

www.int4.com

Groovy Development in SAP Cloud Platform Integration

Eng Swee Yeoh

Integration Architect, SAP Mentor

<https://people.sap.com/engswee.yeoh/>

About Int4

Who we are

- Int4 is an SAP Partner and a market leader in test automation of SAP systems integration.
- Our team is represented by 2 SAP Mentors, 10 SAP PRESS authors, and over 30 SAP integration professionals.
- We have proven experience in consulting and delivery of the biggest integration projects all over the world.
- Our goal is to make an impact through automation; and support the SAP community by sharing our knowledge on paper, in videos, and during events.



We have a one-step solution for **testing all SAP integration scenarios** - Int4 IFTT.



“Integration is like a box of chocolates”

Challenges in integration

No two integrations are the same

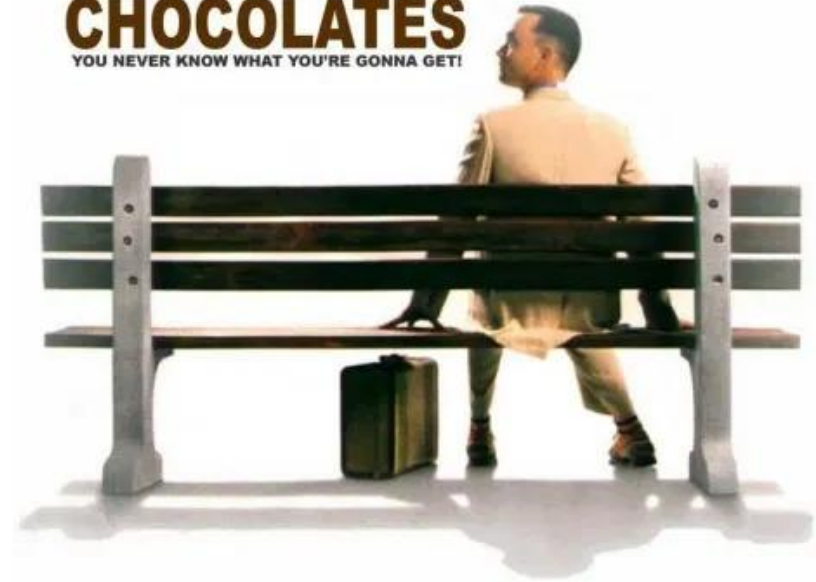
Systems and applications with differing integration requirements

- content type
- connectivity

Not everything can be achieved by standard functionality – custom coding required

“You never know what you’re gonna get!”

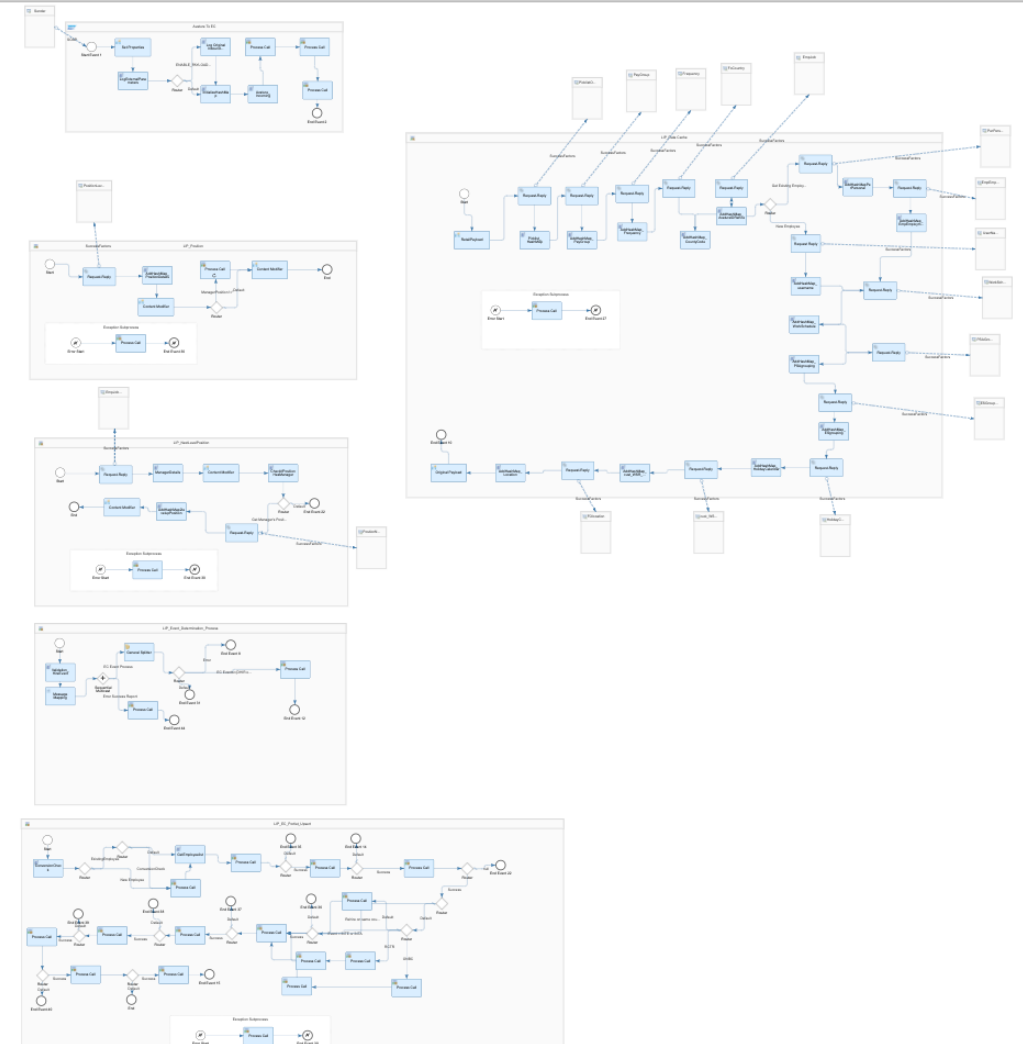
**LIFE IS LIKE A BOX OF
CHOCOLATES**
YOU NEVER KNOW WHAT YOU'RE GONNA GET!



- **Hybrid landscapes**
 - On-premise and cloud
 - Multiple vendors
- **Lack of customization options in SaaS - pushing application logic to integration platform**
- **Shift towards Agile and/or DevOps methodology**

Complexity of Integration Scenarios

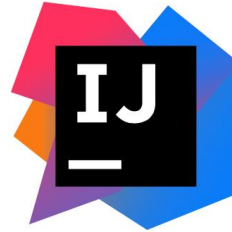
- **SuccessFactors and Marketing Cloud**
- **Heavy usage of OData APIs**
 - Multiple calls
 - Correlation and orchestration
- **Application logic in integration platform**
 - Avoid where possible
 - If not, then modularize Integration Flow design so that it is manageable



- **Script step in Integration Flow**
- **Implementation of custom logic**
- **Powerful, dynamic language, yet easy to use**
- **Flat learning curve for Java developers**



#1 - Use an Integrated Development Environment (IDE)



- For simple logic, Web UI built-in editor is fine
 - Basic syntax check and other features on SAP's roadmap
- For more complicated logic, switch to an IDE
- Recommendation - IntelliJ IDEA Community Edition
 - Native Groovy support
 - Syntax check
 - Auto completion
 - Debug without using `messageLog` statements

A screenshot of the IntelliJ IDEA IDE interface. The main editor window displays a Groovy file named 'GenerateFilterCriteria.groovy'. The code includes imports for 'com.sap.gateway.ip.core.customdev.util.Message' and 'java.text.ParseException'. A method 'processData' is defined, which takes a 'Message' object and processes a query string. An autocomplete dropdown menu is visible over the 'uri.get' call, listing various URI-related methods like 'getClass()', 'getAuthority()', 'getFragment()', 'getHost()', 'getPath()', 'getPort()', 'getQuery()', 'getRawAuthority()', 'getRawFragment()', 'getRawPath()', and 'getRawQuery()'. The left sidebar shows the project structure and a commit window. The bottom status bar indicates the current file is 'GenerateFilterCriteria.groovy'.

#2 - Write “Groovier” code

- Groovy Style Guide
 - Omit semicolons (;) for end of statement
 - Combine usage of static and dynamic typing
- Use Groovy String
 - Interpolated string with embedded placeholders
 - Simplifies creation of String with dynamic values

```
Message processData(Message message) {  
    Reader reader = message.getBody(Reader)  
    Map headers = message.getHeaders()  
    Map properties = message.getProperties()  
}
```

```
def Contract = new XmlSlurper().parse(reader)  
def writer = new StringWriter()  
def builder = new MarkupBuilder(writer)
```

```
message.setBody("Groovy version is ${GroovySystem.version}")
```


#2 - Write “Groovier” code

- Use Groovy shortcuts for Collections
 - Define and populate during initialization
- Iterate using each()

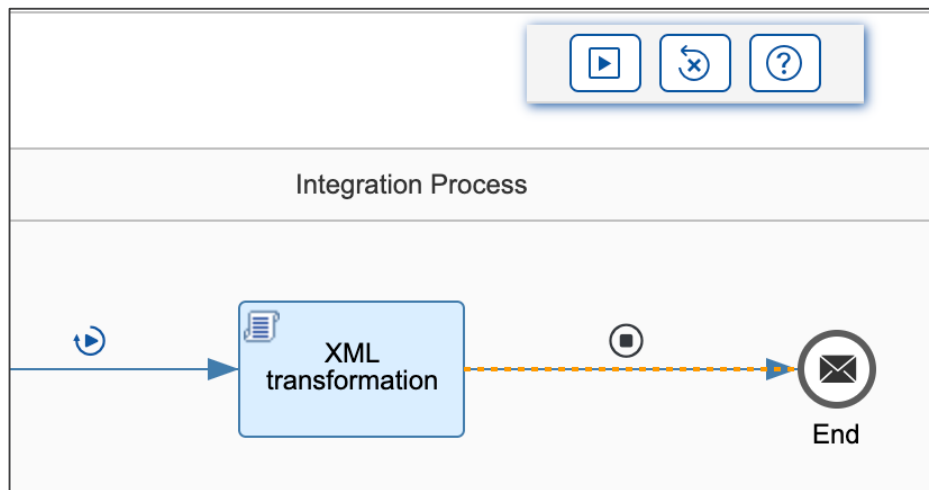
```
def newList = [1, 7, 15, 200]
def newMap = [ firstName: 'John', lastName: 'Doe']
```

```
newList.each { item ->
|   println "${item}"
}

newMap.each { k, v ->
|   println "key = ${k}, value = ${v}"
}
```

#3 - Test Groovy script before deployment

- Integration Flow Simulation Mode in Web UI



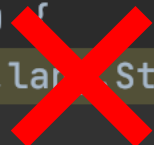
- Local testing in IDE

```
Run: MappingTesterInvoice x
Body:
<INCINV_CREATE03>
  <IDOC BEGIN='1'>
    <EDI_DC40 SEGMENT='1'>
      <IDOC TYP>INCINV_CREATE</IDOC TYP>
      <MESTYP>INCINV_CREATE</MESTYP>
      <MESCOD>008</MESCOD>
      <MESFCT>A</MESFCT>
      <SNDPOR>CPI</SNDPOR>
      <SNDPRT>LS</SNDPRT>
      <SNDP RN>CXC</SNDP RN>
      <RCVPOR>SAPS4D</RCVPOR>
      <RCVPRT>LS</RCVPRT>
      <RCVPRN>S4DCLNT110</RCVPRN>
```

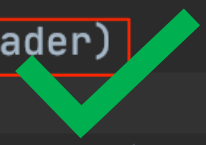
#4 - Handle message body efficiently

- Accessing the message body using `getBody()` is one of the most commonly used statement
- Avoid getting the body as a String, as it consumes additional resources
- Use a Reader to stream the content of the body

```
Message processData(Message message) {  
    def body = message.getBody(java.lang.String)  
}
```



```
VendorInvoiceToINCINV_CREATE.groovy ×  
1  import com.sap.gateway.ip.core.customdev.util.Message  
2  import groovy.xml.MarkupBuilder  
3  
4  Message processData(Message message) {  
5      Reader reader = message.getBody(Reader)  
6      |  
7      def BILLINGDOC = new XmlSlurper().parse(reader)
```



#5 - Process XML and JSON

- Graphical message mapping suitable for simple transformation
- For more complicated transformation, consider using Groovy script
- Native support for XML and JSON
- Accessing nodes using dot notation (GPath)
- WYSIWYG builder for generating XML/JSON output

```
VendorInvoiceToINCINV_CREATE.groovy
35 'E1BP_INCINV_CREATE_HEADER'('SEGMENT': '1') {
36     INVOICE_IND( value: 'X')
37     DOC_TYPE( value: 'RE')
38     DOC_DATE(transformDate(BILLINGDOC.DOCDATE.text()))
39     PSTNG_DATE(transformDate(BILLINGDOC.DOCDATE.text()))
40     REF_DOC_NO(BILLINGDOC.DOCREF)
41     DIFF_INV(BILLINGDOC.CPID)
42     CURRENCY(BILLINGDOC.CURRENCYID)
43     GROSS_AMOUNT(BILLINGDOC.DOCAMOUNT)
44     PMNTTRMS(BILLINGDOC.PAYTERMID)
45     BLINE_DATE(transformDate(BILLINGDOC.DOCDATE.text()))
46     HEADER_TXT(BILLINGDOC.SRCDOCREF)
47     ALLOC_NMBR(BILLINGDOC.CTRREF)
48 }
49 // Filter out invalid lines
50 def InventoryItems = BILLINGDOC.ITEMS.ITEM.findAll { item -> item.QTY.text() && item.DONO.text() }
51 int lastItemIndex = InventoryItems.size() - 1
52 // Find the total line that contains the unit price
53 def totalLine = BILLINGDOC.ITEMS.ITEM.findAll { item ->
54     item.UPRICE.text() &&
55     (item.ITEMDESC.text().toUpperCase().startsWith('TOTAL') || item.ITEMDESC.text().toUpperCase().startsWith('PROVISIONAL'))
56 }
57 BigDecimal headerAmountWithTax = BILLINGDOC.DOCAMOUNT.text() as BigDecimal
58 BigDecimal balanceAmountWithoutTax = totalLine[0].ITEMAMT.text() ? totalLine[0].ITEMAMT.text() as BigDecimal : 0
59 BigDecimal taxAmount = headerAmountWithTax - balanceAmountWithoutTax
60
61 InventoryItems.eachWithIndex { item, Integer index ->
62     BigDecimal qty = item.QTY.text() as BigDecimal
63     BigDecimal unitPrice
```

Developing Groovy Scripts for SAP Cloud Platform Integration

- **Authors**
 - Vadim Klimov and Eng Swee Yeoh
 - Working with SAP CPI (a.k.a. HCI) since 2015
- **Available since 18 June 2020**
- **Easy-to-read E-Bite format (126 pages)**
- **In-depth explanation of above tips with examples**

<https://bit.ly/37BSL9R>



A lot more to be found in the E-Bite!

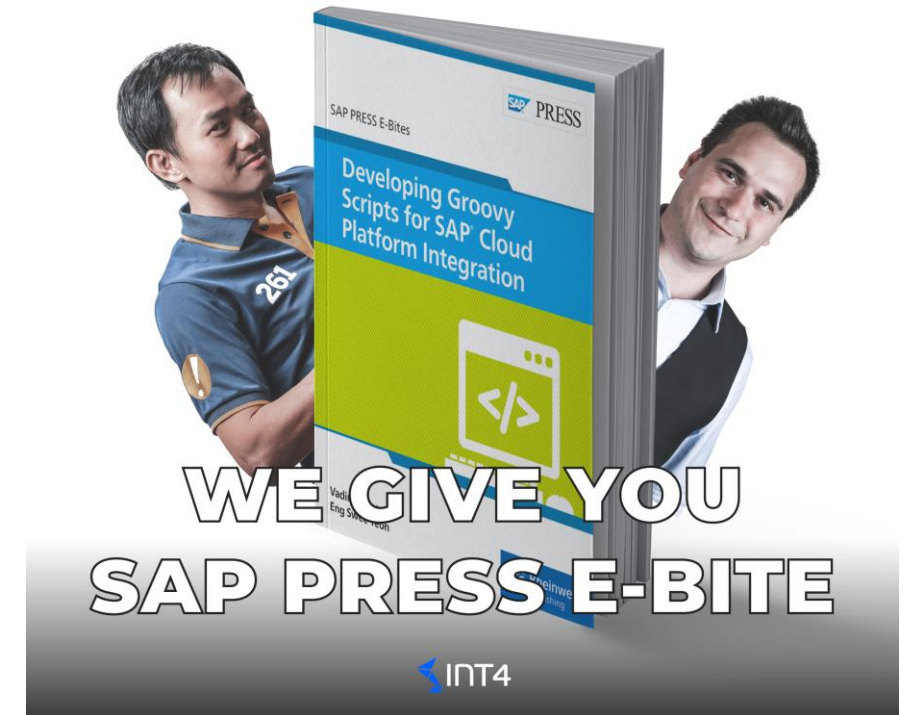
- Regular expressions
- Handling other formats like CSV and PDF
- Dependency management
- Sneak peek into the underlying frameworks of CPI

Join Our Meme Contest

- Post a meme as a comment on the LinkedIn announcement
- Get as many likes on the meme by end of **Tuesday, 4th of August**
- Meme with highest number of likes win a copy of our E-Bite

<https://lnkd.in/erChgk9>

YOU GIVE US MEMES



www.int4.com
office@int4.com

Eng Swee Yeoh

engswee.yeoh@int4.com



SAP® Certified
Integration with SAP® S/4HANA

SAP® Certified
Powered by SAP NetWeaver®