

# IBM Sterling CPQ version 9.4

Sub Model Enhancements, Upgrade BOM and Globalization

Dinup P Pillai

[dinup.pillai@in.ibm.com](mailto:dinup.pillai@in.ibm.com)



---

## Agenda

- Sub-model support via Sterling Configurator XAPIs
- Upgrade Bill of Material
- Enhanced integration with Websphere Commerce
- Extensibility Improvements in Sterling Configurator

## Sub-model Overview and types of Sub-model

- Sub-model allows the modeler to build models inside VM in a more modular way by factoring reusable component/rule out of a huge model into a set of smaller sub-models.
- As of CPQ v9.3, this feature is exposed only in Configurator application.
- Sterling Configurator APIs have been enhanced to support sub-model configuration.
- Configurator/VM supports two types of sub-model configuration:
  - Sub-model punch-in punch-out
  - Dynamic Instantiation

## Sub-model punch in punch out

- Configurator transitions from one model to another and evaluate smaller set of information at a time.
- Actions:
  - punch into a sub-model
  - perform sub-model configuration
  - return from sub-model to the parent model
- The parent model picks, including the derived states, is made available for evaluation inside the child models.
- When done with the child configuration, the resulted configuration is returned to the parent model.
- Generate configuration output (BOM) with the aggregate of parent and child configurations.

## Dynamic Instantiation

- Dynamic instantiation provides a way to allow users to configure products on the fly while avoiding the need to create option items for each possible product configuration in your model.
- Actions:
  - instantiate 'n' number of child models
  - punch into a sub-model
  - perform sub-model configuration
  - return from sub-model to the parent model

## APIs: Design & details

- Focus to simplify the creation of API input to improve API usability.
- Input to all Configurator APIs can be constructed by extracting certain sections/elements of the output from the previous API call.
- As configurator is a stateless engine:
  - The current state of the configuration is passed as input.
  - The derived state of the configuration is returned as output.
- Sub-model APIs re-use the API input/output structure of processConfigurationPicks API with some additional sub-model specific attributes/elements.

---

## Sub Model APIs

- New Sub-model APIs:
  - `instantiateSubModel`
  - `punchIntoSubModel`
  - `returnFromSubModel`
- The following APIs were enhanced to support sub-models:
  - `processConfigurationPicks`
  - `generateConfigurationBOM`
  - `extractPicksFromConfigurationBOM`

## instantiateSubModel API

```
<InstantiateSubModel
  Country="Required" Currency="Required"
  Language="Required" ModelDepth="Required"
  OrganizationCode="Required" Path="Required">
  <Picks>
    <Pick Item="" ItemId="" Quantity="" Type="" Value="" />
  </Picks>
  <AddSubModel Item="" Quantity="" />
  <CopySubModel Item="" Quantity="" />
  <RemoveSubModel Item="" />
</InstantiateSubModel>
```



## punchIntoSubModel API

- The API to punch into the sub-model linked to the current model. The call is made in the context of the current model. At the end of call, the context transitions into the child.
- The API returns sub-model configuration (picks) & parent configuration (configuration BOM & input properties).
- API input elements/attributes:
  - PunchInItem - The path of the option item on which the punch in action is invoked. The API punches into the sub-model associated with this option item.  
`<PunchIntoSubModel ..... PunchInItem="">`
  - Picks - The picks for the current model. Also contains the configuration and state of the sub-model attached to the option item.  
`<Pick Item="" ItemId="" Quantity="" Type="" Value="">  
    <NestedConfiguration>  
        <ConfiguratorBOM/>  
    </NestedConfiguration>  
    <NestedProperties>  
        <properties/>  
    </NestedProperties>  
</Pick>`

## punchIntoSubModel API (cont)

```

<PunchIntoSubModel Country="Required" Currency="Required" Language="Required" ModelDepth="Required"
  OrganizationCode="Required" Path="Required" PunchInItem="Required">
  <Picks>
    <Pick Item="" ItemId="" Quantity="" Type="" Value="">
      <NestedConfiguration>
        <ConfiguratorBOM ModelName="Required" Version=""
          containerOnly="" errors="">
          </ConfiguratorBOM>
        </NestedConfiguration>
        <NestedProperties>
          <properties>
            <property name="" path="" type="" value=""/>
          </properties>
        </NestedProperties>
      </Pick>
    </Picks>
    <ParentConfigurations>
      <ParentConfiguration ModelDepth="" PunchInItem="" SubModelReturn="" SubModelValidate="">
        <InputProperties>
          <Property Name="" Path="" Type="" Value=""/>
        </InputProperties>
        <ConfiguratorBOM ModelName="Required" Version=""
          containerOnly="" errors="">
          </ConfiguratorBOM>
        </ParentConfiguration>
      </ParentConfigurations>
      <InputProperties>
        <Property External="" Name="" Path="" Type="" Value=""/>
      </InputProperties>
    </PunchIntoSubModel>
  
```

## returnFromSubModel API

```
<ReturnFromSubModel Abandon="" Country="Required" Currency="Required" Language="Required"
ModelDepth="Required" OrganizationCode="Required" Path="Required">
  <Picks>
    <Pick Item="" ItemId="" Quantity="" Type="" Value="">
      <NestedConfiguration>
        ....
      </NestedConfiguration>
      <NestedProperties>
        ....
      </NestedProperties>
    </Pick>
  </Picks>
  <ParentConfigurations>
    ....
  </ParentConfigurations>
  <InputProperties>
    ....
  </InputProperties>
</ReturnFromSubModel>
```

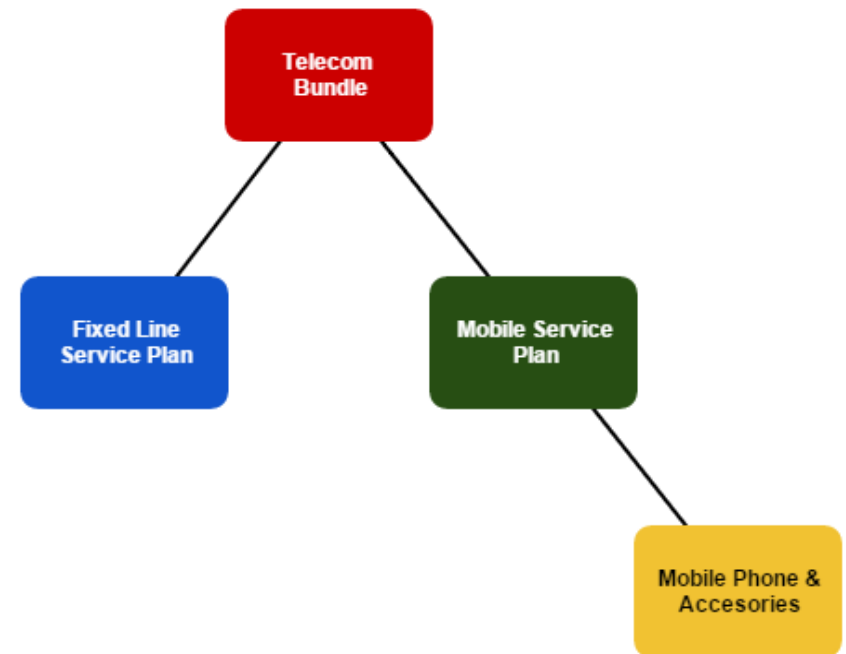
---

## processConfigurationPicks & generateConfigurationBOM API

- Input to these existing APIs have been re-structured to accommodate nested configuration, parent configuration & input properties.
- processConfigurationPicks API re-constructs the state of the current model after evaluating the picks against the rules in the context of both parent and child configurations/properties.
- generateConfigurationBOM API can be invoked from:
  - Parent/Root model: Constructs the ConfiguratorBOM of the current model and nested child configurations.
  - Sub-model: Constructs the ConfigurationBOM of the sub-model and nested child configurations. The parent configurations are ignored.

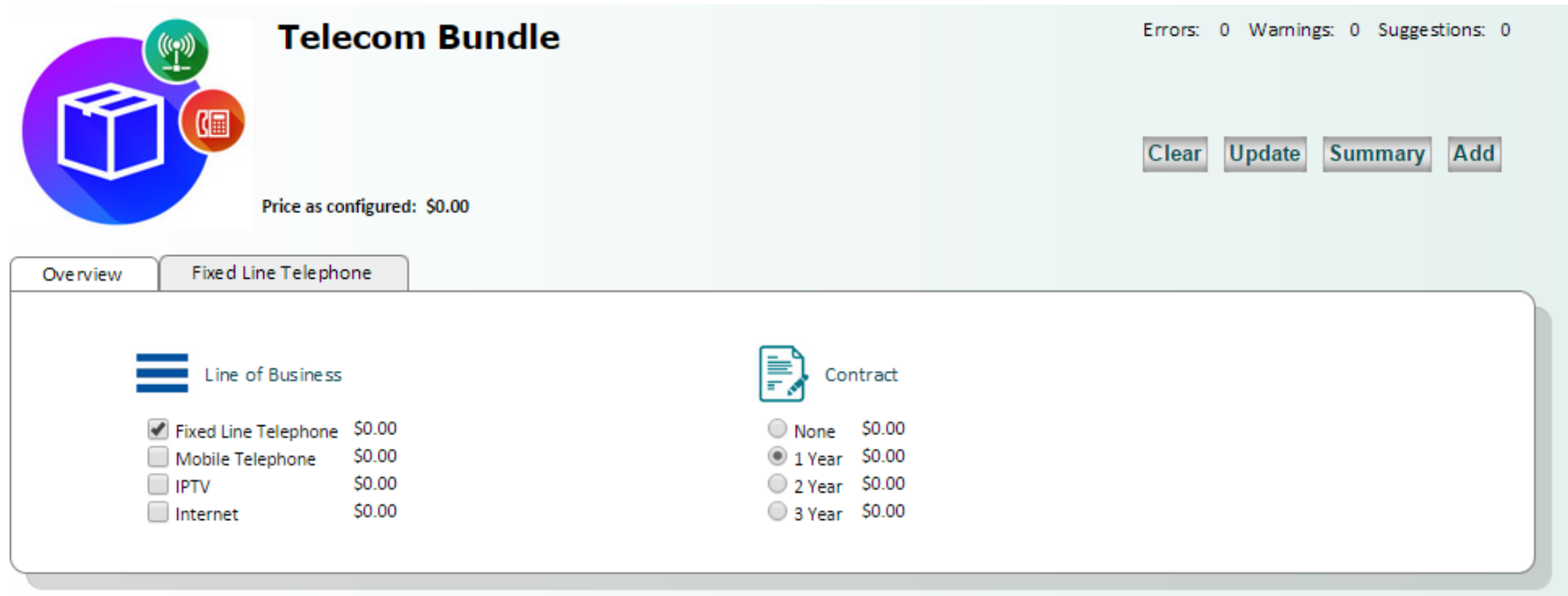
## Sub-model scenario

- Telecom Bundle contains various lines of business:
  - Fixed Line Service Plan
  - Mobile Service Plan & so on
- User can configure one or more accounts for a chosen service plan.
- Service Plan options are modeled as sub-models.
- Within Mobile Service Plan, Mobile phones and its accessories can be added to the selection. These options are modeled as sub-model of Mobile Service plan.



## Sub-model scenario (cont)

User wants to configure a Telecom Bundle



**Telecom Bundle** Errors: 0 Warnings: 0 Suggestions: 0

Price as configured: \$0.00

Clear Update Summary Add

Overview Fixed Line Telephone

Line of Business

<input checked="" type="checkbox"/>	Fixed Line Telephone	\$0.00
<input type="checkbox"/>	Mobile Telephone	\$0.00
<input type="checkbox"/>	IPTV	\$0.00
<input type="checkbox"/>	Internet	\$0.00

Contract

<input type="radio"/>	None	\$0.00
<input checked="" type="radio"/>	1 Year	\$0.00
<input type="radio"/>	2 Year	\$0.00
<input type="radio"/>	3 Year	\$0.00

### Flow:

1. processConfigurationPicks (existing API) – Load the Model
2. processConfigurationPicks – to add “Fixed Line Telephone” & “1 year” contract.

## Sub-model scenario (cont)

User wants to create one or more accounts for a “Fixed Line Service Plan”

### Flow:

1. processConfigurationPicks – to add “Service Plan 2”
2. **instantiateSubModel** (new API) – to create one or more dynamic option item(s) that holds the service plan (sub-model) configuration.

**Telecom Bundle** Errors: 0 Warnings: 0 Suggestions: 0

Price as configured: \$0.00

Overview Fixed Line Telephone

Fixed Line Telephone

<input type="radio"/> None	\$0.00
<input type="radio"/> Service Plan 1	\$0.00
<input checked="" type="radio"/> Service Plan 2	\$0.00
<input type="radio"/> Service Plan 3	\$0.00

Service Plan 2

Add

**Telecom Bundle** Errors: 0 Warnings: 0 Suggestions: 0

Price as configured: \$0.00

Overview Fixed Line Telephone


```
<InstantiateSubModel Country="US"
  Currency="USD" Language="en" ModelDepth="0"
  OrganizationCode="matrix" Path="Telecom/TelecomBundle">
  <Picks>
  .....
  </Picks>
  <AddSubModel Item="TelecomBundle.Fixed Line Telephone.Service Plan 2.Plan"/>
</InstantiateSubModel>
```

## Sub-model scenario (cont)

User wants to configure the “Service plan” account created in the previous step.

### Flow:

1. **punchIntoSubmodel** (new API) – Punch-in to the sub-model to configure the options.
2. processConfigurationPicks – to add “Telephone Set 4” in the sub-model configuration.



### Telecom Bundle

Price as configured: \$0.00

Errors: 0 Warnings: 0 Suggestions: 0

Clear Update Summary Add

Overview Fixed Line Telephone

```

<PunchIntoSubModel Country="US"
  Currency="USD" Language="en" ModelDepth="0"
  OrganizationCode="matrix" Path="Telecom/TelecomBundle"
  PunchInItem="TelecomBundle.Fixed Line Telephone.Service Plan 2.Plan.item1">
  <Picks>
    .....
  </Picks>
</PunchIntoSubModel>

```



### Fixed Line Service Plan 2

Price as configured: \$0.00

Errors: 0 Warnings: 0 Suggestions: 0

Clear Update Summary Save

Service Plan

<p><b>Accesories</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Fixed Line Accesory 3 \$0.00</li> <li><input type="checkbox"/> Fixed Line Accesory 4 \$0.00</li> </ul>	<p><b>Fixed Line Telephone Set</b></p> <ul style="list-style-type: none"> <li><input type="radio"/> None \$0.00</li> <li><input type="radio"/> Telephone Set3 \$0.00</li> <li><input checked="" type="radio"/> Telephone Set4 \$0.00</li> </ul>
<p><b>Number Selection</b></p> <ul style="list-style-type: none"> <li><input type="radio"/> Automatic \$0.00</li> <li><input type="radio"/> Let me choose my number \$0.00</li> </ul>	



## Sub-model scenario (cont)

User wants to save the “Fixed Line Service Plan 2” configuration and return to (parent) Telecom Bundle Configuration.

```
<ReturnFromSubModel Country="US"
  Currency="USD" Language="en" ModelDepth="1"
  OrganizationCode="matrix" Path="matrix/Telecom/FixedLineSP">
  <Picks>
    .....
  </Picks>
  <ParentConfigurations>
    <ParentConfiguration ModelDepth="0"
      PunchInItem="TelecomBundle.Fixed Line Telephone.Service Plan 2.Plan.item1"
      SubModelReturn="N" SubModelValidate="N">
      .....
    </ParentConfiguration>
  </ParentConfigurations>
  <InputProperties>
    .....
  </InputProperties>
</ReturnFromSubModel>
```

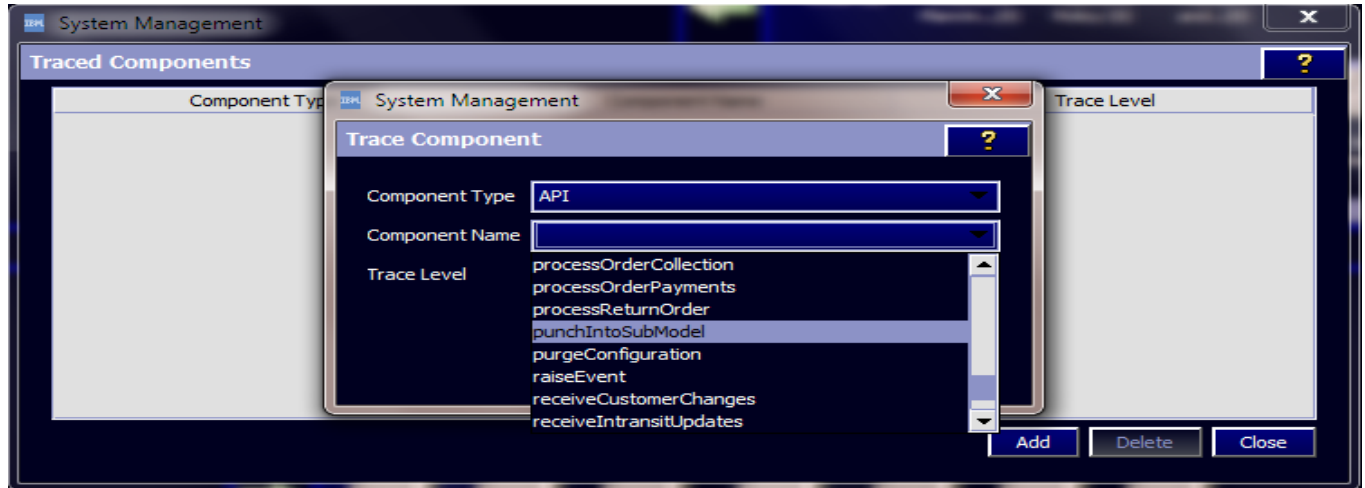
### Flow:

1. **returnFromSubModel** (new API) – Save the sub-model configuration and return to parent model.

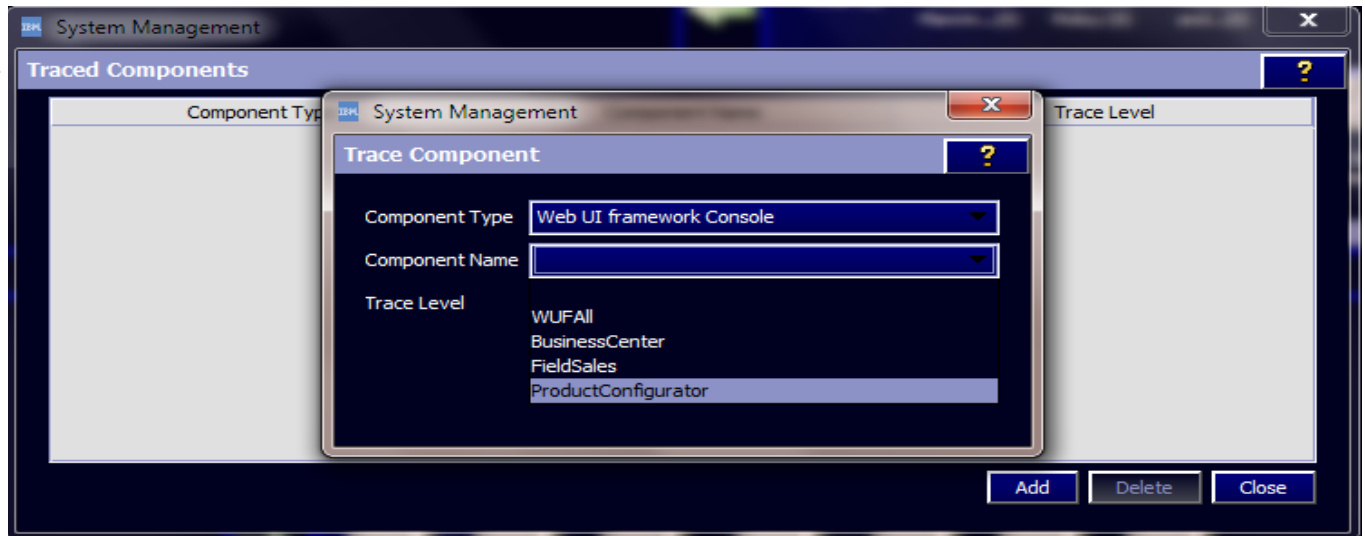
# Demo

# Troubleshooting – Logging and tracing

Configurator APIs →



Configurator application →



## Upgrade Bill of Material

- Upgrade BOM provide a means of capturing the delta changes the seller is making to the current offer.
- Seller is modifying a customer's existing offer and based on the customer's need is upgrading the current offer with additional options.
- The act of modifying the offer is recorded as upgrade to the current offer.
- Comparison view to show the difference between base configuration and updated configuration.
- Upgrade BOM Flow:
  - An external system invokes Configurator passing the base BOM and upgrade flag as true.
  - Configurator is launched in the upgrade mode.

# Upgrade Bill of Material

## Alpha Home Security Package

Errors: 0 Warnings: 0 Suggestions: 0

Price as configured: \$369.10

- 

Configuration Date:

<b>Screen Interface</b>	
<input type="radio"/> None	\$0.00
<input type="radio"/> Security Panel With Full Color Touch-Screen Interface	\$10.00
<input checked="" type="radio"/> Security Panel With Full Color Touch-Next Gen Interface	\$40.00
<b>Remote</b>	
<input type="radio"/> None	\$0.00
<input checked="" type="radio"/> Keychain Remote	\$20.00
<b>Motion Detector</b>	
<input type="radio"/> None	\$0.00
<input checked="" type="radio"/> Pet-Sensitive Motion Detector	\$30.00
<b>Sensor</b>	
<input type="radio"/> None	\$0.00
<input checked="" type="radio"/> Door/Window Sensor	\$40.00
<b>Security Yard Package</b>	
<input type="radio"/> None	\$0.00
<input checked="" type="radio"/> Security Yard Sign & Window Decals	\$50.00
<b>Submodel</b>	
<input checked="" type="radio"/> None	\$0.00
<input type="radio"/> Submodel	\$0.00

# Upgrade Bill of Material

The following two columns summarize your configurations before and after an upgrade, while at the bottom of the page we summarize the order that will be placed in order to upgrade your system to the new configuration you have specified.

Initial Configuration			Resultant Configuration		
Product ID	Description	Qty	Product ID	Description	Qty
Basic1	Alpha Home Security Package	1	Basic1	Alpha Home Security Package	1
Security1	Security Panel With Full Color Touch-Screen Interface	1	Security6	Security Panel With Full Color Touch-Next Gen Interface	1
Security2	Keychain Remote	1	Security2	Keychain Remote	1
Security3	Pet-Sensitive Motion Detector	1	Security3	Pet-Sensitive Motion Detector	1
Security4	Door/Window Sensor	1	Security4	Door/Window Sensor	1
			Security5	Security Yard Sign & Window Decals	1

Product ID	Description	Qty	Unit Price	Price
Security6	Security Panel With Full Color Touch-Next Gen Interface	1	\$40.00	\$40.00
Security5	Security Yard Sign & Window Decals	1	\$50.00	\$50.00

Price as configured: **\$90.00**

[Submit](#)

## Enhanced integration with Websphere Commerce

- We integrated with Websphere Commerce in previous release and continue to improve the integration.
- In the current integration scenarios the URL of the Visual Modeler application and product configurator is static.
- Through this feature, an extra parameter “LocaleCode” is provided which is the user preferred locale.
- Websphere Commerce Business user will now be able to launch Visual modeler in his preferred language from Websphere Commerce when locale code parameter is set as a request parameter by the calling application.
- Websphere Commerce store user will be able to punchin to the configurator in his preferred language from Websphere commerce when locale code parameter is set as a request parameter by the calling application.

## Extensibility Improvements for IBM Sterling Configurator

- In our continued effort to reduce TCO, we have made extensibility improvements
- Standardizes the customization's for Sterling Configurator and Configurator XAPIs to use customer overrides for customizations
- Customer overrides file is used to override
  - functionhandlers.properties
  - controls.properties
  - ObjectMap.properties
  - cachetypes.properties
  - EhCache.xml
  - RegressionTestReport.xml
- The customer override properties file is not changed during installation of upgrades or patches. Prevents from customized settings being overwritten



## Extensibility Improvements for IBM Sterling Configurator

Configurator File Name	Configurator Property Name
functionhandlers.properties	yfs.yfs.function.handler.files
controls.properties	yfs.yfs.control.handler.files
ObjectMap.properties	yfs.yfs.objectmap.extension.file
cachetypes.properties	yfs.yfs.cachetypes.extension.file
EhCache.xml	yfs.yfs.ehcache.override.file
RegressionTestReport.xml	yfs.yfs.regressiontestreport.override.file

---

## Knowledge Center Links for CPQ 9.4

[http://www-01.ibm.com/support/knowledgecenter/SS4QMC\\_9.4.0/com.ibm.help.cpq.newfeatures.doc/c\\_functional.html](http://www-01.ibm.com/support/knowledgecenter/SS4QMC_9.4.0/com.ibm.help.cpq.newfeatures.doc/c_functional.html)