputs it will produce. Whilst this section does not cover how the items are processed, it does document what inputs are expected for the automation to run successfully, as well as the expected outputs:

* Examples of typical inputs are datasets from Excel/CSV files, config files, pdf/image files, etc.
* Examples of typical outputs are updated datasets and emails

#### 

### 3.4 Process Flow Diagram

Before presenting the reader with the full blown solution, a high-level diagram should be presented that covers the key steps in the solution design. Of course, it’s difficult to complete this section before the exact steps have been constructed; however, using this section to iterate ideas of how the “to-be” process will look may help clarify design decisions. The process flow diagram should summarize the individual automation (e.g. task bots in Automation 360) which make up the “to-be” process, how they flow and link together, while summarizing what they do. An example can be seen below:



Whilst a flow chart tool is advantageous, it’s not a necessity. But note that it may help to accelerate the development efforts, as creating an automation in flow view in Automation 360 is very similar to building out a process flow.

#### 

### 3.5 Process Steps

Now, this is where the bulk of the SDD and all of the hard work resides. In this section, the Developer is presented with the “to-be” process. The (hopefully) perfect interpretation of the current process, married with the technical solution to automate it. The devil very much is in the detail. When designing the solution, it is imperative that reusable assets are considered. These could be from the Automation Anywhere [Bot Store](https://botstore.automationanywhere.com/) or your own organization's public repository (in a “LIBRARY” or similar folder to be sure its obvious). The inclusion of such assets dramatically helps to reduce development time and eases the change management burden should an underlying application or process change. In this section, each successive step should represent a task bot in the overall process. A step should contain:

* Step number - e.g. “Step #1”
* Name of the task - e.g. “DownloadPricesBot”
* The path to the task bot in the Control Room - e.g. “BotsFinanceS&P500”
* Description of the task bot - e.g. “This is the main task bot which orchestrates the overall logic of the solution”
* Detailed steps that the task bot should carry out - For example:
  + Call “01\_Read\_Config” subtask to load the config variables into a dictionary
  + Call “01\_Setup\_BotRunner” subtask to set the resolution and clear the desktop of the available Bot Runner
  + Open Yahoo Finance page for S&P 500 historical prices as per the “sp500\_url” config variable
  + Create new Excel file from the “sp500\_output\_dir” config variable
  + Give new Excel file a filename in the format “yyyy-mm-dd\_hhmm - prices\_download”
  + Loop through each of the paginations when on the Yahoo Finance page:
    - Extract the table data into the new Excel File
    - Making sure to append the data
  + Save the Excel file
  + Call “06\_Send\_Email” to send a completion email to the business team

**Note:** Exceptions and logging are kept separate from these process steps to keep the flow clear. The example above highlights the first stage in a typical process; a main task bot that orchestrates many other task bots. It is often this main task bot that is used as the high-level diagram in the process flow section.

### 

### 4. Event / Exception Management

In this section, the management of key events like logging and exceptions are captured and attributed to the process steps from section 3.5.

#### 

### Exceptions

Exceptions tend to be one of two things: process or system/application exceptions. All exceptions tend to follow a standard action of logging or capturing the exception, but how they’re handled is heavily dependent on the process itself. Also, it can be challenging to capture all of the possible exceptions up front in the SDD. Typically, exceptions arise during the building or testing phases when the production systems are run against Bots. Nonetheless, the key to capturing exceptions is to focus on each process step from section 3.5 and ask the question “what could go wrong in this step?”. If we apply this logic to the example from 3.5:

* High level process step - Step #1
* Event code - EXCEPTION\_101
* Description - Unable to load configuration variables as network file share is down
* Actions:
  + Log the exception using the “03\_Log\_Activity” shared task bot
  + Run the “03\_ExceptionMonitoring” task bot
  + Exit the bot

#### 

### Logging

Whilst there is no set pattern to logging, doing so frequently will shorten the debugging lifecycle if it arises. This can be a complex overhead to include in an SDD, so as a blanket statement it may be quicker to ask Developers to log after every application screen or Automation 360 command, applying flags to tone down the level of logging when in production. Most importantly, make sure your approach to logging/log management is consistent across automations. This makes supporting and troubleshooting automations much more straightforward.

### 

### 5. Control Room Configuration

The next stage in the SDD is to break out the necessary configuration which will be required in the Control Room to run the Bot. Remember that Bots often follow the SDLC principle of Dev, Test and Prod environments and will be moved across Control Rooms.

#### 

### 5.1. Folder structure

Firstly, the folder structure levels should be explicitly called out for where the task bots (created for this process) and assets will be stored. In the example from section 3.5, “BotsFinanceS&P500”:

* Level 0 = Bots
* Level 1 = Finance
* Level 2 = S&P500

To ensure that log files for the process can be easily accessed, the folder structure in the Control Room should be mirrored in the network shared drive (or the location of where the log will be stored).

#### 

### 5.2. Global Values

Where values in this Bot can be used across other Bots, utilizing Global Values in the Control Room may make more sense than using configuration variables. An example structure in an SDD is:

* Name - e.g. “env”
* Data type - e.g. “String”
* Description - e.g. “The current environment of the Control Room such as Dev, UAT or Prod“
* Sample Value - e.g. “Dev”

**Note:** Currently Automation 360 Control Rooms only support up to 150 Global Values.

#### 

### 5.3. Credentials

Credentials are an increasingly important item to think about upfront while designing your automation. If using the default Credential Vault then an example structure for defining credentials in the SDD is:

* Locker - e.g. “Finance”
* Credential - e.g. “Yahoo Finance login”
* App - e.g. “Chrome”
* Attributes - e.g. “Username”
* Type - e.g. “Standard”

If you utilize external credential applications like CyberArk or AWS Secrets Manager then the structure above may need amending.

#### 

### 5.4. Roles

Finally, any customized roles which access or execute the Bot should be captured in this section. It should be noted that the transferring of roles between Control Rooms is a common item for Bots not executing as intended. As a good practice though, the custom roles that support the specific solution pattern you’re using is a great model to follow. See the Solution Patterns section for more details.

### 

### 6. Automation Configuration

#### 

### 6.1. Folder structure - Network shared drive

In order to mimic the structure in the Control Room, additional folders may need to be created on the shared drive. These should be explicitly called out in this section.

#### 

### 6.2. Use case values / Configuration variables

Finally, the configuration variables which the Bot requires to run should be explicitly called out in this section alongside a suitable description.

### 

### 7. Bot summary

#### 

### 7.1. Existing reusable Bots

As noted in section 3.5, the consideration of reusable Bots is one of the most important tasks in the SDD production. In this section, they should be explicitly noted. Whilst this may feel like duplication, the consideration around change management for reusable assets should not be underestimated and explicitly calling out the Bots makes it easier to determine the risk factor.

#### 

### 7.2 New Bots

Typically in the production of a Bot, many new task bots will be developed which may have the potential to be reused in future endeavors. After section 3.5 is completed, these should be assessed and marked as potentially reusable in this section. Splitting out the task bots in this manner also enables Delivery Team Leads to more effectively plan on how the Bot can be developed. For example, grouping task bots based on function and sharing across many developers to reach a tight delivery timeline.

### 

### 8. Assumptions

In this last section of the SDD, any assumptions made in the design of the solution are called out. Most commonly, SDD authors will assume that test versions of applications are identical to production versions and available during the company working hours. Finally, any out-of-scope assumptions should also be called out.

### 