**Run custom X++ scripts with zero downtime**

This feature lets you upload and run deployable packages that contain custom X++ scripts without having to go through Microsoft Dynamics Lifecycle Services (LCS) or suspend your system. Therefore, you can correct minor data inconsistencies without causing any disruptive downtime.

All deployable packages that are uploaded into the system go through a mandatory workflow. As a safety precaution, and to help ensure segregation of duties, the user who uploads a deployable package isn't allowed to approve it for the next steps in the workflow. Another user must approve it. However, after the package is approved, the user who uploaded it will be allowed to complete the remaining steps.

The system requires that all deployable packages go through a test run. Before the script will be allowed to run on production data, a user must validate that the output is correct by selecting **Accept test log**. If the output isn't correct, the user must mark the package as failed by selecting **Abandon**. In this case, the script won't be allowed to run on production data.

Every uploaded package is saved in the system and goes through a defined workflow of events. For each event, the system keeps a log that includes a timestamp and the identity of the person who performed the event. In this way, the system ensures that there is an audit trail.

Your deployable package must contain exactly one runnable X++ class. In other words, it must have one class that includes a method that has the following signature.

The following code example shows how a deployable package can be structured.

[Graphical user interface, application

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEjw1kS3i4E3A3euf8FapIzpuZHnKWxbxUVMks1SxYWi3EKwtGFMLAKSdYE0hmFrBXXefUzugzHZSd5p7b8JjUon1tSwYolNRUieKkPCe7iGr29d_s6HJ6-EgJO51C-uJDU_e-1b_4UxB_MIOQAy3nxdHa_OzQNR5D34wNPMSQiZnLbNog-Ns9KBnNFJ)

**CODE EXAMPLE**

The following code example shows how a deployable package can be structured.

[Graphical user interface, text, application, email

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEhrfPPWY1k2l2ru1Pc8dSlv1S53SroCpegZ97bor-TwaDuZbm_XvmbcN0qmH_ljH9BZF1ArhUN47D_IQylHZkxfIgPpYgyGKHW-QnvBG59Ucor3Iti6Bw0J8C_DLz8IhpMK1-lIC6Abh-zTAvofCYd5ElwBnPLePfFj1LPgXkMBlfK-LuckKZ7NsNSq)

**BEST PRACTICES**

The following list describes some best practices for successfully writing, implementing, and running a script. The list isn't exhaustive, and it should be considered only guidance.

·        **Do** write a success message at the end of the script. In this way, you will be able to see that the script ran without exceptions.

·        **Do** add explicit handling of the transaction scope.

·        **Do** use existing business logic, such as update() methods, but **do not** bypass business logic by using doUpdate(), doInsert(), and doDelete() methods. This approach will help ensure that dependent data is handled correctly. It will also significantly reduce the risk of further data inconsistencies.

·        **Do** assert the company context. This approach will expose common mistakes as a script runs. For example, it will reveal whether the script is being run in the wrong company.

·        **Do** assert that the number of affected records matches your expectations. This approach will reveal whether data unexpectedly shifted in the system while the script was being prepared.

·        **Do** use unique class names for each script (for example, by including a reference to a work item in the name). This approach will prevent name clash issues when you upload the script. If a new iteration of a script is required, be sure to give it a new name.

·        **Do** test each script in a non-production environment first. Test for the intended impact and for unintentional side-effects on related data. Ensure that all business processes that might be affected can be successfully and fully completed afterwards.

**CREATE A DEPLOYABLE PACKAGE**

[Graphical user interface, application

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEhNoJnFp8Gm-Ju1nuMqFH7pzLXQ4deCznDUXlGCtKwRpSPZEomVBfjNuezVtgsBHpFu3ejQUB5A2AKes_oJQC66-HNsY6PXGKN-XcE8ZD3OBWNJvLskkgKc98GaL1BFistMKZSz_AWyjV-YBaa662vgMofQzDwR9zsMESjiF4bwf8vf_YZGOmMCJuTl)

**ASSIGN DUTIES TO USERS TO CONTROL ACCESS**

This feature provides the following duties. Admins can use these duties to help control access to the feature.

·        **Maintain custom scripts** – This duty grants the ability to upload, test, verify, and run custom X++ scripts in environments (user acceptance testing [UAT] and production).

·        **Approve custom scripts** – This duty grants the ability to approve an uploaded custom X++ script. Approval is a mandatory step before any script can be tested, verified, and run.

**UPLOAD AND RUN A DEPLOYABLE PACKAGE**

Use the following procedure to upload and run a script.

    1.     In your Finance and Operations app, go to **System administration > Periodic tasks > Database > Custom scripts**.

    2.     Select **Upload**.

[Graphical user interface, text, application

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEisIun1rZj7RrATab4z3_OVTloMT1snnZOcahBUUEW6nSiYd7sRB3ea9P2ZWMySsBkuHQwHV_wj4_TBIxExi-AkftJqSLPCZtzAzhs7U0zO3CiGx4Yl1-LwUDAYKaQOK-AKIeCT_dA4j9L4B_lDhl11vDp-2Ipd6BluDYJKFx06-uFW5s6h_nCifUFd)

3.     Select the deployable package that you created as described earlier in this article. You will be prompted to specify the purpose of the script.

o

[Graphical user interface, application

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEj2hs9Ed9WHQix3tHsTQpaBXL-K3QX-Tq6Yp57NSaXhc3ZwlyhwIN8bwGNad5V6FWooHX5pkUTYts5udGNC0N-U_0_HhvHMgMreSyr2KlGfKCGBGi9LMIF7nkdCY1pPqjMxPmva8cLztyMbdUr7GRFJxVB-UeFn5pYaD6ZMDsud2kCoxMDl6YyC0wQY)

The script must now be approved by a user other than the user who uploaded it. The approver must follow these steps:

1-    Go to **System administration > Periodic > Data base > Custom scripts**.

2-    Select the script to approve, and then select **Details**.

3-    On the Action Pane, on the **Process workflow** tab, in the **Start** group, select **Approve** or **Reject**. If you select **Approve**, the script is marked as approved and is unlocked for testing. If you select **Reject**, the script is locked. In both cases, the event is logged, and a copy of the script is kept in the system.

[Graphical user interface, application, email, Teams

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEik7D7K55azyBgOBF0Dx8zQ9F1yCv3MTpuSWwqtRlKo7WB15lraGUJJ-hJslYocgUGxjePbAPHkdPbLUMwzewyy7cJmTrL6Qo-i4NgS8m7utGCZgzsvOzAPZ1GeLxHaxRLrK_qea7wbrxI2VlgmU_ocoM-Wj-88CWN2SdfAE33Fqa_62EAbqpNQSTB4)

[Graphical user interface, text, application, email

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEifGEDkqKeWGm127Y83uaTRQFl1CyKbevcbTXqIEI3xFT0mBJAPhgUm8I-gy0eravmQdbUjypvt_WHUtypfmHt6BcRn4QwoxHRowLQozWri5pRPe5TC2DzTXTGjNcd3Lnq-J8Sgpds6INQP2Eb4tZEYnRCn02GqHjwkaXNMHGxxTW9m9OQ4B05IVmUB)

     The script must be tested to ensure that it does what it's intended to do. The tester can be the same as the uploader or the approver, or it can be a third user who has the required permissions. The tester must follow these steps:

1.     Go to **System administration > Periodic > Data base > Custom scripts**.

2.     Select the script to test, and then select **Details**.

3.     On the Action Pane, on the **Process workflow** tab, in the **Test** group, select **Run test**. The script is run inside a temporary transaction that the system will automatically abort while it collects various logs and SQL statements.

4.     When the script has finished running, review the logs, and verify that the results meet your expectations. Follow one of these steps:

§  If you're satisfied with the test result, select **Accept test log** in the **Test** group on the **Process workflow** tab of the Action Pane to allow the script to be run. The event log will reflect the fact that the script was tested, and it will indicate who tested it and when.

§  If you aren't satisfied with the test result, select **Abandon** in the **End** group on the **Process workflow** tab of the Action Pane to prevent the script from being run. The system will keep a copy of the script together with a log of its history.

[Graphical user interface, application, Teams

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEhpiMMoMh8rK9UFW-dBKslNJgoe5dm-qNpcP6pLHz4b96DlwM7d2QzqJGi0BgMjWucvethGHmXnaMoeo7huQiYdqZTJMZ3cOoZpX22Wef8XSRfvV7fztS9XMeblxqur03htpgdJoiu9grEd5Th8Fv8wnYcXBTaXgEGMUFW62Q-sO8vgQVCN4Dj5Z0Vm)

[Graphical user interface, text, application, email

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEgb0veSu3jlQS1VaZ6v8jwczndrxOrwV27UcTKBSCg90nruX5AXvE7QDXKLkntNqGpyepq3AL38P-jfXX5rRZ4LVsHO0TSLZ6TwxGCucPAc2B8tHJvkEgWsz52MJ4aju1cBL6PJWG7kWsHLKP65mdkctChmPcws5tOcXEMkE5fqRbr6uC4JS1FrVqEk)

1.     When you're sure that the script meets your expectations, select **Run** in the **Run** group on the **Process workflow** tab of the Action Pane to run it. This command does the same thing as the previous test run, but the transaction will be committed at the end.

2.     After the script has finished running, check the result, and confirm that the script worked as you intended. Follow one of these steps:

o   If you're satisfied with the result, select **Purpose resolved** in the **End** group on the **Process workflow** tab of the Action Pane. The event log will reflect the fact that the script ran successfully, and it will indicate who verified the script and when. The script is saved, but it's now locked and can't be run again.

o   If you aren't satisfied with the result, select **Purpose unresolved** in the **End** group on the **Process workflow** tab of the Action Pane. The event log will reflect the fact that the script failed to fulfill its intended purpose, and it will indicate who ran the script and when. The script is saved, but it's now locked and can't be run again. However, the system doesn't automatically undo the script action. You might have to write, import, and run a new script to undo the effect that the failed script had on your system.

Your selection in the last step defines the final state for the script. You can repeat the process as you require.

[Graphical user interface, text, application

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEj1l8wZxFCCzc3nc8iO2otwlZKtzGwCby_ake9yNvV-yr6m38s2Zj-iaDXDHhSFV3wnYhpHa-4xSls5uQaIYIUPAm1cV0Y2ndkJ23CE1cFPznI54P3C0bq5J51_npV_un6igmtua4E66BkgHDFu9S7ArWsk82kNsCfBUxTp-w0r9wFSpXll9DaJ3_HQ)

[Graphical user interface, text, application, Teams

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEgta33v072hKR9TzJIvz79OEYDWV97Syk7Kbuoh82XByl8_ttiHLL84uGkBY6fjrHWfVkqcsNgrXte3QbGOibuz50Y5UAlk9LCP-BMfFcVnTxfBzjR-JTGpZ4Dd1j99Ble_WFafoZkebf6kVG18MHVo-7OENgkkKgR8YlmZ6eFCDkCcJgcxumTU8W5B)

[Graphical user interface, text, application, email

Description automatically generated](https://blogger.googleusercontent.com/img/a/AVvXsEhcZzDIIxJ599Z3Fr1W9JeifVoPKRI0Mx8MHg9v7Rw54Lp_dy3CTmbAbdzT7cneQIKSEuF1lmXSZm9waDP1MLDjiNKidaux9Ov0oitkxLCKopAoa20rDud6IOcLLIdLeOmOgP2GuPC-uvluYAwSQoQyhtycyz5ZO0iHjjdUWwYSCHo2C4a0Ui6Tokpi)