

*S.E.E.R., mod_openopc, et. Al deployed in: any *nix flavor*

GETTING STARTED WITH DEPLOYMENT

- Determining your goals and needs
- 2011_0322
- Guide version 1 (proofed)



SUMMARY

So you've decided to install S.E.E.R. - great; now what? This document aims to answer that question. What seems like a lot of hard to grasp and abstract purpose is actually a collection of highly specific, easy to master principles.

S.E.E.R. Is made up of several 'Models'. The list is always growing, with more models being developed (and revisions to old models) a constant activity. The thing to remember is that you are welcome to use ALL of the models, some of the models, or just one of the models - it's up to you. This approach allows a plant to deploy in 'steps', rather than 'all at once', minimizing headache and maximizing success.

PEOPLE AND PLACES

S.E.E.R. Requires Administrators, Oversight, and Backup personnel. So, you'll have to decide who that shall be for your location.

Typically, an Administrator will perform the 'day to day' answering of any user's questions, have a solid understanding of the OPC Devices (PLC's) that are providing data to S.E.E.R., and can manage the adding or removing of hardware from the system. Administrators should have a solid understanding of Unix / Linux as well as Win32 based operating systems, OPC Servers, and Networking - or be very eager and self-motivated to gain an understanding. There should be

at least 2 Administrators for your system, but we advise about having too many "cooks in the kitchen".

Oversight shall be one and only one person, who by virtue of this title is granted sole governance of the S.E.E.R. System, the hardware it resides on (computer server), and the devices it connects to (PLC's or other). This person does not have to be someone who is normally in charge of such things - otherwise it would be the Manager of IT in every location, which may or may not be the best person for the task. Rather, choose someone with a wide scope of knowledge in the areas of production / industrial equipment, computer systems in general, and the ideals and spirit of your company. Oversight's duty is not to run the day to day of the system, but to be an arbitrator who can 'split the baby in half' when there can not be agreement between (for example) the PLC Programmer / the Production Manager / the Machine Operator / the IT desktop group / the IT Infrastructure group.

Additionally, Oversight is responsible to any regulatory bodies (whether that be State, Local, or Federal government) with regard to ensuring the validity and security of any required documentation (ex. Weights-and-Measures).

Oversight shall not be an Administrator, and while he/she may very well typically support much of the system's backend devices / infrastructure, it is advisable that he/she be someone who does not typically support / maintain the system itself ... for the simple reason that a person cannot remain unbiased if they work on something every day and take pride in it.

If the plant needs, Oversight can provide Project Planning and Direction function with regard to S.E.E.R. -- "where are we going with this system?" / "how shall we use this system?" / "what are our goals, and how will I ensure they are met?"

Backup personnel will need to be assigned to perform daily (ideally 7 day, but many installations will only perform 5 day) backups of the system to tape. The preferred backup tool is STaRBUC (Standard Tape Resource Backup Console for Linux). STaRBUC is accessible via web-browser, and someone need only insert the days tape, and initiate the backup via a few clicks in the browser. This person should be someone who is normally authorized to be around and operate IN PRODUCTION MISSION CRITICAL COMPUTER SYSTEMS. This is a secure backup system which tracks the authenticated user performing the backup / what tapes were used in the backup / and what data was actually backed up.

You can read more on STaRBUC here:

http://www.spinellicreations.com/spark/project_starbuc.php

STaRBUC is available for download here:

<http://download.spinellicreations.com/starbuc/>

Typically, IT Help Desk and Support personnel are utilized for this role, however if need be an Administrator can also pull 'double duty' as a Backup Person.

Lastly, choose somewhere to place the S.E.E.R. Server and associated hardware. There will typically be a Near Line grade computer server, uninterruptable power supply backup, tape drive, and cabling. Best practice is to purchase rack-mount components and then rack the server, UPS battery, and tape drive together. Plug the server and tape drive into the UPS, and then plug the UPS into a DEDICATED CIRCUIT. Also, be sure you can provide robust ethernet to the location.

You should NEVER route OPC comm outside of your facility. While we're not a fan of routed communication to begin with, routing outside of your facility is both insecure and potentially unstable as degradation in response time will cause spurious results.

You should NEVER route S.E.E.R. Outside of your facility (or at least not outside of your company's secure internal network - we realize that your company may exist in many different locations; that's fine, provided the link back and forth is NOT PUBLIC).

S.E.E.R. Was built and tested to run in a LAN environment (from a security point of view). It is just secure enough to keep out any would be bad-guys within your organization, provided you don't make a habit of hiring pHp hackers, in which case, we can't help you. At its highest security setting, S.E.E.R. Is so tight that it's annoying to use, which is good (in a way). However we do not endorse the use of S.E.E.R. Over the World Wide Web / Internet, as the potential for exploiting the user-interface by moderately experienced hackers is very present (in order to ensure compatibility with old / deprecated browsers, and in order to ensure that S.E.E.R. Would work on slow / antiquated hardware [from the client's point of view], it was intentionally coded in a such a manner that lacks certain modern security measures). That said, we'd still be impressed if someone busted through it.

- WORKSHEET

Complete SEER_-_WORKSHEET_ADMINISTRATION_SETTINGS.ods and provide to your INSTALLER.

Things to Get Out of the Way

Non existent sensors / data / and how to deal with it...

Programmers, do not fear - S.E.E.R. Was built realizing that you may have MUCH of this data available from your devices, but not all of it. That's alright. Critical values are either easily identified or declared as such, whereas the rest of the data (while highly useful and generally awesome) can be 'faked' by simply providing 'zero' values to S.E.E.R.

Right now you're saying, "but I don't want to congest my backend network with a billion 'zero' bits flying around for no reason other than to fill a silly form!". Neither do we - you can use the same 'zero' bit over and over again. Typically, an OPC server is going to cache its reads, and in doing so, if you have (for sake of argument) 250 items that are 'null' or 'zero', and you take care to define all of them as (for example) B3:5/1 from the same device... then that data point will be read by the OPC server only once.

A best practice here is to define a "null bit" or "null variable" inside of each of your PLC's or OPC Devices. That way, anytime you have to pull from that device, and you're in the unfortunate position of not having all the sensors you require (and therefore need to toss in a 'zero'), you can use the pre-existing "null bit" every time.

Old gear / finnick communication protocols / what do you want me to do about it ?...

Got token ring? Not quite as charming as 'Got Milk?', but we digress. Ultimately, S.E.E.R. Sits on ethernet, which means all of your equipment has to (somehow) get to ethernet. You've got options, lots of them, but hardened industrial means are the best.

Bridging Profibus to Ethernet via a standalone PC/Workstation on the manufacturing facility floor is not a 'robust' solution; and generally it's a bad idea. But, if that's all you've got, we understand. That said, there are a lot of solutions out there for getting your squirrely networks over to ethernet.

Allen Bradley Controllogix PLC racks can support 'see-through' bridging of networks, where a rack outfitted with only an Ethernet card and a DHRIO card can effectively bridge up to 2 discrete Data Highway networks to a single (or multiple) Ethernet networks.

Go downstream a little further, an SLC-504 processor can be used to bridge a DH-485 network to Data Highway... and if that processor is tied to one of the afore-mentioned DHRIO cards in a Controllogix rack, then you have just figured out how to pull Data Sidewalk (DH-485) via ethernet (note: this will likely require Rockwell's proprietary OPC Server built into RSLinx - but the point is, it can be done).

The same can be said for various makes / models, but the point remains, a hardened industrial protocol bridge supporting the specific protocol of your OPC Device (and thereby supporting OPC communication over the bridge - to an OPC Server that supports both your device and viewing your device via a bridge) will be vastly superior in terms of reliability than trying to hack-the-world via software bridging.

That said, if you absolutely must, a performance gaming / CAD workstation PC with battery backup, hardware raid mirror, and outfitted with a proprietary interface card (such as an AB PKTX network PCI card) and an OPC Server software capable of negotiating that card - can and will work quite well for harvesting data via mod_openopc for delivery to S.E.E.R.

What's this Gonna Cost Me?

Depends - how far do you want to push it... your initial cost of ownership is semi-large (depending on deployment size), but will typically range between \$5,000 and \$30,000 USD, which includes a beefy server / backup mechanism / and uninterruptable power. The software is free (well, 99% of it - you will have to buy an OPC Server from a recognized firm such as Rockwell / Kepware / Matrikon / et. al. And you'll need a license for 32 bit Windoze [until such time as they build an OPC Server for *nix - {read as 'forever'}]), but that's typically less than \$1,500 USD.

Where it's going to hurt is in bringing your facility into line - making all of your necessary OPC Devices ethernet friendly, or worse having to replace non-OPC devices with those that are OPC friendly. Gladly, virtually all name brand industrial controllers and many many devices such as cameras are OPC friendly if manufactured since 1995.

The good news is that you can build and install your S.E.E.R. Infrastructure now, and then add

your device I/O as you go along. For example, setup the Tank Model today with only level data... over the next 2 to 5 years budget for adding the necessary sensors for temperature, valve sensing (determine state - filling / emptying / etc...), and so on.

In the end, remember one thing - you're paying the same amount for your industrial equipment no matter what solution you choose to use. But wouldn't you like to save 0.5M USD on software, and use that money to buy more control and I/O? We sure would... and that is just one reason why S.E.E.R. Is free software.

Backbone

S.E.E.R. Is driven by `mod_openopc`, which utilizes plugins such as `syphon`, `fieldgate_dm3_reporter`, and others; it populates robust MySQL database tables built (by default) with the InnoDB engine capable of 'on-the-fly' row-locked backups (rather than table-locked offline backups). `mysql_innodb_backup` is the preferred method for DB dump (backup) and restore in event of catastrophic hardware failure. Also, you may be wondering about the data types that will be described in the next section (MODELS). Each data type listed is a MySQL compliant DB data type. For reference as to what each term means, please see the following article...

MySQL Numeric Types (this article is required reading):

<http://dev.mysql.com/doc/refman/5.0/en/numeric-types.html>

For info on plugins and utilities...

You can read more about `mod_openopc` here:

http://www.spinellicreations.com/spark/project_modopenopc.php

You can download `mod_openopc` here:

http://download.spinellicreations.com/mod_openopc/

You can read more on `syphon` here:

http://www.spinellicreations.com/spark/project_syphon.php

You can download syphon here:

<http://download.spinellicreations.com/syphon/>

You can read more on fieldgate_dm3_reporter here:

http://www.spinellicreations.com/spark/project_fieldgatedm3reporter.php

You can download fieldgate_dm3_reporter here:

http://download.spinellicreations.com/fieldgate_dm3_reporter/

You can read more on mysql_innodb_backup here:

http://www.spinellicreations.com/spark/project_mysqlinnoddbbackup.php

You can download mysql_innodb_backup here:

http://download.spinellicreations.com/mysql_innodb_backup/

Development Status

S.E.E.R. Is, as of the writing of this document, still in SUBVERSION. That means that no official "version 1" has been released. The current version 4.x is actually 0.4.x - so this begs the question why? What's missing? What doesn't work?

We're missing a few things:

- a complete user's guide / a complete admin's guide.
 - Negative effect = some things you will not have documentation for as of yet and will have to learn by word of mouth or training.
- cleanup of the "settings" page in the user interface.
 - Negative effect = it looks skewed depending on the user's access level, and generally is due for a rework.
- Code cleanup to the core model include files.

- Negative effect = none (to you, the user). From a developer's point of view, the code needs to be neatly typed, in order to allow easy reading by other developers and additions in the future.
- Generic non-Lactalis-ized artwork and options files in order to fully comply with GPL licensing and a proper public release.
 - Negative effect = none (to you, the user). If you're not a Lactalis facility, you'll simply have to strip out any reference to Lactalis in an OPTIONS files, and swap out the main banner logo with your own logo. From a legal standpoint, this is a minor, but morally important issue. Lactalis has been kind enough to not press the issue (as the software is still in subversion, and the time necessary to properly "clean" it would take away from time that could be used to write manuals [like this one!]). Expected "clean" date is first of the year, 2012.

You can read more on S.E.E.R. here:

http://www.spinellcreations.com/spark/project_seer.php

You can download S.E.E.R. here:

<http://download.spinellcreations.com/seer/>

CURRENTLY AVAILABLE MODELS

As of March, 2011; the following Core Models have been built, tested, and deployed - so they're ready for your use!

- *ATMOSPHERIC MODEL*

- allows tracking the TEMPERATURE, Barometric PRESSURE, and HUMIDITY of an environment. This may be a room, a cooler / freezer, a sauna, or any other controlled environment (computer server room, etc...).
- go ahead and run it on Spiral Refridgerators, Distribution Center Coolers, and Brine Canals.
- You do not have to use all available fields - for example, if you only wish to track temperature, then the Pressure and Humidity data is always registered as "zero"... no problem. You'll still get a nice report on your temperature over time.
- For any area you wish to use this model, you will require the following data points:

```
+-----+-----+
| MACHINENAME | varchar(30) |
| TEMPERATURE | float(7,2)  |
| HUMIDITY     | float(7,2)  |
| PRESSURE     | float(7,2)  |
+-----+-----+
```

- Also define (GLOBAL):
 - Unit of Measure (TEMPERATURE)
 - Unit of Measure (PRESSURE)
 - Unit of Measure (HUMIDITY)
 - Graph Range (TEMPERATURE) for example, "0 to 200"
 - Graph Range (HUMIDITY)
 - Graph Range (PRESSURE)
- You should break down your items by location (LOCAL). For example, if you have 5 Coolers, 2 Freezers, and 1 Outdoor area; then you may declare as follows:
 - Local_COOLERS

- COOLER_1 (or however you wish to distinguish)
- COOLER_2
- COOLER_3
- COOLER_4
- COOLER_5
- Local_FREEZERS
 - FREEZER_1
 - FREEZER_2
- Local_OUTDOOR
 - OUTDOOR_1
- Each Local group will have the following options which need to be declared:
 - The parameters from the 'Global' options, above, ONLY IF THEY ARE NOT THE SAME AS YOU ENTERED THERE! (you are allowed to override the defaults on a per-Local 'set' basis).
 - Snapshot time (in minutes); which dictates how far back (from the present time) the system will analyze averages when a user displays the "Current Status" of a room / machine.
 - A small drawing (by hand) of the general "layout of land", to allow users to understand where rooms or machines are located in relation to each other. Your administrator will then build a PNG image and integrate this into the system.
- Ultimately, it'll all come together like this...
 - [0] http://www.spinellcreations.com/spark/projects/seer/screencap_mctrl_6.png
 - [1] http://www.spinellcreations.com/spark/projects/seer/screencap_rpt_23.png
- WORKSHEET
 - Complete SEER_-_WORKSHEET_ATMOSPHERIC_MODEL.ods and provide to your INSTALLER.

- BULK MODEL

- track inventory of bulk materials, such as chemical storage, water, ingredient or other storage.
- For any area you wish to use this model, you will require the following data points:

```
+-----+-----+
| BULKNAME          | varchar(30) |
| INVENTORY_PERCENT | smallint(6) |
| INVENTORY_QUANTITY | int(11)     |
+-----+-----+
```

- Also define (GLOBAL):
 - none
- You should break down your items by location (LOCAL). For example, if you have 5 Chemical Tanks, 2 Ingredient Tanks, and 1 Water Tank; then you may declare as follows:
 - Local_CHEM
 - CHEM_Acid (or however you wish to distinguish)
 - CHEM_Caustic
 - CHEM_Foamer
 - CHEM_Diesel
 - CHEM_Oil
 - Local_INGREDIENT
 - INGREDIENT_Vineger
 - INGREDIENT_CornSyrup
 - Local_OTHER
 - OTHER_Water
- Each Local group will have the following options which need to be declared:
 - unit of measure for 'quantity'; such as "gal" for "gallons" or "lbs" for pounds.
 - Full point (percent); quantity in percent to consider an item as having

- adequate stock to maintain production for a while. (typically 60 to 75 %)
- Reorder point (percent); quantity in percent to consider an item as being necessary to reorder or replenish soon. (typically 25 to 50 %)
 - Empty point (percent); quantity in percent to consider an item as being critical / possibly insufficient to support any further production. (typically 5 to 15 %)
 - Appreciable change (percent); as many bulk product sensors are 'rough' in their accuracy (feedback can bounce a few percent back and forth, in either direction, without any actual inventory change), a change in inventory level must be greater than this value in order to be considered in calculations by the reporting system. (this is a trial and error value, however a 5 % value is often sufficient for sensors with accuracies equal to or less than 2 %... for 5 % accurate sensors, set this value to 10 %, and so on).
 - Appreciable change for Restock (percent); a change in inventory level greater than this value (in the positive direction) will be used to indicate that the item was restocked / refilled. This should be roughly 3 times the value of Appreciable change.
- And, you're finished product will look something like this...
- [0] http://www.spinellicreations.com/spark/projects/seer/screenshot_mctrl_7.png
 - [1] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_24.png
 - [2] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_25.png
- WORKSHEET
- Complete SEER_-_WORKSHEET_BULK_MODEL.ods and provide to your INSTALLER.

- CIP (Clean In Place) MODEL

- track the performance of various CIP systems throughout your facility, including what line circuits were washed, when they were washed, how much water was used during the wash, alarms during wash, and graph various critical control points (such as temperature, conductivity, and flow) throughout the wash cycle.
- includes regulatory compliance by allowing supervisors to digitally sign and comment on wash records.
- effectively replaces paper chart recorders!
- complies with Pasteurized Milk Ordinance when properly installed and administered
- For any area you wish to use this model, you will require the following data points:

CIPNAME	varchar(30)
STATUS	smallint(6)
LINE_BEING_CLEANED	smallint(6)
STEP	smallint(6)
SUPPLY_TEMP	smallint(6)
RETURN_TEMP	smallint(6)
SUPPLY_FLOW	smallint(6)
RETURN_CONDUCTIVITY	mediumint(9)
WATER_USAGE	mediumint(9)
WATER_TYPE	tinyint(4)

- Also define (GLOBAL):
 - Unit of Measure (FLOW) ex. "gallons / minute"
 - Unit of Measure (CONDUCTIVITY)
 - Unit of Measure (VOLUME)
 - Unit of Measure (TEMPERATURE)
 - Graph Range Low (TEMPERATURE)
 - Graph Range High (TEMPERATURE)
 - Graph Range Low (CONDUCTIVITY)
 - Graph Range High (CONDUCTIVITY)

- Graph Range Low (FLOW)
 - Graph Range High (FLOW)
 - Manual Record Entry Time Interval (minutes)
 - Scantime (how often - in seconds - do you wish to poll data)
 - Zerotime faults report as 1 minute duration? (yes or no - if a fault shows up for one data scan, but not afterward, then it effectively has a duration of 'zero' seconds... should we ignore this fault [choice = 'no'] or give this fault a duration of 1 minute [choice = 'yes'])
- You should break down your items by location (LOCAL). For example, if you have 5 CIP Systems in the Mozzarella Dept., 2 in the Ricotta Dept., and 1 in the String Cheese Dept.; then you may declare as follows:
- Local_Mozzarella
 - MOZZ_CIP1 (or however you wish to distinguish)
 - MOZZ_CIP2
 - MOZZ_CIP3
 - MOZZ_CIP4
 - MOZZ_CIP5
 - Local_Ricotta
 - RIC_CIP1
 - RIC_CIP2
 - Local_String
 - STRING_CIP1
- Each Local group will have the following options which need to be declared:
- a list of CIP system status, integer values from 0 to 32767, where 0 is always "Ready (Waiting)" (as in, System OK but not Running).
 - Common status values are:
 - 0 = active and ready
 - 1 = cip running
 - 2 = fill required
 - 3 = cip in manual hold
 - 4 = low water
 - 5 = low temperature
 - 6 = low conductivity
 - 7 = cip stopped

- If you notice, all states (except 1) indicate a fault or hold condition. If you wish to indicate other non-fault conditions, then do so at the BEGINNING. For example, you may declare status values 0 through 3 as non-fault values and then declare "anything over 3" as a fault. However you cannot mix-and-match. So make sure all your non-fault statuses are listed first!
- a list of available CIP line circuits, integer values from 0 to 32767, where 0 is always "none" or "not selected". As with the previous variable, you may list other states of selection, but all "real" lines must be last. For example, you may say that 0 is "none", 1 is "bad selection" and 2 is "idle"... then declare that anything "above 2" is a "real CIP line".
 - Common example:
 - 0 = none
 - 1 = invalid selection
 - 2 = receiving line A
 - 3 = receiving line B
 - 4 = silo #1
 - 5 = silo #2
 - ** real CIP lines > '1'
- a list of sequential CIP steps, integer values from 0 to 32767, where 0 is always "reset or off" (or similar). As with the previous variable, you may list other steps, but all "real" steps must be last. (we will forgo the example this time - you get the idea...)
 - Common example:
 - 0 = reset or off
 - 1 = fill tanks
 - 2 = pre-rinse
 - 3 = airblow
 - 4 = return to drain
 - 5 = acid wash
 - 6 = airblow to acid tank
 - 7 = acid recovery
 - 8 = rinse to drain
 - 9 = airblow to drain
 - 10 = caustic wash

- 11 = air blow to caustic tank
 - 12 = caustic recovery
 - 13 = rinse to drain
 - 14 = air blow to drain
 - 15 = return to drain
 - 16 = sanitize to drain
 - 17 = airblow to drain
 - 18 = airblow to drain
 - 19 = vent pressure
 - 20 = stop
- Now, perhaps you have 20 different line circuits on this hypothetical system. And let's say that some of those line circuits do NOT use all of these steps... for example, they may "skip" steps 5,6,7,8, and 9 because they do not have an acid wash. That's fine. But you still need to list each step (or "state of function") that the CIP system itself can be in. Just be sure to program your OPC device such that it displays the appropriate step sequence for each line being washed. It is not uncommon to have some line circuits start washing on "step 45" and end washing on "step 82" -- even though "step 45" is actually the first step of their particular wash cycle... it is the 45th state of existence for that CIP skid.
- a list of all the different types of water used on these CIP systems, in integer values from 0 to 32767, where 0 is always "none".
 - Common example:
 - 0 = none
 - 1 = treated water
 - 2 = city water
 - 3 = filtered water
 - 4 = treated and city water simultaneously
- S.E.E.R. Provides HMI functionality in the form of CIP Step Controls.
 - Note: this is OPTIONAL - and not really necessary at all unless you're in an environment where you wish to 'at will' have remote CIP HOLD/STEP access. Most installations will not bother with this.
 - If you choose to enable Step Controls, then you will also have to provide the following information:

- OPC Register (Write to force HOLD)
 - Value to Write to force HOLD
 - OPC Register (Write to force STEP)
 - Value to Write to force STEP
 - OPC Register (Disable machine operator from putting CIP into HOLD or STEPPING)
 - Value to Write to force DISABLE
- When you're all set, it'll look like this...
- [0] http://www.spinellicreations.com/spark/projects/seer/screenshot_mctrl_5.png
 - [1] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_18.png
 - [2] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_19.png
 - [3] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_20.png
 - [4] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_21.png
 - [5] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_22.png
- WORKSHEET
- Complete SEER_-_WORKSHEET_CIP_MODEL.ods and provide to your INSTALLER.

- CHECKWEIGHER MODEL

- here is a common, but troublesome issue: you are an industrial facility which utilizes product mass checks (via scale and/or high-speed checkweigher). Obviously, you require logs of the checkweigher's performance (analytics regarding the quality of your accepted vs. rejected items, as well as a list of rejects and when they occurred, among other things).
- S.E.E.R. Is a universal checkweigher reporting system; regardless of manufacturer or model, and tested at speeds in excess of 200 units per checkweigher per minute across multiple simultaneous checkweighers, S.E.E.R. Provides the reporting functionality to satisfy both your internal analysis and most localities weights-and-measures agencies.
- To accomplish this, S.E.E.R. Requires the syphon add-on, which communicates with any checkweigher that is capable of exporting "individual mass (weight) output" via Terminal Server. While some checkweighers may have this functionality built in over ethernet, many do not - however virtually any checkweigher will have (or have as an option) the ability to output individual mass via RS-232 / RS-485 or some other serial-protocol. In this case, you would then attach (typically via null-modem cable) a serial-to-ethernet terminal-server capable device (which typically can be had from reputable manufacturers for \$100 to \$300 USD per unit). S.E.E.R.'s preferred (only due to quality of operation) device is the Moxa NPort series, available from <http://www.moxa.com>. Although any such device will do the job nicely, provided it is run in "Terminal Server" or "TCP Server" mode. This will allow syphon to effectively utilize the telnet protocol to interface with the unit.
 - Example device: http://www.moxa.com/product/NPort_5100A.htm
 - Note: if your checkweigher outputs a different unit of measure than the one you wish to use (for example, your checkweigher outputs "grams" but your accepted unit of measure is "ounces"), syphon will translate this automatically for you. However you must indicate this for EACH checkweigher being used, otherwise it will be assumed that the unit of measure in which your checkweigher is outputting data is the same unit of measure you wish to view your reports in.

- For any area you wish to use this model, you will require the following data points:

+	-----+	-----+	
	RECIPE	varchar(30)	(for example "8 oz. Shampoo")
	TARGET	float(9,3)	(for example 8)
	DELTA_MIN	float(9,3)	(for example, 0.1 yields a 'min' of 7.9)
	DELTA_MAX	float(9,3)	(for example, 0.5 yields a 'max' of 8.5)
	TARE	float(9,3)	(for example, 0.25)
+	-----+	-----+	

- RECIPE indicates what you wish to call the 'preset' / 'routine' / or 'mode' of your checkweigher. Usually this corresponds to some product type or name.
- The following parameters should be set so as to IDENTICALLY MATCH those on your checkweigher (or group of checkweighers). Once set, the installer will typically WORM the settings database, meaning they cannot be changed (and should not be changed). If a recipe is updated on a checkweigher, then its NAME on the checkweigher should be changed (for example "8 oz. Shampoo" would become "8 oz. Shampoo R1" or something similar) and a NEW recipe preset created in S.E.E.R. Failure to do so, or attempting to modify an existing recipe within S.E.E.R. Will void compliance with weights-and-measures.
 - TARGET is the target mass (in whatever unit of measure you are using)
 - DELTA_MIN is the difference between the "minimum acceptable mass" and the TARGET... so if you wish to reject anything below 7.5 oz, and you're target is 8 oz., then set DELTA_MIN to 0.5
 - DELTA_MAX is the difference between the "maximum acceptable mass" and the TARGET... so to reject anything above 10 oz, and you're target is 8 oz., then set DELTA_MAX to 2.0 - be advised of the special case where you wish to accept all over-weight items and only reject underweight ones. In this case, set DELTA_MAX to "99999999" or some very large number.
- TARE is the mass of any packaging around your product / item. This determines the difference between GROSS and NET mass.
 - This is critical... if your checkweigher outputs NET MASS then you MUST set TARE to "0" (zero). If your checkweigher outputs GROSS MASS, then you MUST set TARE to whatever the TARE setting is within your checkweigher for each recipe!

- Also define (GLOBAL):
 - Default Unit of Measure (MASS)
 - Default Unit of Measure (LARGE MASSES) ...
 - such as monthly reports of production.
 - Default Unit of Measure (SCALE FACTOR)
 - SCALE FACTOR = value to multiply a measurement in MASS by if you wish to know its equivalent measurement in LARGE MASS.
 - Example... if MASS = "ounce" and LARGE MASS = "pound", then SCALE will be "1/16" or "0.0625"
 - Default Unit of Measure (UNIT)
 - how would you like to describe each unit? For example, valid answers are "bottle", "cup", "each", "box", "jar", etc...
- You should break down your items by location (LOCAL). For example, if you have 5 checkweighers in the Mozzarella Dept., 2 in the Ricotta Dept., and 1 in the String Cheese Dept.; then you may declare as follows:
 - Local_Mozzarella
 - MOZZ_SCALE1 (or however you wish to distinguish)
 - MOZZ_SCALE2
 - MOZZ_SCALE3
 - MOZZ_SCALE4
 - MOZZ_SCALE5
 - Local_Ricotta
 - RIC_SCALE1
 - RIC_SCALE2
 - Local_String
 - STRING_SCALE1
- Each Local group will have the following options which need to be declared:
 - if and only if you wish to override the default settings for Units of Measure and such, then you may handle your 'odd ball' checkweighers as follows, otherwise, the default settings will be used (which we talked about above).
 - Local Unit of Measure (MASS)
 - Local Unit of Measure (LARGE MASSES)

- such as for monthly reports of production.
 - Local Unit of Measure (SCALE FACTOR)
 - SCALE FACTOR = value to multiply a measurement in MASS by if you wish to know its equivalent measurement in LARGE MASS.
 - Example... if MASS = "ounce" and LARGE MASS = "pound", then SCALE will be "1/16" or "0.0625"
 - Local Unit of Measure (UNIT)
 - how would you like to describe each unit? For example, valid answers are "bottle", "cup", "each", "box", "jar", etc...
 - SNAPSHOT Time: defines how long (in minutes) the 'live view' will look back to analyze your most recent performance. For low-speed checkweighers, 10 to 30 minutes is typically good. For high-speed units, 1 to 10 minutes is much more informative of the units actual current performance.
- Once complete, your reports will look like this - although the CHECKWEIGHER MODEL is now complete, so they'll actually be more robust than these older screenshots (which were taken in early development)...

[0] http://www.spinellicreations.com/spark/projects/seer/screencap_mctrl_3.png

[1] http://www.spinellicreations.com/spark/projects/seer/screencap_mctrl_4.png

[2] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_16.png
note: yes, we know the plot is wrong. The image just hasn't been updated in a very long time. The current plots are CORRECT!

[3] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_17.png

- WORKSHEET

Complete SEER_-_WORKSHEET_CHECKWEIGHER_MODEL.ods and provide to your INSTALLER.

- Separation Pasteurization and Filtration MODEL

- the goal of this model was to effectively analyze "what is going in" and "what is coming out" of a dairy plant's production processes. In such an environment, it is not practical to look at "material produced per hour", or any other hardened metric of that nature. Rather, we want to know what resources are being used in a process, and then measure how much useable material is produced.
- As the name implies, this model can track SEPARATORS, CLARIFIERS, ULTRA FILTRATION SYSTEMS, REVERSE OSSMOSIS SYSTEMS, and HTST / PASTEURIZER SYSTEMS.
- For any area you wish to use this model, you will require the following data points:

Field Name	Data Type	Description
MACHINE_NAME	varchar(30)	
MACHINE_TYPE	tinyint(4)	
STATE	tinyint(4)	<i>ex. running, washing, idle.</i>
ALARM	smallint(6)	<i>ex. any of various faults</i>
TURBIDITY	tinyint(4)	<i>typically relative, in percent</i>
SOURCE	tinyint(4)	
DESTINATION1	tinyint(4)	
DESTINATION2	tinyint(4)	
SOURCE_FLOW	smallint(6)	<i>the flow rate</i>
DESTINATION1_FLOW	smallint(6)	<i>the flow rate</i>
DESTINATION2_FLOW	smallint(6)	<i>the flow rate</i>
SOURCE_TOTAL_FLOW	int(11)	<i>the rolling totalizer</i>
DESTINATION1_TOTAL_FLOW	int(11)	<i>the rolling totalizer</i>
DESTINATION2_TOTAL_FLOW	int(11)	<i>the rolling totalizer</i>
POWER	smallint(6)	<i>the power rate</i>
POWER_TOTAL	int(11)	<i>the rolling totalizer</i>
BOWL_MOTOR_RPM	smallint(6)	
BASELINEPRESSURE	mediumint(9)	
CONCENTRATION_RATIO	smallint(6)	
CONCENTRATION_VALVE_POSITION	tinyint(4)	
PRESSURE_RAW	smallint(6)	<i>past. - raw = differential</i>
PRESSURE_PASTEURIZE	smallint(6)	
TEMPERATURE_INLET	float(5,2)	
TEMPERATURE_PASTEURIZE	float(5,2)	<i>or "outlet"</i>
HRS_SINCE_CLEAN	smallint(6)	<i>hours since last wash</i>

CIP_STEP	tinyint(4)	<i>for self cleaning systems only</i>
CIP_TEMP	smallint(6)	<i>- self clean only</i>
CIP_FLOW	smallint(6)	<i>- self clean only</i>
CIP_WATER_TYPE	tinyint(4)	<i>- self clean only</i>
CIP_WATER_USAGE	mediumint(9)	<i>- self clean only</i>

+-----+-----+

- Also define (GLOBAL):

```

$SPFMODEL_MACHINE_TYPE[10] = "SEPARATOR";
$SPFMODEL_MACHINE_TYPE[11] = "SEPARATOR SELF-CLEANING";
$SPFMODEL_MACHINE_TYPE[20] = "CLARIFIER";
$SPFMODEL_MACHINE_TYPE[21] = "CLARIFIER SELF-CLEANING";
$SPFMODEL_MACHINE_TYPE[30] = "ULTRA FILTRATION";
$SPFMODEL_MACHINE_TYPE[31] = "ULTRA FILTRATION SELF-CLEANING";
$SPFMODEL_MACHINE_TYPE[40] = "REVERSE OSSMOSIS";
$SPFMODEL_MACHINE_TYPE[41] = "REVERSE OSSMOSIS SELF-CLEANING";
$SPFMODEL_MACHINE_TYPE[50] = "PASTEURIZER";
$SPFMODEL_MACHINE_TYPE[51] = "PASTEURIZER SELF-CLEANING";
/*      -- machine type descriptions */
/*      -- should not need to be changed, but you may wish to */
/*      translate the same values into your own language */

```

- If you use different terms to describe these machines, substitute as necessary.

- Default Units of Measurement...

- UM_MASS
- UM_VOLUME
- UM_WATER
- UM_TURBIDITY
- UM_FLOW
- UM_TEMPERATURE
- UM_POWER
- UM_POWER_RATE
- UM_ROTATIONAL_SPEED
- GRAPH_RANGE_TURBIDITY_LOW
- GRAPH_RANGE_TURBIDITY_HIGH

- GRAPH_RANGE_FLOW_LOW
 - GRAPH_RANGE_FLOW_HIGH
 - GRAPH_RANGE_PRESSURE_LOW
 - GRAPH_RANGE_PRESSURE_HIGH
 - GRAPH_RANGE_TEMPERATURE_LOW
 - GRAPH_RANGE_TEMPERATURE_HIGH
 - CONCENTRATION RATIO DIVIDED BY
-
- Totalizer meter rollover value
 - a totalizer ultimately must be reset at some point, so any totalizer that you program into an OPC device will AUTOMATICALLY reset itself to zero (and begin totalizing from zero onward again) after it hits this value.
 - this number must be large enough that it offers meaningful cycling, for example it should not rollover in less time than it takes to run one data scan.
 - typically, 30,000 is a good choice; it can be held by an unsigned integer register, is large enough to suit most applications, and will provide meaningful rollover.
-
- Manual record entry interval (MINUTES)
 - in the event of a power outage, for regulatory checked systems (where you are eliminating circular paper chart recorders) data should be written down by hand (see the manual entry form for only the few critical items necessary to log).
 - this is typically done every 5 to 15 minutes, but each plant is different.
 - INTERVAL FOR PASTEURIZERS
 - INTERVAL FOR ALL OTHER SYSTEMS
-
- Scantime (how often - in seconds - do you wish to poll data)
-
- Zerotime faults report as 1 minute duration? (yes or no - if a fault shows up for one data scan, but not afterward, then it effectively has a duration of 'zero' seconds... should we ignore this fault [choice = 'no'] or give this fault a duration of 1 minute [choice = 'yes'])

- You should break down your items by location (LOCAL). For example, if you have 5 machines in the Mozzarella Dept., 2 in the Ricotta Dept., and 1 in the String Cheese Dept.; then you may declare as follows:
 - Local_Mozzarella
 - MOZZ_PASTEURIZER1 (or however you wish to distinguish)
 - MOZZ_SEPARATOR1
 - MOZZ_SEPARATOR2
 - MOZZ_UF1
 - MOZZ_UF2
 - Local_Ricotta
 - RIC_CLARIFIER1
 - RIC_PASTEURIZER1
 - Local_String
 - STRING_UF1
- Each Local group will have the following options which need to be declared:
 - Each device's associated CIP source ("cleaned by East Cleaning System 1"), or, if it is self cleaning, list it as "SELCLEAN".
 - Cleaning warnings
 - WARN @ ___ hours since clean (typically 16)
 - ALARM @ ___ hours since clean (typically 20)
 - FAULT @ ___ hours since clean (typically 24)
 - An integer list of all possible states of all possible machines (in one list from 0 to 32767), which must include a CLEANING state.
 - An integer list of all possible alarms of all possible machines (in one list from 0 to 32767).
 - An integer list of all possible product sources (in one list from 0 to 32767).
 - An integer list of all possible product destinations (in one list from 0 to 32767).
 - If any machines are SELCLEAN, then provide a list of all possible CIP steps (in one list for all machines from 0 to 32767).
 - Denote (as with the CIP model) where the "real steps" start.
 - If any machines are SELCLEAN, then provide a list of all possible CIP water types (in one list for all machines from 0 to 32767).

- If any machines are SELFCLEAN, indicate whether you would like to use CIP STEP CONTROLS (see CIP MODEL for a description of how these work), and provide the following...
 - OPC DEVICE LEAF (HOLD)
 - OPC DEVICE LEAF (STEP)
 - OPC DEVICE LEAF (LOCKOUT)
 - VALUE TO WRITE TO DECLARE HOLD
 - VALUE TO WRITE TO DECALRE STEP
 - VALUE TO WRITE TO DECLARE LOCKOUT

- Indicate whether each machine has a Turbidity Sensor installed or not (in its drain system).
 - If 'yes', then decide if you wish to use Turbidity Alarm Acknowledgment or not.
 - If 'yes', then provide...
 - OPC DEVICE LEAF (ACK ALARM)
 - VALUE TO WRITE TO ACK ALARM

- Once complete, your reports will look like this...

[0] http://www.spinellicreations.com/spark/projects/seer/screenshot_mctrl_1.png

[1] http://www.spinellicreations.com/spark/projects/seer/screenshot_mctrl_2.png

[2] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_7.png

[3] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_8.png

[4] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_9.png

[5] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_10.png

[6] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_11.png

[7] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_12.png

[8] http://www.spinellicreations.com/spark/projects/seer/screenshot_rpt_13.png

[9] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_14.png

[10] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_15.png

*** we're aware the gantt chart is not spaced perfectly; this is an issue with the way the gantt chart function works (for all gantt charts) and will be addressed some point in the near future).*

- WORKSHEET

Complete SEER_-_WORKSHEET_SPF_MODEL.ods and provide to your INSTALLER.

*** NOTE - when completing this worksheet, there is an additional tab titled "help with table structure", which will advise you how to properly setup registers for the various machines, based upon what type of machine it is, etc... as not all machines will use all fields - in fact no machine uses all fields.*

- TANK (and Silo) MODEL

- the TANK MODEL is designed to provide essential SCADA control to any tank / receiving room supervision personnel, as well as provide historical and analytical data regarding tank inventory, tank alarm / status, and tank temperature.
- includes regulatory compliance by allowing supervisors to digitally sign and comment on all records.
- effectively replaces paper chart recorders!
- complies with Pasteurized Milk Ordinance when properly installed and administered
- For any area you wish to use this model, you will require the following data points:

Field Name	Data Type	Description / Example
SILONAME	varchar(30)	
STATE	smallint(6)	ex. Filling, emptying, blending, idle.
SOURCE	smallint(6)	ex. Receiving bay 1, silo 4, etc.
DESTINATION	smallint(6)	ex. Silo 2, pasteurizer 5, separator 1.
ALARM	smallint(6)	ex. Outlet valve manually open, etc.
PRODUCT	smallint(6)	ex. Raw milk, blend #35, sweet cream... (typically manually entered via SEER)
LEVEL_DENSITY	float(5,3)	typically mass per volume (lbs / gal) (typically manually entered via SEER)
LEVEL_PERCENT	float(6,3)	
LEVEL_MASS	mediumint(9)	ex. 100 pounds
LEVEL_VOLUME	mediumint(9)	ex. 12 gallons
TIME_SINCE_CLEAN	mediumint(9)	hours since last CIP sequence
AGITATOR_MODE	tinyint(4)	which agitation preset is in use
AGITATOR_LEVEL_ON	tinyint(4)	percent level agitator turn on (independent of agitator preset)
AGITATOR_LEVEL_OFF	tinyint(4)	percent level agitator turn off (independent of agitator preset)
AGITATOR_SPEED	tinyint(4)	current agitator speed (typically 0 to 100% of 60 hz)
TEMPERATURE	float(7,2)	

- Subsequently, if you wish to enable AGITATOR CONTROL via SEER, you will also need the following data points (separate table)...

```

+-----+-----+
| PRESETNAME | varchar(30) | agitation preset name (ex. "Group1_Blend_1-5")
| HIGHSPEED  | tinyint(4)  | agitation high speed setpoint (0 - 100% of 60 hz)
| LOWSPEED   | tinyint(4)  | agitation low speed setpoint (0 - 100% of 60 hz)
+-----+-----+

```

- Also define (GLOBAL):
 - Units of Measure
 - Unit of Measure DENSITY
 - Unit of Measure MASS
 - Unit of Measure TEMPERATURE
 - Graph Range TEMPERATURE LOW
 - Graph Range TEMPERATURE HIGH
 - What STATE will be used to indicate when a tank is in cleaning mode?
 - Typically this is "CIP" or "Cleaning" and must match (case sensitive) the value from the STATE list that you use to indicate cleaning mode.
 - Manual record entry interval (MINUTES)
 - in the event of a power outage, for regulatory checked systems (where you are eliminating circular paper chart recorders) data should be written down by hand (see the manual entry form for only the few critical items necessary to log).
 - this is typically done every 5 to 15 minutes, but each plant is different.
 - INTERVAL FOR ALL TANKS
 - Scantime (how often - in seconds - do you wish to poll data)
 - Zerotime faults report as 1 minute duration? (yes or no - if a fault shows up for one data scan, but not afterward, then it effectively has a duration of 'zero' seconds... should we ignore this fault [choice = 'no'] or give this fault a duration of 1 minute [choice = 'yes'])

- You should break down your items by location (LOCAL). For example, if you have 5 tanks in the Mozzarella Dept., 2 in the Ricotta Dept., and 1 in the String Cheese Dept.; then you may declare as follows:
 - Local_Mozzarella
 - MOZZ_TANK1 (or however you wish to distinguish)
 - MOZZ_TANK2
 - MOZZ_TANK3
 - MOZZ_TANK4
 - MOZZ_TANK5
 - Local_Ricotta
 - RIC_TANK1
 - RIC_TANK2
 - Local_String
 - STRING_TANK1
- Each Local group will have the following options which need to be declared:
 - List of TANK NAMES (in one list from 0 to 32767)
 - OPC Device Leaf corresponding to the register which indicates each tank's current PRODUCT.
 - OPC Device Leaf corresponding to the register which indicates each tank's current PRODUCT DENSITY.
 - An integer list of all possible ALARMS of all possible tanks (in one list from 0 to 32767)
 - An integer list of all possible PRODUCTS in all possible tanks (in one list from 0 to 32767)
 - An integer list of all possible STATES of all possible tanks (in one list from 0 to 32767), which must include a CLEANING state (and that must correspond to the GLOBAL option you chose to indicate cleaning!)
 - An integer list of all possible SOURCES of all possible tanks (in one list from 0 to 32767)
 - An integer list of all possible DESTINATIONS of all possible tanks (in one list from 0 to 32767)


- Cleaning warnings
 - WARN @ ___ hours since clean (typically 48)
 - ALARM @ ___ hours since clean (typically 64)
 - LOCKDOWN (or FLAG) @ ___ hours since clean (typically 72)
- Should these tanks use a CLEANING LOCKDOWN system?
 - 'Yes' or 'No'
 - If 'Yes', then your OPC Device will tally hours since clean, and when this exceeds "LOCKDOWN @ ___ hours since clean", your OPC Device will automatically disable the function of the tank inlet valve (or inlet AND outlet valve). At this point the tank will be un-usable, and all product trapped inside.
 - Note: you should be tallying hours since clean and reporting that to S.E.E.R., even if you have no lockout function present, regardless.
 - You must then code a bit in your OPC Device, which may be written to by SEER, which when written to with the proper value will allow the LOCKDOWN to be released. SEER allows users with only the highest level credentials (Quality Control / Super User / Administrators) to unlock a locked tank.
 - A list of OPC Device Leafs which correspond to the register for each tank used to initiate a release of lockdown.
 - What value should be written to any of the registers to release that tank? (must be same value for all tanks, for example, writing a "1" to N7:10 will release tank 1, N7:11 will release tank 2, and so on and so forth.
- Would you like to enable AGITATION CONTROL for these tanks?
 - 'Yes' or 'No'
 - If 'Yes', then you should have a multi-preset agitation system in place already which uses 2 speeds per preset (high and low), and is configurable via an OPC Device.
 - Which of the tanks in this model would you like to use AGITATION CONTROL on (you can choose to use all or just some - you do not have to use all).
 - If you have variable frequency drives in service for this task, list the drive manufacturer and model for each tank.

- If these drives have ethernet-enabled web-servers that show the drive's current status, then list the IP address (and port if necessary) of each drive's web server.
- For each tank or silo, indicate the following...
 - OPC Device Leaf corresponding to register where AGITATION PRESET (current running preset) is stored. This leaf must be writable so that SEER can overwrite it with the user's choice when a user chooses to change the PRESET.
 - OPC Device Leaf corresponding to AGITATOR TURN ON LEVEL LOW (in percent). This leaf must be writable to allow SEER to push user's modifications to register.
 - OPC Device Leaf corresponding to AGITATOR TURN ON LEVEL HIGH (in percent). This leaf must be writable to allow SEER to push user's modifications to register.
- An integer list of all possible AGITATION PRESETS (INSTALLERS ONLY - read this as "AGITATOR STATE in the Local Options file") of all possible tanks (in one list from 0 to 32767)
 - typically one of these presets (the first one, 0) will be "OFF" or "OUT OF SERVICE", allowing the user to be able to de-select agitation all together if they wish.
- An integer list of all possible AGITATION PRESET COMMANDS (in one list from 0 to 32767, and corresponding to the PRESETS) that will command an OPC Device to put a particular tank into the chosen PRESET state...

for example:

```
preset[0] = raw milk
preset[1] = blends #1 - #5
preset[2] = blends #6 - #8
preset[3] = whey
preset[4] = cream

command[0] = 1
command[1] = 2
command[2] = 4
command[3] = 8
command[4] = 16
```



in this example, we see that when a COMMAND value of "4" is written to the specified OPC Device Leaf (of whatever silo or tank we are working with), then that tank will be in the the 'state' of PRESET #2 (which is "blends #6 - #8")

- MIN / MAX values for AGITATOR TURN ON (what is the lowest value, in percent, that a user may set as the 'turn on' point for a tank's agitator - and the highest)
 - MIN / MAX values for AGITATOR TURN OFF (what is the lowest value, in percent, that a user may set as the 'turn off' point for a tank's agitator - and the highest)
 - MIN / MAX values for AGITATOR SPEED (what is the lowest speed, [usually in percent] that a user may choose to use in making an AGITATOR PRESET - and the highest)
 - when using standard AC VFD's and squirrel cage motors, the typical choices here are 20% low and 80% high to prevent motor and drive damage.
 - A list of OPC Device Leafs that correspond to the speed setting location for each AGITATION PRESET's high and low speed setting.
 - For example, let's say you have a preset called "milk 1". Now "milk 1" will have a low speed and a high speed, and let's say we store the value for high speed at N7:10 and the value for low speed at N7:11
- Once complete, your reports will look like this...
- ** note, these reports have been improved upon since these screenshots were taken, and now indicate even more information.

[0] http://www.spinellicreations.com/spark/projects/seer/screencap_mctrl_0.png

[1] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_0.png

[2] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_1.png

[3] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_2.png

[4] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_3.png

[5] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_4.png

[6] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_5.png

[7] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_6.png

** we're aware the gantt chart is not spaced perfectly; this is an issue with the way the gantt chart function works (for all gantt charts) and will be addressed some point in the near future).

- WORKSHEET

Complete SEER_-_WORKSHEET_TANK_MODEL.ods and provide to your INSTALLER.

- *W.A.R.R.I.O.R. Overall Equipment Efficiency*

- the Workplace Authenticated Runtime Resource Input and Output Reporter is, in and of itself, both a functional O.E.E. / T.E.E.P. (Overall Equipment Efficiency / Total Equipment Effective Productivity) module for S.E.E.R. - and a proof of concept that such systems do not have to be complicated, can store inordinately large amounts of data (have a very long history - 4 years was the design goal, 10 years is not unreasonable given enough storage space), and can run in a fast paced industrial production facility reliably.
- That said, W.A.R.R.I.O.R. Is the most difficult model to understand if you are unfamiliar with S.E.E.R.'s inner workings. It is highly recommended that it not be the first model which is deployed in a facility, as the learning curve is a bit steeper, and the potential aggravation much greater than with other models. This is not to detract from the robustness or efficacy of the model, but simply to state that level of understanding of both one's own industrial processes and machines, as well as one's understanding of S.E.E.R., is much greater than is required for the other models (with respect to setup and install - daily use / operation is as simple as any other model).
- Expected early 2012: Administrators may select a GLOBAL option, which will automatically cause integration of the W.A.R.R.I.O.R. Module with Lactalis's Corporate Industrial Labeling Application (tentatively named Bar-tender). This will allow an operator to simply enter their production information on one screen (an expanded upon version of the current W.A.R.R.I.O.R. Primary operator interface within S.E.E.R.) and control both W.A.R.R.I.O.R. And Bart-ender simultaneously from a single interface. Bar-tender's licensing information is unknown at this time, other than that it is property of Lactalis, USA - any desire to use it outside of Lactalis should be addressed to the appropriate contact personnel within Lactalis (to request licensing, etc., as it may not even be available for third party use). Regardless, standalone use of W.A.R.R.I.O.R. Will always remain; and we'll simply add options to 'integrate' with one or more external apps (such as Bar-tender). So, if you've got a labeling app and want to integrate it, go ahead! Just don't break standalone functionality (or the ability to add multiple integrations based upon admin selection) or your code revisions will be rejected when submitted back upstream.

- For any area you wish to use this model, you'll require the following data points:

```

+-----+-----+
| MACHINE_NAME      | varchar(30) |
| OPERATOR          | mediumint(9) |
| STATE             | tinyint(4) |
| ALARM             | smallint(6) |
| CORRECTIVE_ACTION | tinyint(4) |
| PACKAGE_CLASS     | mediumint(9) |
| PACKAGES_PER_CYCLE | smallint(6) |
| CYCLES            | bigint(20) |
| JOB_NUMBER        | bigint(20) |
| SCHEDULE_NUMBER   | varchar(35) |
+-----+-----+

```

**STORAGE ONLY*
**YOU PROVIDE!*
**YOU PROVIDE!*
**STORAGE ONLY*
**YOU PROVIDE!*
**YOU PROVIDE!*
**SIZE 1*
**SIZE 1 *STORAGE ONLY*
**INTERNAL - DOES NOT XMIT TO OPC DEVICE*

- Items marked STORAGE ONLY:

- These are items that S.E.E.R. Will fill in for you. S.E.E.R. Simply needs some place to store this information on your OPC Device. So make sure that the register is unused by anything else, and is read/write accessible.

- Items marked SIZE 1:

- These should be a register capable of holding an integer of minimum size 1×10^{18} (for an Allen Bradley PLC, this would be a Floating Point register [such as F8:1, or F8:7, etc...], as that register type can handle this size number).

- Items marked INTERNAL - DOES NOT XMIT TO OPC DEVICE:

- Is handled by W.A.R.R.I.O.R. Internally, so from the OPC Device programmer's point of view, you can ignore it... don't setup a register for it or anything; pretend it doesn't exist (even though it does).

- Items marked YOU PROVIDE!:

- this is the feedback your machine is going to constantly generate, and will be read by S.E.E.R. In order to determine your efficiency. You will have to write logic within your OPC Device in order to generate this information, and you MUST comply with the following keys...

- STATE:

- integer value of 0 = MACHINE IS IDLE or UNKNOWN CAUSE OF DOWNTIME
 - if machine has not cycled in "X" minutes (where you choose a value that is reasonable... we're partial to 2 minutes as a good starting point), then let's assume it is down but since we didn't detect any faults we don't know if it is down because it is scheduled down or because it is broken down... (this is where a user reading a report will refer to the CORRECTIVE ACTION for more information.

- Integer value of 1 = MACHINE IDLE AND THE FAULT WAS DETECTED
 - push this state when your OPC Device has picked up a hard fault on your machine (for example, "main motor aux input dropped out" or anything else that is known for certain will cause the machine to STOP).
 - Indicate the cause of the downtime by pushing the appropriate integer value into the ALARM register.

- Integer value of 2 = MACHINE UNDER MAINTENANCE SUPERVISION, DIAGNOSTIC OR REPAIR.
 - You do NOT choose when to push this state. Instead, declare some register or bit on your OPC Device, and when S.E.E.R. Writes a pre-determined value to that bit (for example, write "1" to B3:0/1), your OPC Device will push a value of "2" into the STATE register

FOR AS LONG AS THE TRIGGERING REGISTER IS SET TO THE PRE-DETERMINED VALUE (usually "1").

- Additionally, if STATE = 2, then force the ALARM register value to be "1" (which is indicated as "ALARM: In Maintenance Mode").
 - Additionally, if STATE = 2, then inhibit increment of cycle count. That is to say, do not allow your CYCLE tally count to increase for the entire duration of time that STATE = 2. This allows Maintenance personnel to run a machine while empty and not have the cycles count toward production. Also, it allows the time a machine is in maintenance to be omitted from efficiency calculations.
 - OPTION! - depending on your facility's policy, you may CHOOSE to consider all "Maintenance Mode" time as DOWNTIME. In order to do this, you will have to add a rung of logic to your OPC Device as follows...
if STATE = 2, set CORRECTIVE ACTION = 1, which is indicated "Maintenance Mode".
 - Otherwise, do NOT alter the corrective action based upon Maintenance Mode... just leave it alone!
- Integer value of 3 = MACHINE RUNNING WITH WARNING (OR MINOR FAULT)
- If the machine is running, but you pick

up a warning, such as "low oil" or "low bags" or "low product", and while this warning is something the operator and staff should be warned about, it HAS NOT caused the machine to stop, then indicate the state as "3" (RUNNING WITH WARNING).

- Indicate the cause of the warning by pushing the appropriate value into the ALARM register.

- Integer value of 4 = MACHINE RUNNING - NO FAULTS!

- If and ONLY IF the machine is running with no faults (STATE Key #4), then the following MUST be adhered to...
- 1) make sure ALARM value goes to '0' (hey, you said there weren't any faults or warnings, so there better not be any showing up!)
- 2) set CORRECTIVE ACTION to '0', do not allow an operator to enter a corrective action if nothing is wrong!!

- ALARM:

- integer value of 0 = no faults or warnings present
- integer value of 1 = MAINTENANCE MODE
- integer value of 2 = your first custom alarm
- integer value of 3 = your second custom alarm
- and so on and so forth, up to integer value of 32767 if you need

** ALL ALARM VALUES ARE GENERATED BY THE OPC Device AND READ BY S.E.E.R.

- CORRECTIVE ACTION:

- integer value of 0 = no faults or warnings present
- integer value of 1 = MAINTENANCE MODE
- integer value of 2 = your first custom action
- integer value of 3 = your second custom action
- and so on and so forth, up to integer value of 32767 if you need

** ONLY CORRECTIVE ACTION VALUES '0' and '1' ARE GENERATED BY THE OPC Device... THE REST ARE GENERATED BY S.E.E.R., SO BE SURE TO PROGRAM YOUR OPC DEVICE SUCH THAT IT ONLY WRITES VALUES TO THE CORRECTIVE ACTION REGISTER IN THE TWO SPECIFIC CASES WHEN IT SHOULD BE DOING SO... THE REST OF IT TIME IT SHOULD ALLOW S.E.E.R. TO WRITE WHATEVER IT WISHES TO THIS REGISTER.

- CYCLES:

- this register is to be tallied by the OPC Device, and should increment up each time a 'full package' is detected. For example, if you are making cases of mouth wash, at 12 bottles per case, then each time a CASE is detected the CYCLE register should be incremented up by '1' (not 12!).
- this register must be RESET to ZERO by the OPC Device any time the S.E.E.R. Sends a reset signal to a preset OPC Device register. For example, your if B3:0/1 is our 'reset' bit, then your OPC Device should monitor B3:0/1, and if it is set to "1" (or some other pre-defined value), then your OPC Device should reset the CYCLE count to '0', and then CLEAR the reset register (B3:0/1).

- PACKAGE CLASS:

- this register will correspond with a list of package types. For example, let's say you have 5 different package types and declare for them integer values from 0 to 32767:
 - 12pack_shrink_wrap = value 0
 - 12pack_box = value 1
 - 12pack_display_case = value 2
 - 18pack_box = value 3
 - 24pack_box = value 4

** So, anytime the machine is running "12pack_display_case", your OPC Device must detect that (whether it be from an operator HMI, a toggle switch, or whatever), and write a value of "2" to the PACKAGE CLASS register.

** This will also allow S.E.E.R. To know how many 'eaches' or 'units' are in a single package.

- PACKAGES_PER_CYCLE

- if your machine is able to detect each package produced, then you may set this value to "1" always. Otherwise, in the event that your machine is not able to detect packages, but, instead is able to detect how many times it has cycled... then (depending on what product you're running) your OPC Device should set this register to the NUMBER OF PACKAGES PRODUCED BY THE MACHINE IN ONE CYCLE.
- For example, let's say you have a cup-filling machine that can produce up to 12 cups per cycle. You do not have the ability to detect individual cups. Now... for product "A", you run the machine at half capacity, filling only half of its banks, and product 6 cups every time the machine cycles. When you run product "B", you run the machine at full capacity, filling all the banks, and eject 12 cups every time the machine cycles. Your OPC Device must write a value of "6" to the PACKAGES PER CYCLE register when running product "A", and a value of "12" when running product "B".

- Also define (GLOBAL):

- Current Status Time Window (minutes)
 - how far back do you want to examine when displaying "current status" statistics to operators?
 - typically 2 to 4 hours is good.
- Define the CATEGORIES (integer list from 0 to 32767) that you wish to group DOWNTIME and UNSCHEDULE TIME by...
 - a typical plant will use values similar to the following:

- 0 = unexplained (required)
- 1 = other
- 2 = breakdown
- 3 = waiting for product or supplies
- 4 = machine adjustment during run
- 5 = product or machine changeover
- 6 = product / packaging handling issue
- 7 = scheduled break or lunch break
- 8 = scheduled down / not scheduled to run / or cleaning

** NOTE, later, we will take the entire list of ALARMS and CORRECTIVE ACTIONS that you provide for each Local 'set', and link each one of them to one of these CATEGORIES. That way, we can say (for example) that 40 minutes down due to ALARM '3' (failed motor), and 20 minutes down due to ALARM '9' (piston failed to reach reed switch) can be summarized as "60 minutes down due to CATEGORY '2' - breakdown".

- You should break down your items by location (LOCAL). For example, if you have 5 lines in the Mozzarella Dept., 2 in the Ricotta Dept., and 1 in the String Cheese Dept.; then you may declare as follows:

- Local_Mozzarella
 - MOZZ_LINE1 (or however you wish to distinguish)
 - MOZZ_LINE2
 - MOZZ_LINE3
 - MOZZ_LINE4
 - MOZZ_LINE5
- Local_Ricotta
 - RIC_LINE1
 - RIC_LINE2
- Local_String
 - STRING_LINE1

- Each Local group will have the following options which need to be declared:
 - An integer list of all possible SHIFTS of work time of all possible lines (in one list from 0 to 32767), along with START and END hours (24 hour clock) for each.

- typically...
 - shift 0 = FIRST // start @ 06:00 / end at 15:00
 - shift 1 = SECOND // start @ 15:00 / end at 23:00
 - shift 3 = THIRD // start @ 23:00 / end at 06:00
- Define OPC Device registers for each line as follows:
 - Operator register
 - Corrective Action register
 - Job_Number register
 - Cycle_Reset register
- Define a value to write to any CYCLE RESET register in order to trigger your OPC Device to clear (zero / reset) its CYCLE COUNT:
 - typically "1"
- Unit of Measure PACKAGE UNIT:
 - when you break down a package (whether it be a box or a shrink-wrap or a carton...) what are the individual units called?
 - typically "eaches" or "units".
- Unit of Measure PACKAGE UNIT MASS:
 - when you weigh one of the individual units (as described above), what do you want the weight to be in?
 - typically "pounds" or "kilograms"
- An integer list of all possible ALARMS of all possible machines (in one list from 0 to 32767)
 - 0 is required to be "none"
 - 1 is required to be "maintenance mode"
 - all others from 2 to 32767 may be whatever you choose
 - be sure to define a CATEGORY (from the GLOBAL options, above) for each ALARM)
- An integer list of all possible CORRECTIVE ACTIONS of all possible machines (in one list from 0 to 32767)
 - 0 is required to be "none entered"
 - 1 is required to be "maintenance mode"
 - 2 is required to be "scheduled down or cleaning"
 - 3 is required to be "operator on break or lunch"
 - all others 4 through 32767 may be whatever you choose.

- Examples include:
 - 4 = waiting on product
 - 5 = product changeover
 - 6 = machine or tooling changeover
 - 7 = machine adjustment during run
 - 8 = breakdown - other.
- be sure to define a CATEGORY (from the GLOBAL options, above) for each CORRECTIVE ACTION.
- An integer list of all possible PACKAGE CLASSES of all possible machines (in one list from 0 to 32767)
 - 0 is required to be "unipac" (this means 1 unit or each per package)
 - all others from 2 through 32767 may be whatever you wish
 - be sure to include the UNIT COUNT (number of units) per each PACKAGE CLASS type.
 - be sure to include the UNIT MASS (mass of each unit) per each PACKAGE CLASS type (and be sure this mass is in the unit of measure defined by 'Unit of Measure PACKAGE MASS').
 - be sure to include a TARGET PRODUCTION RATE for each PACKAGE CLASS.
 - ** NOTE the TARGET PRODUCTION RATE is defined as "UNITS (or 'EACHES') PER HOUR" !! If your company's figures are listed by day or by shift, then divide accordingly!! If your company's figures are listed by poundage (mass produced over a time period), then simply divide the mass value by the "UNIT MASS" value, and you will have the TARGET PRODUCTION RATE (per hour) in UNTIS (or 'EACHES').
- Provide a JOB LIST:
 - this is a list of all Job / Task / or Resource Numbers, and a brief description of what each one of them is.
 - For example...

- JOB	DESC
- -----	-----
- XA567	12 pack, 3 oz. Toothpaste, 'Crest' brand
- 98C27869	6 pack, 12 oz. Beer, 'Budweiser' brand
- etc...	etc...

- Once complete, your reports will look like this...
 - ** note, these reports have been improved upon since these screenshots were taken, and now indicate even more information.

[0] http://www.spinellicreations.com/spark/projects/seer/screencap_mctrl_8.png

[1] http://www.spinellicreations.com/spark/projects/seer/screencap_mctrl_9.png

[2] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_26.png

[3] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_27.png

[4] http://www.spinellicreations.com/spark/projects/seer/screencap_rpt_28.png

- WORKSHEET

Complete SEER_-_WORKSHEET_WARRIOR.ods and provide to your INSTALLER.