Pts from scrap:

**public** String getRelativeUri(String input) {

// input doesn't have /p/ : prefix client/org/app/p/{domain-root} from incoming command

**int** iFirstQ = StringUtils.indexOf(input, "?");

**final** String searchSeq = (iFirstQ != StringUtils.INDEX\_NOT\_FOUND) ? StringUtils.substring(input, 0, iFirstQ) : input;

**if**(!StringUtils.contains(searchSeq, Constants.SEGMENT\_PLATFORM\_MARKER.code)) {

String prefix = buildUri(Type.PlatformMarker) + getRootDomainUri();

**return** prefix + input;

}

// input starts with /p/ : prefix client/org/app from incoming command

**if**(StringUtils.startsWith(input, Constants.SEGMENT\_PLATFORM\_MARKER.code)) {

String prefix = buildUri(Type.AppAlias);

**return** prefix + input;

}

// input is complete: use as is

**return** input;

}

/\* **TODO** Refactor -- START -- \*/

**public** **boolean** isView() {

String domainRoot = getRootDomainAlias();

**return** StringUtils.startsWith(domainRoot, Constants.PREFIX\_FLOW.code);

}

**public** String getAppAlias() {

**return** getAlias(Type.AppAlias);

}

**public** String getRootClientAlias() {

**return** getAlias(Type.ClientAlias);

}

**public** CommandElement getRootDomainElement() {

**return** getElement(Type.DomainAlias).get();

}

**public** String getRootDomainAlias() {

**return** getRootDomainElement().getAlias();

}

**public** String getRootDomainUri() {

**return** getRootDomainElement().getUri();

}

/\*\*

\* Returns the absolute domain alias of this command.

\*

\* <p>

\* <b>Examples:</b>

\* <p>When <b>absoluteUri</b> = <i>/Acme/ab/cd/domain/ef/gh/\_process?fn=\_set</i> then getAbsoluteDomainAlias() returns <i>/domain</i></li>

\* **@return** the absolute domain alias of this command.

\*/

**public** String getAbsoluteDomainAlias() {

String a = buildAlias(root().findFirstMatch(Type.DomainAlias));

**return** a;

}

/\*\*

\* Returns the absolute domain URI of this command.

\*

\* <p>

\* <b>Examples:</b>

\* <p>When <b>absoluteUri</b> = <i>/Acme/ab/cd/domain/ef/gh/\_process?fn=\_set</i> then getAbsoluteDomainAlias() returns <i>/domain/ef/gh</i></li>

\* **@return** the absolute domain URI of this command.

\*/

**public** String getAbsoluteDomainUri() {

String u = buildUri(root().findFirstMatch(Type.DomainAlias));

**return** u;

}

**public** String getProcessAlias() {

String a = buildAlias(root().findFirstMatch(Type.ProcessAlias));

**return** a;

}

**public** String getProcessUri() {

String u = buildUri(root().findFirstMatch(Type.ProcessAlias));

**return** u;

}

/\*\*

\* Returns the absolute alias of this command.

\*

\* <p>

\* <b>Examples:</b>

\* <p>When <b>absoluteUri</b> = <i>/Acme/ab/cd/domain/ef/gh/\_process?fn=\_set</i> then getAbsoluteAlias() returns <i>/Acme/ab/cd/domain/ef/gh</i></li>

\* **@return** the absolute alias of this command.

\*/

**public** String getAbsoluteAlias() {

String a = buildAlias(root());

**return** a;

}

/\*\*

\* Returns the absolute alias with only the action included of this command.

\*

\* <p>

\* <b>Examples:</b>

\* <p>When <b>absoluteUri</b> = <i>/Acme/ab/cd/domain/ef/gh/\_process?fn=\_set</i> then getAbsoluteDomainAlias() returns <i>/domain/ef/gh/\_process</i></li>

\* **@return** the absolute alias with only the action included of this command.

\*/

**public** String getAbsoluteAliasWithAction() {

String a = buildAlias(root());

**return** a + "/" + **this**.getAction();

}

/\*\*

\* Returns the absolute alias up to the root domain of this command.

\*

\* <p>

\* <b>Examples:</b>

\* <p>When <b>absoluteUri</b> = <i>/Acme/ab/cd/domain/ef/gh/\_process?fn=\_set</i> then getAbsoluteAliasTillRootDomain() returns <i>/Acme/ab/cd/domain</i></li>

\* **@return** the absolute alias up to the root domain of this command.

\*/

**public** String getAbsoluteAliasTillRootDomain() {

String a = buildAlias(root(), Type.DomainAlias);

**return** a;

}

/\* **TODO** Refactor -- END -- \*/

**public** String buildAlias(CommandElementLinked startElem) {

**return** traverseElements(startElem, (cmdElem, sb) -> sb.append(cmdElem.getAliasUri()));

}

**public** String buildAlias(Type endWhentype) {

**return** traverseElements(root(), endWhentype, (cmdElem, sb) -> sb.append(cmdElem.getAliasUri()));

}

**public** String buildAlias(CommandElementLinked startElem, Type endWhentype) {

**return** traverseElements(startElem, endWhentype, (cmdElem, sb) -> sb.append(cmdElem.getAliasUri()));

}

**public** String buildUri(CommandElementLinked startElem) {

**return** traverseElements(startElem, (cmdElem, sb) -> sb.append(cmdElem.getUri()));

}

**public** String buildUri(Type endWhenType) {

**return** traverseElements(root(), endWhenType, (cmdElem, sb) -> sb.append(cmdElem.getUri()));

}

**public** String buildUri(CommandElementLinked startElem, Type endWhenType) {

**return** traverseElements(startElem, endWhenType, (cmdElem, sb) -> sb.append(cmdElem.getUri()));

}

**public** String traverseElements(CommandElementLinked startElem, BiConsumer<CommandElement, StringBuilder> cb) {

StringBuilder sb = **new** StringBuilder();

traverseElements(startElem, (cmdElem) -> cb.accept(cmdElem, sb));

**return** sb.toString();

}

**public** **void** traverseElements(CommandElementLinked startElem, Consumer<CommandElement> cb) {

**while** (startElem != **null**) {

cb.accept(startElem);

startElem = startElem.next();

}

}

**public** String traverseElements(CommandElementLinked startElem, Type type, BiConsumer<CommandElement, StringBuilder> cb) {

StringBuilder sb = **new** StringBuilder();

traverseElements(startElem, type, (cmdElem) -> cb.accept(cmdElem, sb));

**return** sb.toString();

}

**public** **void** traverseElements(CommandElementLinked startElem, Type type, Consumer<CommandElement> cb) {

**while** (startElem != **null**) {

cb.accept(startElem);

**if** (startElem.getType().equals(type)) {

**break**;

}

startElem = startElem.next();

}

}

**public** CommandElementLinked createRoot(Type type, String uri) {

CommandElementLinked root = **new** CommandElementLinked();

root.setType(type);

root.setUri(uri);

setRoot(root);

**return** getRoot();

}

**public** String toUri() {

String baseUri = buildUri(getRoot());

StringBuilder sb = **new** StringBuilder(baseUri);

/\* action \*/

sb.append(Constants.SEPARATOR\_URI.code).append(getAction().name());

/\* event \*/

**if**(isEvent()) {

sb.append(Constants.SEPARATOR\_URI.code).append(getEvent());

}

/\* behavior(s) \*/

sb.append("?").append(Constants.MARKER\_URI\_BEHAVIOR.code).append("="); // ?b=

sb.append(getBehaviors().get(0).name()); // $execute (or other behavior)

getBehaviors().stream().sequential().skip(1).forEach(b->{

sb.append(Constants.SEPARATOR\_AND.code).append(b.name());

});

/\* **TODO**: other request params \*/

**return** sb.toString();

}

**public** String[] getParameterValue(String requestParameter){

**if**(requestParams != **null** && requestParams.containsKey(requestParameter)){

**return** requestParams.get(requestParameter);

}

**return** **null**;

}

**public** String getFirstParameterValue(String requestParameter){

**if**(requestParams != **null** && requestParams.containsKey(requestParameter)){

String[] value = requestParams.get(requestParameter);

**if**(value != **null** && value.length > 0){

**return** value[0];

}

}

**return** **null**;

}

**public** **boolean** hasRawPayload() {

**return** getFirstParameterValue("rawPayload") != **null**;

}

**public** String getRawPayload() {

**return** getFirstParameterValue("rawPayload");

}

}