SYSTEMS ENGINEERING

DESKBOOK

GUIDE 11 · DXL ESSENTIALS · QUICK REFERENCE

THE ONE SAFETY FACT

A DXL write **commits to the database the instant the script runs** — no confirmation, no preview, no undo.

The baseline you take first is the only way back.

01 THE FOUR OPERATIONS

SPINE OF NEARLY EVERY INHERITED SCRIPT · CODE REPRODUCED VERBATIM

i. ITERATE · GUARDED LOOP

Walk every object; *guard* so it skips soft-deleted ones.

```
Object o
for o in current Module do {
  if (!isDeleted o) print identifier(o)
"\n"
}
```

ii. READ AN ATTRIBUTE

Object, dot, attribute name in quotes. Read-only.

```
string vm = o."Verification Method" ""
```

Trailing "" coerces the value to a string.

iii. MODIFY · THE WRITE

Looks almost identical to a read — read carefully.

```
o."Verification Method" = "Test"
```

Committed instantly. No undo.

iv. GENERATE OUTPUT

Print to the pane, or open a *Stream* to a file.

```
Stream out = write "C:/temp/gap.txt"
out << identifier(o) << "\n"
close out
```

02 READ THE COLORS

TOKEN KEY

- **Function** · isDeleted, print
- **Attribute** · "Object Type"
- **String / number**
- **Type** · Object, Stream
- **Comment**
- **Keyword** · for, if, do

`// · /* */`	Comments — where the last author told you the intent.
`for…in…do`	The loop, body in braces. Walks every object.
`o."Name"`	Attribute access — the read; `""` coerces to string.
`Object · Module`	Types: also string, int, bool, Stream.

Name these on sight and most inherited scripts stop being opaque.

03 == VS =

MOST DANGEROUS

== **Compares**
a filter — belongs inside an `if`

= **Assigns**
a write — changes the database

One keystroke apart, opposite effects. A **= typed where == was meant** turns a filter into an always-true overwrite — and DOORS gives no warning. Read every = inside an `if` as a suspected bug until proven otherwise.

04 READ BEFORE YOU RUN

3 QUESTIONS

1. Find the **loop** — the `for … in … do` — and what module it walks.
2. Find what it **filters** on; confirm each test uses `==`, not `=`.
3. Find every line that **writes** (= on an attribute) or **removes** (`delete`, `purge`).

Only `prints` → **safe to run.** Any = on an attribute or any `delete` → **it changes the database.**

05 INTERACTION WINDOW

EDIT DXL…

DXL Interaction — two-pane editor

INPUT — YOUR DXL

```
// which module?
print name(current Module) "\n"
```

OUTPUT

```
01-UAS-System-Requirements
```

▸ RUN

Runs against `current Module` — the front window. **Check the title bar first.**

06 WHERE IT LIVES

EITHER WAY

.dxl **A loose file**
In a program folder — load or paste it into the editor.

▸ **Installed on a menu**
One click from running; source sits in the DOORS add-ins folder.

A runbook line
"Run the coverage script before the review."

Wherever it comes from, the questions you put to it are the same.

07 FIVE FOOTGUNS

SYMPTOM → FIX

Write with no undo
Assigned an attribute, never baselined. → **Baseline first.**

Loop hits deleted objects
No `isDeleted` guard. → **Guard `if (!isDeleted o)`.**

Wrong module changed
Ran against the wrong `current Module`. → **Check the title bar.**

=-for-== slip
Filter became an overwrite. → **Read every = in an `if` as a bug.**

`delete` / `purge`
As final as a write. → **Baseline first.**

08 SAFE-WRITE ORDER

IN SEQUENCE

1. **BASELINE**
   File ▸ Baseline ▸ New… — the only undo a write has.
2. **SCOPE**
   Narrow the loop to only the objects you intend.
3. **RUN**
   Press Run against the baseline you just took.
4. **VERIFY THE COUNT**
   Check the count printed against the count you expected.

A number **larger than your scope** *is the tell.*

09 PRE-RUN CHECKLIST

EVERY RUN

- ☐ Read the whole script — you know whether it **writes**.
- ☐ If it writes, the module is **baselined now**.
- ☐ `current` is the module you mean — **title bar confirms**.
- ☐ The loop is **scoped** and guards `isDeleted`.
- ☐ You know the **count** it should change, and will verify it.
- ☐ Every = inside an `if` is **cleared as not-a-bug**.