Quick Reference

HeliumDev Client Setup

```
java -jar HeliumDev-app-1.5.0-jar-with-dependencies.
jar
he-dev> set server=dev
he-dev> server set uri=https://dev.mezzanineware.com
he-dev> server set user=usrname
he-dev> server password
he-dev> set mobileNumber=278212345678
he-dev> set project=Helium Tutorial
he-dev> project set roles=System Admin
he-dev> project set dir=/workspace/helium-tut
```

HeliumDev Client Common Commands

```
java -jar HeliumDev-app-1.5.0-jar-with-dependencies.
jar
he-dev> help
he-dev> new-sandbox
he-dev> build
he-dev> run
he-dev> release
```

Project Folder Structure

Hello World

XML view snippet

```
<info label="some.lang_file.label">
   <binding variable="helloWorld" />
</info>
```

```
notification-templates
   en.json
builtin-reports
   foo-report
       foo_image.png
       foo_report.jrxml
jasper-reports
   bar-report
       master.jrxml
       report.json
model
   MyCustomObject.mez
sql-scripts
   myStartupScript.sql
services
   UserInvite.mez
utilities
   dateStringFormatter.mez
web-app
        email-templates
       template1.html
    images
       BarMenuIcon.png
       FooMenuIcon.png
    lang
       en.lang
    presenters
       PresenterFilesCon.mez
       TainingDslUnits.mez
    views
       BarView.vxml
       FooView.vxml
        static
                dashboard.html
        dashboard.css
                dashboard.js
```

DSL presenter snippet

```
string helloWorld;
void init() {
  helloWorld = "Hello, World";
}
```

DSL Basics

```
//this is a comment

/* this is a multi-
        line comment */

unit UnitName;

MyCustomFoo globalScopedFoo;

void myFunction () {
        globalScopedFoo = MyCustomFoo:new();
}

MyCustomFoo myReturnTypeFunction () {
        MyCustomFoo localScopedFoo = MyCustomFoo:
new();
        return localScopedFoo;
}
```

View Basics

```
<?xml version="1.0" encoding="UTF-8"?>
<ui xmlns="http://uiprogram.mezzanine.com/View"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
   xsi:schemaLocation="http://uiprogram.mezzanine.
com/View ../View.xsd">
   <view label="view_heading.for_this_view" unit="
LinkedUnit"
         init="onPageLoadFunction">

      <!-- this is a comment -->

      <info label="at.least.one.widget">
         <binding variable="variableInLinkedUnit"
/>
         <binding variable="otherUnit:
variableInOtherUnit" />
      </info>
   </view>
</ui>
```

Navigation

NurseManagement.vxml

```
<view label="view_heading.nurse_management" unit="
NurseManagement">
```

NurseDetails.vxml

```
<view label="view_heading.nurse_details" unit="
NurseDetails">
        <button label="action.back" action="back"
/>
```

NurseDetails.mez

```
unit NurseDetails;

//returns member of auto-generated enum DSL_VIEWS
DSL_VIEWS back() {
    return DSL_VIEWS.NurseManagement;
//corresponds to vxml file name
}
```

Objects

```
object Person
{
    string fname;
    string sname;
    int age;
}
```

```
persistent object Person
{
    string fname;
    string sname;
    int age;
}
```

Collections

```
int[] integerCollection;
MyCustomObject[] objectCollection;
MyCustomObject fourthObject = objectCollection.get
(3);
MyCustomObject objectAtIndex = objectCollection.get
(i);
```

Built-in Data Types

type	operators	samples	comparison operators	samples
int	+ - * / %	int a = (2 + 3 * 15) % 2	> < == >= <= !=	if (a <= b) {...
string			> < == >= <= !=	string a = "alpha"; string b = "bravo"; if (a < b) {... //true

decimal	+ - * / %	decimal a = (2 + 3 * 15) % 2	> < == >= <= !=	if (a <= b / 5) {...
bool	|| &&	bool a = b || c	== !=	if ((i == 1) && (i == j)) {...

date			> < == >= <= !=	date d = Mez:now();
				date d2 = Date:addDays(d, 10);
				if (d2 > d) {... //true

datetime			> < == >= <= !=	datetime dt = Mez:now();
				datetime dt2 = Date:addDays(dt, 10);
				if (dt2 > dt) {... //true

uuid			== !=	if (u == u2) { ...
void				
blob				
json				
jsonarray				

(Variables of all primitive types are null after declaration, until assigned a value)

Control Structures

```
if(name == "Tom") {
    Mez:log("Hello Tom!");
} else if (name == "Frank") {
    Mez:log("Hello Frank!");
} else {
    Mez:log("Hi! Who are you?");
}
```

```
for(int i = 1; i <= 10; i++) {
    Mez:log(i);
}
```

```
int i = 0;
for(; i < 10;) {
    Mez:log("While style loop!");
}
```

```
SomeObject[] objectCollection = SomeObject:all();
foreach(SomeObject currentObject:
objectCollection) {
        currentObject.description = "Foreach style
loop!";
}
```

Type Conversion

```
date value = Date:fromString("2017-01-02");
datetime value = Date:fromTimeString("2017-1-20 08:
45:12 GMT");
```

```
int value = Integer:fromString("12");
```

```
decimal value = Decimal:fromString("12.01");
```

```
uuid value = Uuid:fromString("c1bd11cb-4bb2-43e5-
88b4-009ed40fc69f");
```

```
string s = value; //value = any of above types
string s = String:concat("", value);
```

```
json jsonPerson = /%
        {
                "name": "John",
                "surname": "Smith",
                "phoneNumber": "555-6162"
        }
%/;
```

```
try {
    Mez:log("inside try statement");
    // throws an exception now ...
} catch (ex){
    Mez:log("inside catch statement");
} finally {
    Mez:log("inside finally statement");
}
```

```
void throwStatementExample(){
    Mez:log("Throwing an exception now ...");
    string error = "Oops, something went wrong!";
    throw error;
}
```

```
jsonarray pets = /%
        [
                {
                        "name": "Jasmine",
                        "type": "Dog"
                },
                {
                        "name": "Markus",
                        "type": "Mole-rat"
                }
        ]
%/;
```

```
jsonarray primeNumbersJsonArray = /%[2, 3, 5, 7,
11, 13, 17, 19, 23, 29]%/;
int[] primeNumbersPrimitiveArray =
primeNumbersJsonArray;
```

Built-in Functions (BIFs)

Date BIFs

```
date d = Date:addSeconds(startDate, 10);
date d = Date:addDays(startDate, 10);
date d = Date:addMonths(startDate, 10);
int diff = Date:secondsBetween(dt1, dt2);
int diff = Date:daysBetween(d1, d2);
int diff = Date:monthsBetween(d1, d2);
int yr = Date:extract(dt, "year");
date d = Date:fromString("2017-01-02");
datetime dt = Date:fromTimeString("2017-1-20 08:45:
12 GMT");
datetime dt = Date:now();
date d = Date:today();
```

String BIFs

```
string s = String:concat("abc","def");
string[] result = String:split("abc def", " ");
string[] result = String:split("abc|def", "\\|");
int i = String:length("Hello world");
string s = String:substring("Hello World", 1, 4);
string s = String:lower("Hello World");
string s = String:upper("hello world");
bool b = String:startsWith("Hello World", "Hello");
bool b = String:endsWith("Hello World", "World");
bool b = String:regexMatch("27000111222", "^27[0-9]
{9,}$");
int i = String:indexOf("Hello World", "ello");
string s = String:join(stringCollection, " ");
string s = String:translate("translation.key");
string s = /%
        This is a
        multi line string
        declaration
%/;
```

Integer BIFs

```
int value = Integer:fromString("12");
```

Decimal BIFs

```
decimal value = Decimal:fromString("12.01");
```

Uuid BIFs

```
uuid value = Uuid:fromString("c1bd11cb-4bb2-43e5-
88b4-009ed40fc69f");
```

Math BIFs

```
int i = Math:pow(2, 8);
decimal d = Math:sqrt(3);
decimal r = Math:random();
```

Mez BIFs

Persistent Entity BIFs

```
datetime t = Mez:now();
date d = Mez:today();
Mez:log("Hello, World");
Mez:warn("Something could go wrong");
Mez:error("Something went wrong");
Mez:alert("translation.key");
Mez:alertWarn("translation.key");
Mez:alertError("translation.key");
Mez:userRole();
Mez:email(person, "email.descriptionKey", "email.
subjectKey", "email.bodyKey");
Mez:email(person.emailAddress, "email.
descriptionKey", "email.subjectKey", "email.
bodyKey");
Mez:email(person.emailAddress, "email.
descriptionKey", "email.subjectKey", "email.
bodyKey", EMAIL_TEMPLATES.email_template);
Mez:email(person.emailAddress, "email.
descriptionKey", "email.subjectKey", "email.
bodyKey", "farmer_purchase_invoice_report
/FarmerPurchaseInvoice.jrxml",
"additional_farmer_report_1
/AdditionalFarmerReport1.jrxml");
Mez:emailAttach(person.emailAddress, "email.
descriptionKey", "email.subjectKey", "email.
bodyKey", {"farmer_purchase_invoice_report
/FarmerPurchaseInvoice.jrxml",
getFarmerPurchaseInvoiceFileName()});
Mez:sms(client, "mobileNumber", "exampleKey.
smsText");
MezBatch stockUpdateBatch = Mez:createBatch
(fileUpload._id, fileUpload.data);
```

```
Person p = Person:new();
p.save();
Person p = Person:read(getUserUUID());
Person:delete(p);
Person currentPerson = Person:user();
p.invite(p.mobileNumber);
p.invite(p.mobileNumber, p.emailAddress);
person.removeRole();
person.notify("description.key", "sms.content.key",
   "email.subj.key", "email.content.key");
Person[] plist = Person:fromCsv(uploadedFile.
blobData);
person.pay(shop, "ZAR", amount);
person.payWithRef(shop, "ZAR", amount, ourPayId,
extraPayInfoString);

Person p = Person:fromCsvLine(personBatch.header,
personBatchItem.value);
```

Selector BIFs

Collection BIFs

```
Person p = plist.pop();
Person p = plist.drop();
int len = plist.length();
Person firstPerson = plist.first();
Person lastPerson = plist.last();
plist.append(p);
plist.prepend(p);
plist.add(i, p);
Person p = plist.get(i);
plist.remove(i);
plist.sortAsc("dob"); //sort according to date
attr. "dob"
plist.sortDesc("dob");
plist.clear();
plist.notify("description.key", "sms.content.key",
   "email.subj.key", "email.content.key");
Person[] plistDoctorsSubset = plist.select(equals
(title, "Dr."));
```

```
Person[] plist = Person:all();
Person[] plist = Person:equals(deleted, false);
Person[] plist = Person:empty(mobileNum);
Person[] plist = Person:between(dob, date1, date2);
Person[] plist = Person:lessThanOrEqual(age, num);
Person[] plist = Person:lessThan(age, num);
Person[] plist = Person:greaterThan(age, num);
Person[] plist = Person:attributeIn(state, "Any,
One, Of, These");
Person[] plist = Person:relationshipIn(reportsTo,
person);
Person[] plist = Person:contains(name, "a");
Person[] plist = Person:beginsWith(name, "A");
Person[] plist = Person:endsWith(name, "a");
Person[] plist = Person:notEquals(deleted, true);
Person[] plist = Person:notEmpty(mobileNum);
Person[] plist = Person:notBetween(dob, date1,
date2);
Person[] plist = Person:notContains(name, "a");
Person[] plist = Person:notBeginWith(name, "A");
Person[] plist = Person:notEndsWith(name, "a");
Person[] plist = Person:notAttributeIn(state,
"Any, One, Of, These");
Person[] plist = Person:notRelationshipIn
(reportsTo, person);
Person[] plist = Person:union(equals(rating,
"good"), equals(rating, "excellent"));
Person[] plist = Person:diff(equals(), equals());
Person[] plist = Person:intersect(equals(deleted,
false, equals(active, true));
Person[] plist = Person:and(equals(deleted,
false), equals(active, true));
```

SQL BIFs

```
QueryResult[] result = sql:query(selectQuery,
param_1, ..., param_n);
int updates = sql:execute(updateQuery, param_1,
..., param_n);
```

JSON Types BIFs

```
json personJsonObject = "{}";
personJsonObject.jsonPut("name", "John");
string personName = personJsonObject.jsonGet
("name");
```

Object Annotations

```
@Role("Manager")
object Manager { ... }
```

```
@Restrict("DevManager", equals(dept,
"development"))
object Employee { ... }
```

```
@RolesAllowed("Manager", "rw")
object EmployeeRecord { ... }
```

```
@NotTracked
persistent object Person { ... }
```

Function Annotations

```
@RoleName
string getGymEmployeeRoleName(GymEmployee
employee) { ... }
```

```
// Run at 2:15 AM every day
@Scheduled("15 2 * * *")
void foobar() { ... }

// Run at 06:00 every week-day  (days 1 to 5)
@Scheduled("0 6 * * 1-5")
void foobar() { ... }

// Run every ten minutes
@Scheduled("*/10 * * * *")
void foobar() { ... }

//minute | hour | day-of-month | month | day-of-
week
```

```
@ReceiveSms("Test description")
void receiveSmsNumberText(string number, string
text) { ... }

@ReceiveSms("Test description")
void receiveSmsObjectText(Nurse nurse, string
text) { ... }

@ReceiveSms("Test description")
void receiveSmsObjectNumberText(Nurse nurse,
string number, string text) { ... }
```

```
@OnSmsResultUpdate
void smsResultUpdateCallback(__sms_result__
smsResult) { ... }
```

```
@OnScheduledFunctionResultUpdate
void scheduledFunctionResultUpdateCallback(
   __scheduled_function_result__
scheduledFunctionResult) { ... }
```

```
@OnPaymentUpdate
void paymentUpdate(uuid id, PAYMENT_STATUS status,
string message){
   Mez:log(Strings:concat("Updated Payment ", id,
status, message);
}
```

```
@POST("v1/farmer/profile/documentation")
void postFarmerDocumentation(FarmerDocumentation
farmerDocumentation) {...}
```

```
@ResponseExclude("middleName")
@ResponseExpand("crops")
@GET("v1/farmer/mobileNumber/{mobileNumber}")
Farmer getFarmer(string mobileNumber) {...}
```

```
@PUT("v1/farmer")
void updateFarmer(Farmer farmer) {...}
```

```
@DELETE("v1/farmer/mobileNumber/{mobileNumber}")
json deleteFarmer(string mobileNumber, bool purge)
{...}
```

Object Attribute Annotations

```
persistent object Person
{
    @FirstnameValidator("validr.msg.fname")
    string fname;
...
```

```
persistent object Person
{
    string name;

        @OneToOne
        IdentityDocument idDoc via person;

    @OneToMany
    Vehicle vehicles via owner;

    @ManyToOne
    House home via residents;

        @ManyToMany
        Person children via parents;
}
```

Atomic Validators

```
validator PersonNameValidator {
    notnull(); minlen(2); maxlen(50);
}
```

```
validator BondRepaymentMonths {
        notnull(); minval(0); maxval(72);
}
```

```
validator CTMetroPhoneNumber {
    notnull(); regex("021-[7..9]{7}");
}
```

```
validator EmailValidator {
        notnull(); regex("\b[A-Za-z0-9._%-]+@[A-Za-
z0-9.-]+[.][A-Za-z]{2,4}\b");
}
```

View Widgets

button

```
<button label="button.exec" action="exec" />
```

submit

```
<submit label="submit.exec" action="exec" />
```

action

```
<action label="action.exec" action="exec" />
```

info

```
<info label="info.shop_owner_firstname">
    <binding variable="shop">
        <attribute name="owner.firstName"/>
    </binding>
</info>

<info label="info.welcome" value="info.
system_admin_welcome"/>
```

textfield

```
<textfield label="my.textfield.label" datatype="
text">
    <binding variable="MyUnit:string_value" />
</textfield>
```

textarea

```
<textarea label="my.textfield.label">
    <binding variable="MyUnit:string_value" />
</textarea>
```

datefield

```
<datefield label="my.datefield.label">
    <binding variable="MyUnit:date_value"/>
</datefield>
```

invite (legacy; use invite BIF)

```
<invite label="my.invite.label">
   <binding variable="MyUnit:user_object"/>
   <role>MyUserRole</role>
</invite>
```

fileupload

```
<fileupload label="my.fileupload.label">
    <binding variable="MyUnit:object">
        <attribute name="attributeName" />
    </binding>
</fileupload>
```

filebrowser

```
<filebrowser label="my.filebrowser.label">
    <collectionSource function="MyUnit:values" />
</filebrowser>
```

map

```
<map lat="item_lat" long="item_long">
   <collectionSource variable="MyUnit:map_data">
   <markerAction label="Mmy.map.action.open"
action="MyUnit:open_item"/>
   <markerTitle value="MyUnit:get_title"/>
   <markerIcon value="MyUnit:get_icon"/>
   <markerDesc value="MyUnit:get_desc"/>
</map>
```

code

```
<code title="code_title.code_snippet">
    <binding variable="QuestionManagement:question"
>
        <attribute name="codeSnippet"/>
    </binding>
</code>
```

checkbox

```
<checkbox label="my.checkbox.label">
   <binding variable="MyUnit:bool_value" />
</checkbox>
```

select

```
<select label="my.select.label">
    <binding variable="MyUnit:selected_value" />
    <enum>SDS_SADSA</enum>
</select>
<select label="my.select.label">
    <binding variable="MyUnit:selected_value" />
    <collectionSource variable="MyUnit:values"/>
</select>
```

table

```
<table title="my.table.title" defaultSortColumn="
1" defaultSortDirection="descending" csvExport="
disabled">
    <collectionSource variable="MyUnit:plist"/>
    <column heading="Firstname">
        <attributeName>fname</attributeName>
    </column>
    <column heading="Surname">
        <attributeName>sname</attributeName>
    </column>
</table>
```

menuitem

```
<menuitem icon="icon_name" label="my.menu.item">
   <userRole>MyUserRole</userRole>
</menuitem>
```

navigation (legacy; use navigation described earlier)

```
<navigation outcome="success" target ="myview_name"
/>
```

wall

```
<wall label="my.wall.label" defaultSort="
timeStamp" commentHandler="MyUnit:commentFunction"
buttonLabel="my.custom.label.for.comment.button">
    <collectionSource function="MyUnit:collection"
/>
    <itemTitle value="MyUnit:getTitle"/>
    <itemText value="MyUnit:getText"/>
    <itemOwner value="MyUnit:getOwner"/>
    <itemTime value="MyUnit:getTime"/>
    <itemIcon value="MyUnit:getIcon"/>
</wall>
```

gallery

```
<gallery title="my.gallery.heading"
imageAttribute="image" descriptionAttribute="
description">
        <collectionSource function="
getObjectsWithImageBlobs"/>
        <binding variable="selectedObject"/>
</gallery>
```

Dynamic Content

Dynamic Widget Titles/Labels

view

```
<table title="translation.key">
   ...
</table>

<submit label="translation.key">
```

en.lang

```
 translation.key = {UnitFoo:globalVar}
```

presenter

```
Unit UnitFoo;

string globalVar;

init() {
   globalVar = getLabel();
   ...
}
```

Dynamic Widget Visibility

view

```
<table title="table.label">
        <visible variable="isTableVisible"/>
        <collectionSource variable="items"/>
        <column heading="col.name.label">
                <attributeName>name</attributeName>
        </column>
        <column heading="col.desc.label">
                <attributeName>description<
/attributeName>
                <visible variable="isDescVisible"/>
        </column>
</table>

<checkbox label="have.at.least.one.visible.widget">
        <visible variable="isCheckboxVisible"/>
        <binding variable="checkboxVar"/>
</checkbox>
```

presenter

```
bool checkboxVar;
void init() {
        checkboxVar = getCheckboxVar() {
        ...
}
```

Built-in Objects

object	about	usage sample

object Identity {
  string _firstNames;
  string _nickName;
  string _surname;
  string _locale;
  string _timeZone;
}
	Implicit interface that is implemented by every @Role persistence object.	@Role("Person")
		persistent object Person {
		}
		void test() {
		  Person p = getRandomPerson();
		  string name = p._firstNames;
		  Identity i = p;
		  string sameName = i._firstNames;
		}

@NotTracked
persistent object
__sms_result__ {
  datetime
datetimestampStarted;
  datetime
datetimestampFinished;
  string destination;
  int attempt;
  bool success;
  string error;
  bool doneProcessing;
  uuid
smsOutboundConfiguration
VersionId;
  string
smsOutboundConfiguration
Name;
  uuid smsId;
}
	SMS results from Helium duplicated to the app schema.	void test() {
		  __sms_result__[] results = __sms_result__:all();
		}
		
		void test() {
		  __sms_result__[] results = __sms_result__:and(
		  equals(success, false),
		  equals(doneProcessing, true));
		}

@NotTracked
persistent object
 	
__scheduled_function_res
ult__ {
  datetime
datetimestampStarted;
  datetime
datetimestampFinished;
  string qualifiedName;
  string schedule;
  string error;
  string stackTrace;
  bool success;
}
	Scheduled function results from Helium duplicated to the app schema.	void test() {
		  __scheduled_function_result__[] results =
		  __scheduled_function_result__:equals(
		  qualifiedName,
		  "SomeUnit:SomeScheduledFunctionName");
		}

@NotTracked
persistent object
MezBatch {
  string name;
  string header;
}
	Used by Helium to create a CSV batch for later processing.	MezBatch stockUpdateBatch = Mez:createBatch(fileUpload._id,
		fileUpload.data);

@NotTracked
persistent object
MezBatchItem {
  bool processed;
  string value;
  @ManyToOne
  MezBatch batch via
batchItems;
}
	Used by Helium to create CSV batch items for later processing. Each record represents a line from a CSV file and is related to a MezBatch.	MezBatchItem[] stockUpdateBatchItems = MezBatchItem:
		relationshipIn(batch, stockUpdateBatch);
		StockUpdate[] stockUpdates;
		
		for(MezBatchItem stockUpdateBatchItem:
		stockUpdateBatchItems) {
		  StockUpdate stockUpdate = StockUpdate:fromCsvLine
		(stockUpdateBatch.header, stockUpdateBatchItem.value);
		  stockUpdates.append(stockUpdate);
		  stockUpdateBatchItem.processed = true;
		}