# Usage CCDI API

## Inserting component types in DB

* CDDI holds the following component types

|  |  |  |
| --- | --- | --- |
| **COMPONENT Type** | **Input** | **Output** |
| PROPERTY | name | Value |
| ENVIRONMENT | key | Value |
| GATEWAY | action | Name |
| VERBIAGE | key | Text |
| VALIDATOR | action | Value |
| GENERIC | category, page, block | Text |
| GLOSSARY | term | Text |

* For, each component type, we need to insert data in the below tables
  + **CMPNT** – Output is stored in this table
    - CMPNT\_TYPE\_CD (is the type of the component)
    - CNTXT\_TXT (output to be retrieved)
    - CMS\_STTS\_CD = PUBLISHED
    - CDI\_STTS\_CD= PUBLISHED
  + **CMPNT\_ATRB** – Input is stored in this table
    - ATRB\_NAME – (Refer the above table and appropriate input has to be passed for component type)
    - ATRB\_VALUE (Input to be passed)
  + **CNTXT –** Context information is stored in this table
* Below is the sample data where, content in ATRB\_VAL is the input and CNTNT\_TXT is output

|  |  |  |  |
| --- | --- | --- | --- |
| CMPNT | | CMPNT\_ATRB | |
| **CMPNT\_TYPE\_CD** | **CNTNT\_TXT** | **ATRB\_NM** | **ATRB\_VAL** |
| PROPERTY | Hios Job, Sally Job, Renewal Job | name | e3\_jobs |
| ENVIRONMENT | DEV | key | e3.environment |
| GATEWAY | ActionGateway | action | UserFlow |
| VERBIAGE | TextBox | key | label/ccdi/admin/entitlement/options |
| VALIDATOR | ValidateUser,ValidatePassword | action |  |
| GENERIC | Welcome to E3 | category | search |
| page | 9 |
| block | 9 |
| GLOSSARY | EEE | term | site |

**Note:** For GENERIC component type, we need to insert three rows in CMPNT\_ATRB as shown in the above table.

## To retrieve component data from DB

|  |  |  |
| --- | --- | --- |
| **COMPONENT TYPE** | **Sample code to retrieve component data from DB** | |
| **Pass INPUT as below**  **(Content highlighted in orange color is input)** | **Retrieve OUTPUT as below** |
| **PROPERTY** | In client class, **PropertyValueResolver** will be injected.  @Autowired  PropertyValueResolver propertyValueResolver;  Use resolver as below  Property property = propertyValueResolver.resolveProperty(contexts, **name**); | String output = property.getValue(); |
| **ENVIRONMENT** | In client class, **EnvironmentEntryResolver** will be injected.  @Autowired  EnvironmentEntryResolver environmentEntryResolver;  Use resolver as below  EnvironmentEntry environment = environmentEntryResolver.resolveEnvironmentEntry(contexts, **key**) | String output = environment.getValue(); |
| **GATEWAY** | In client class, **GatewayNameResolver** will be injected.  @Autowired  GatewayNameResolver gatewayNameResolver;  Use resolver as below  Gateway gateway = gatewayNameResolver.resolveGateway(contexts, **action**) | String output = gateway.getText(); |
| **VERBIAGE** | In client class, **VerbiageTextResolver** will be injected.  @Autowired  VerbiageTextResolver verTxtResolver;  Use resolver as below  Verbiage verbiage = verTxtResolver.resolveText(contexts, **key**) | String output = verbiage.getText(); |
| **VALIDATOR** | In client class, **ValidatorRulesetResolver** will be injected.  @Autowired  ValidatorRulesetResolver valRulResolver;  Use resolver as below  Validator validator = valRulResolver.resolveRuleset(contexts, **action**) | String output = validator.getValue(); |
| **GENERIC** | In client class, GenericComponentResolver will be injected.  @Autowired  GenericComponentResolver genericComponentResolver;  Use resolver as below  GenericComponent generic = genericComponentResolver.resolveGenericComponent(contexts, **page**, **category**, **block**) | String output = generic.getText(); |
| **GLOSSARY** | In client class, GlossaryComponentResolver will be injected.  @Autowired  GlossaryComponentResolver glossaryCmptRslvr;  Use resolver as below  GlossaryComponent glossary = glossaryComponentResolver.resolveGlossaryComponent(contexts, **term**) | String output = glossary.getValue(); |

## To use property place holder

**DB**

* Insert property place holder details in CCDI with the below details
  + CMPNT.CMPNT\_TYPE\_CD='PROPERTY'
  + CMPNT.CMS\_STTS\_CD='PUBLISHED'
  + CMPNT.CMPNT.CDI\_STTS\_CD='PUBLISHED'
  + CMPNT.TTL\_TXT like '%PLACEHOLDER\_CONFIG%'
  + CMPNT\_ATRB.ATRB\_NM='key'
  + CMPNT\_ATRB.ATRB\_VAL='input’ (sample input *e3.log4j2.config*)
  + CMPNT.CNTNT\_TXT='VALUE'

Where, VALUE is the content which you would like to map with the key

With appropriate context information key and value

**Sample code**

* Use property value place in spring context file

|  |
| --- |
| <bean id=*"log4jConfigManager"* class=*"com.wlp.e3.logging.api.config.Log4jConfigurationManager"* init-method=*"initLogConfig"*>  <property name=*"logConfig"* value=*"${e3.log4j2.config}"*></property>  <lookup-method name=*"createBasicAsyncAppender"* bean=*"basicAsyncAppender"*/>  </bean> |

In the above code, **e3.log4j2.config** will be replaced with the value available in **CMPNT.CNTNT\_TXT** field.