eXtended Productivity Facility

Implementation Guide

Version 6.1.0

June 2016

improvIT Software Innovations GmbH
Große Elbstraße 141 a
22767 Hamburg

Telephone:	+49 (0)40 540 90 29 - 7
Fax:	+49 (0)40 540 90 29 - 9
Email:	Contact@improvIT-Software-Innovations.de
Web:	www.improvIT-Software-Innovations.de

This page intentionally left blank

I. Content

II. Graphics

1. Introduction

It was recognised that a solution was required to solve the installation, maintenance and support problems caused by applications running within the IBM z/OS/ TSO/ISPF environment.

The IBM supplied services allow a certain amount of flexibility, but require that every site creates a bespoke solution. Additionally the integration of ISV products had to be handled.

As a result, the product “eXtended Productivity Facility (XPF)“ was developed. XPF solves the problems by offering an integrated solution that allows an easy method of maintaining applications and their required resources. The meta definitions are stored within a repository and when required are processed by XPF. The normal IBM services are encapsulated and the end user does not require technical knowledge.

Additionally XPF also offers application runtime control functions which are not normally available within TSO/ISPF environment as well as enhancing the TSO/ISPF user environment.

Existing z/OS maintenance concepts can also be supported.

1.1. Historical Difficulties

Every IBM z/OS installation was required to develop and maintain its own methods and procedures to allow the integration and maintenance of applications within the TSO/ISPF environment.

The solutions need to handle the logon phase, native TSO and TSO/ISPF applications as well as batch requirements. ISV[1] software also needs to be integrated.

In many cases the procedures supplied by independent software vendors are also used. These generally do not use the newer functions of the TSO/ISPF services, often cannot handle individual site standards and each ISV only sees their own environment. Collisions between applications are therefore unavoidable and can result in an unstable environment.

As a result each site has to spend a large amount of time maintaining the TSO/ISPF environment.

Until now, there was no standard software solution available which allows a simple and centralised approach to maintaining the runtime environment of TSO based applications.

[1] Independant software vendor

1.2. TSO/ISPF

In a large number of z/OS installations, TSO/ISPF is not treated as a production environment. In other subsystems, application are centrally managed and coordinated. Within TSO however, everybody is free to "dabble" and change the environment as they wish. Although a customised user environment is important, it must not interfere with production systems and applications. Other users must not be affected when a single user encounters a local problem and a TSO logon should always be possible without system programmer intervention. XPF permits a high level of application control and isolation without stopping users creating individual environments.

TSO/ISPF is an important production environment and should be treated as such!

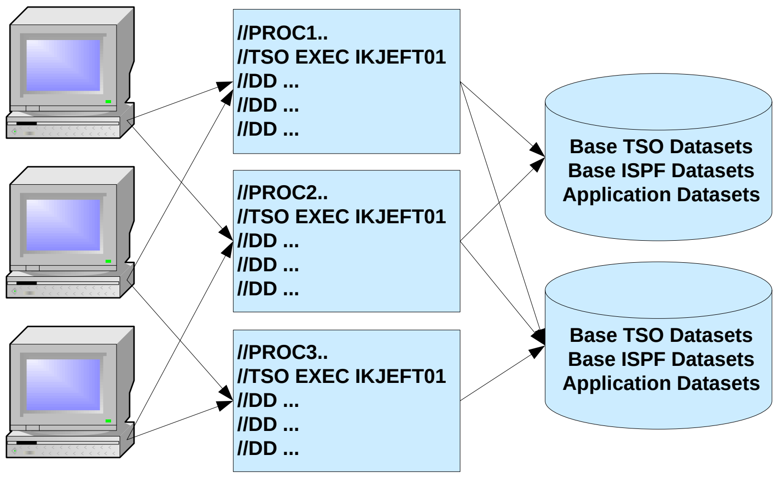
2.Managing TSO/ISPF

This chapter will show a number of methods in use today by various z/OS installations to manage their TSO/ISPF environments and the installed applications.

The corresponding advantages as well as the disadvantages are also shown.

2.1. Multiple Logon Procedures

Individual logon procedures are created and maintained for every application, groups of applications or groups of users.

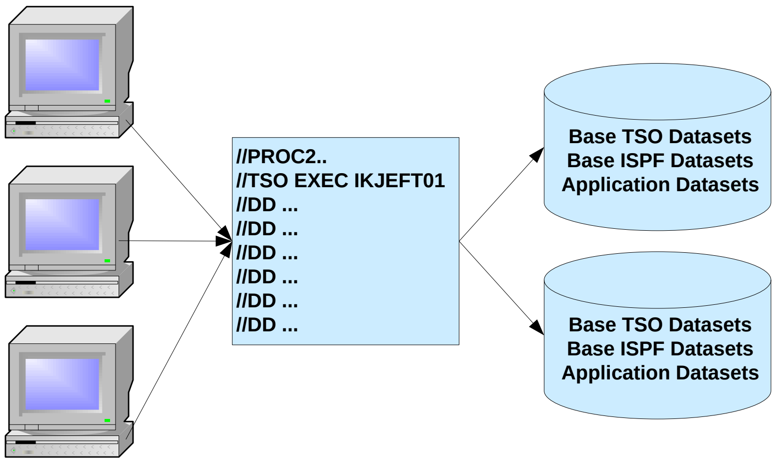


Graphic 1: Multiple Logon Procedures

- ↑ Good runtime performance
- ↑ Dataset security is possible
- ↓ No single point of control
- ↓ Release dependant
- ↓ Large maintenance overhead
- ↓ Prone to errors during maintenance
- ↓ Large number of datasets which are constantly in use
- ↓ Other required datasets must be allocated dynamically
- ↓ Difficult to control which logon procedures are in use
- ↓ Not user friendly as a lot of logon procedures need to be used
- ↓ Application security difficult

2.2. Single Logon Procedure

Only one logon procedure is in use, in which all required datasets are pre-allocated for all users and all applications.

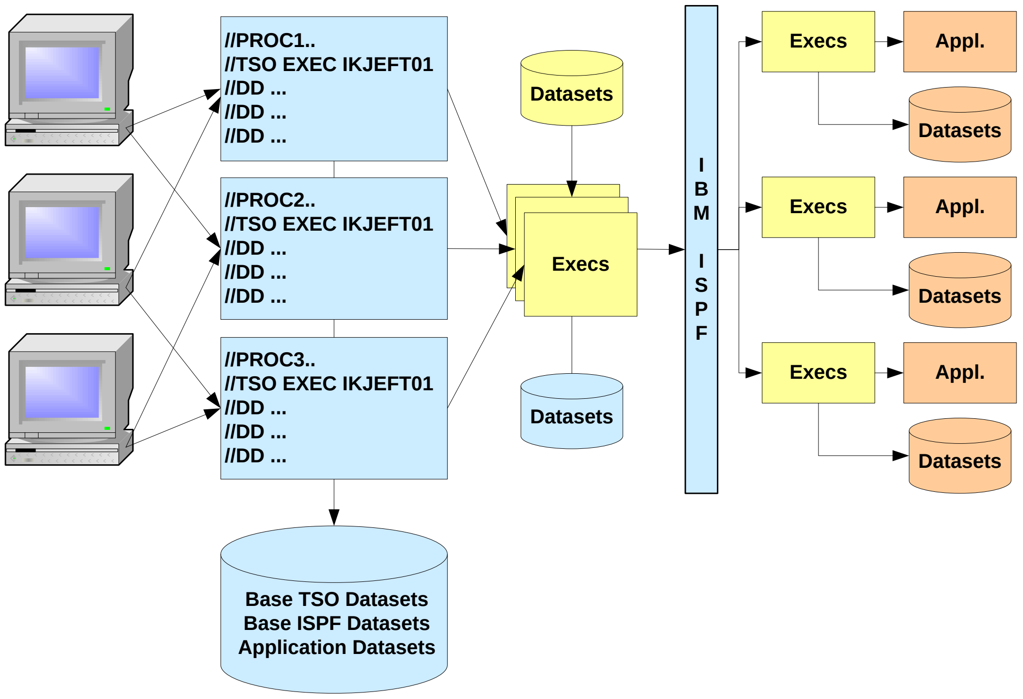


Graphic 2: Single Logon Procedure

- ↑ Single point of control
- ↑ Small maintenance overhead
- ↓ Bad runtime performance
- ↓ Release dependant
- ↓ Large number of datasets which are constantly in use
- ↓ Other required datasets must be allocated dynamically
- ↓ Dataset and application security very difficult

2.3. Pre- and Dynamic Allocation

A small number of logon procedures are used to allocate base datasets. These then call various execs to allocate additional datasets. Which execs and datasets are used, is generally controlled by runtime information (e.g. Userid, RACF group, ...). After the logon has completed, other software and application datasets are dynamically allocated if needed.



Graphic 3: Pre- and Dynamic Allocation

- ↑ Less release dependent
- ↑ User friendlier
- ↑ Dataset security is possible
- ↓ No single point of control
- ↓ Large maintenance overhead
- ↓ Prone to errors during maintenance
- ↓ Poor performance
- ↓ Small runtime overhead
- ↓ The same datasets are generally maintained in multiple procedures and execs
- ↓ Dataset conflicts between applications
- ↓ Some datasets are constantly in use
- ↓ Application security difficult

This method tends to be the most widely used !

3. Using eXtended Productivity Facility

This chapter will explain the concepts behind the eXtended Productivity Facility (XPF) and demonstrate how the TSO/ISPF environment & applications are managed by XPF and the resulting advantages.

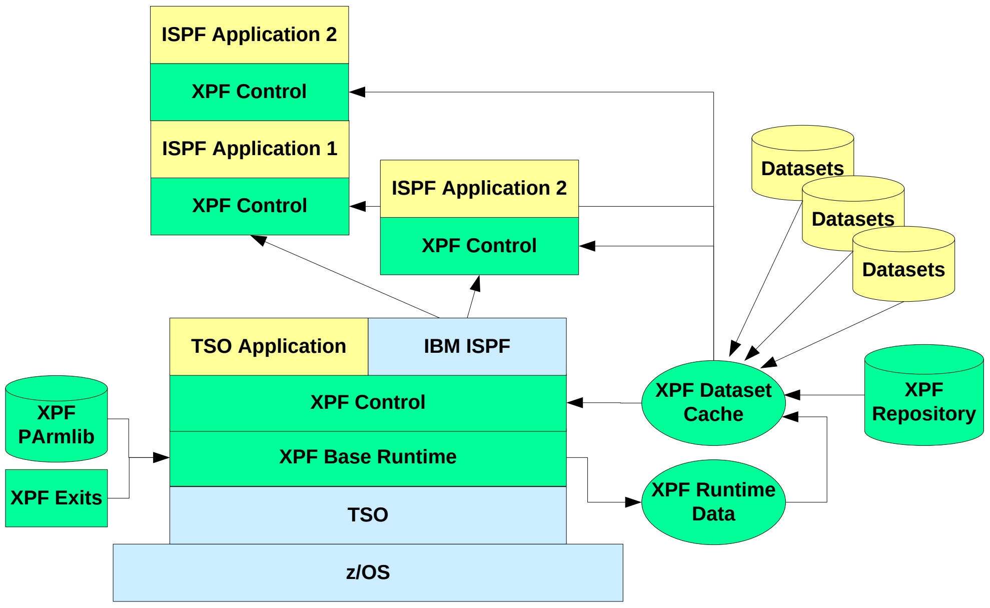
3.1. Why XPF?

XPF offers the following features and advantages:

– A single TSO logon procedure for all users and applications
– Single Point of Control for all TSO and ISPF applications
– Online and batch support
– Datasets are only allocated when required
– Support of existing z/OS maintenance concepts
– Security through SAF
– Sysplex Support
– Improved performance within TSO/ISPF
– Easy to use
– Flexible definition of application data
– Dataset definitions using generic names
– Easy customisation of individual user environments
– Lower workload for system programmers
– Additional ISPF edit macros and tools
– A very powerful but inexpensive product

3.2. TSO & ISPF using XPF

Only one logon procedure is used by all users for all applications. No datasets are allocated by this logon procedure. All system and application datasets are defined within the XPF repository and are only allocated by XPF if required. All other application attributes are also stored within the repository.

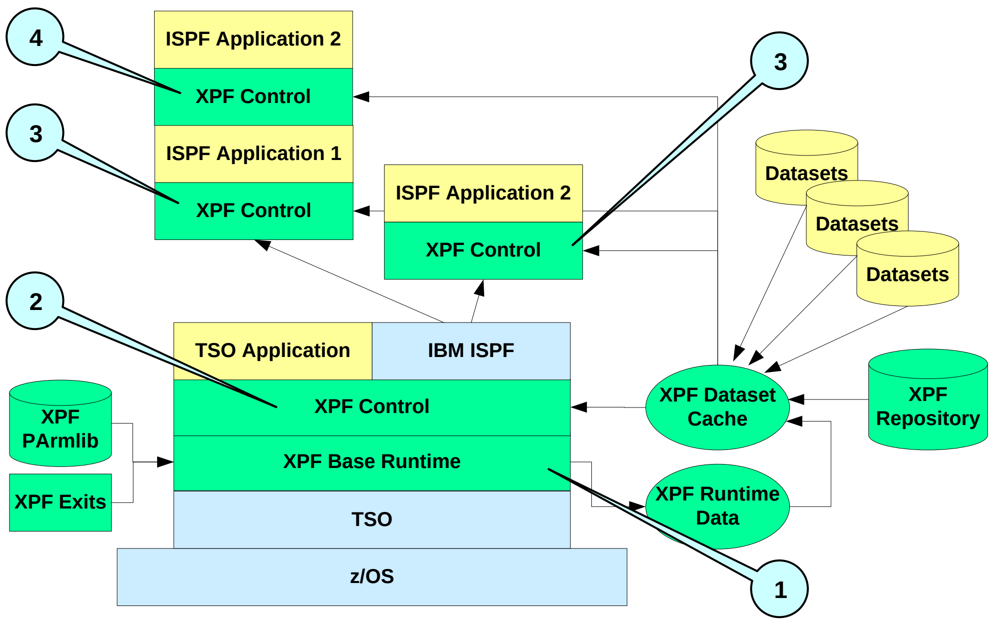


Graphic 4: TSO & ISPF using XPF

- ↑ Single point of control & maintenance
- ↑ Very small maintenance overhead
- ↑ Single point of failure after changes
- ↑ Sysplex support
- ↑ Multi-layer repository
- ↑ Easy integration into change management systems
- ↑ Release independent application definitions
- ↑ Support of parallel application releases
- ↑ Dataset allocations only if required
- ↑ Allows a similar approach as in batch - datasets are only allocated when a program is used and not permanently
- ↑ Better performance
- ↑ User customisation possible
- ↑ Lower workload for system programmers
- ↑ Easy security integration
- ↑ User friendly
- ↓ Small runtime overhead

4. XPF Control

This chapter will demonstrate the way XPF controls the environment and the applications. Additionally all XPF components and the architecture are explained.



Graphic 5: Process flow when using XPF

The following diagram shows the process flow when using XPF. Each step is described in detail on the following pages.

1. When a TSO logon takes place, control is immediately passed to XPF. The runtime environment is created, the parmlib is processed and if required the start-up exits are called. All allocated application DDNames are also freed. If processing was successful, then the first application Gateway is started. Gateway is the name used for an application definition. It contains all relevant application information and a list of required datasets.
2. When a Gateway is started, XPF first checks whether the required data is already stored in the cache. If it was not found, then the repository is searched and when located, the application data is loaded into the cache. Afterwards all defined datasets are allocated and the application is started. The repository consists of up to 4 levels (User, Group, System & Global). It is therefore possible to create application definitions (in addition to the global environment) for individual users and work groups. An implicit 5th repository level (current Gatelib) may also be activated.
3. A further application needs to be started. The same processing takes place as in step 2.
4. Now an application is restarted, which has previously been used during this TSO session. The required application data is located in the cache. Only the system datasets need to be activated and the application is directly restarted. As a result the application start-up time is much faster compared with the first usage. The application remains in the cache until explicitly removed or the user terminates TSO.

4.1. XPF Repository

The Repository contains all application definition data.

The Repository is a logical concatenation of the four possible levels:

- User
- Group
- System
- Global

Each level can potentially point to a different physical dataset for every user (not recommended). The levels or physical datasets are known as Gatelibs. An application definition contained within the Repository is known as a Gateway.

An application definition is normally only read once per TSO session from the repository. Afterwards only the stored information from the cache is used (unless the information is explicitly removed).

4.2. XPF Gatelib

Up to four base Gatelibs are normally available:

- User
- Group
- System
- Global

The search sequence for Gateways is as shown above. An implicit fifth level (current Gatelib) can be activated in the XPF parmlib. This is searched before the four base Gatelibs. Additional Gatelibs can be explicitly specified. Gatelib names can potentially differ for every user (using variables).

A Gatelib is a normal partitioned dataset with the following attributes:

- LRECL 80
- PDS or PDSE

Gatelibs are **not** automatically created by XPF.

4.3. XPF Gateways

Gateways contain all required information for an application to execute. For example:

- Datasets
- Program name
- Options
- Runtime attributes

Gateways can contain variable definitions. For example:

- Dynamically constructed dataset names and commands
- System and XPF variables can be used

Every Gateway can exist once in each Gatelib. Gateways may only be maintained using the supplied ISPF dialog. Gateways may be copied or moved between Gatelibs using the ISPF dialog or a change control system (name & content must remain unchanged).

4.4. XPF Runtime Data & Cache

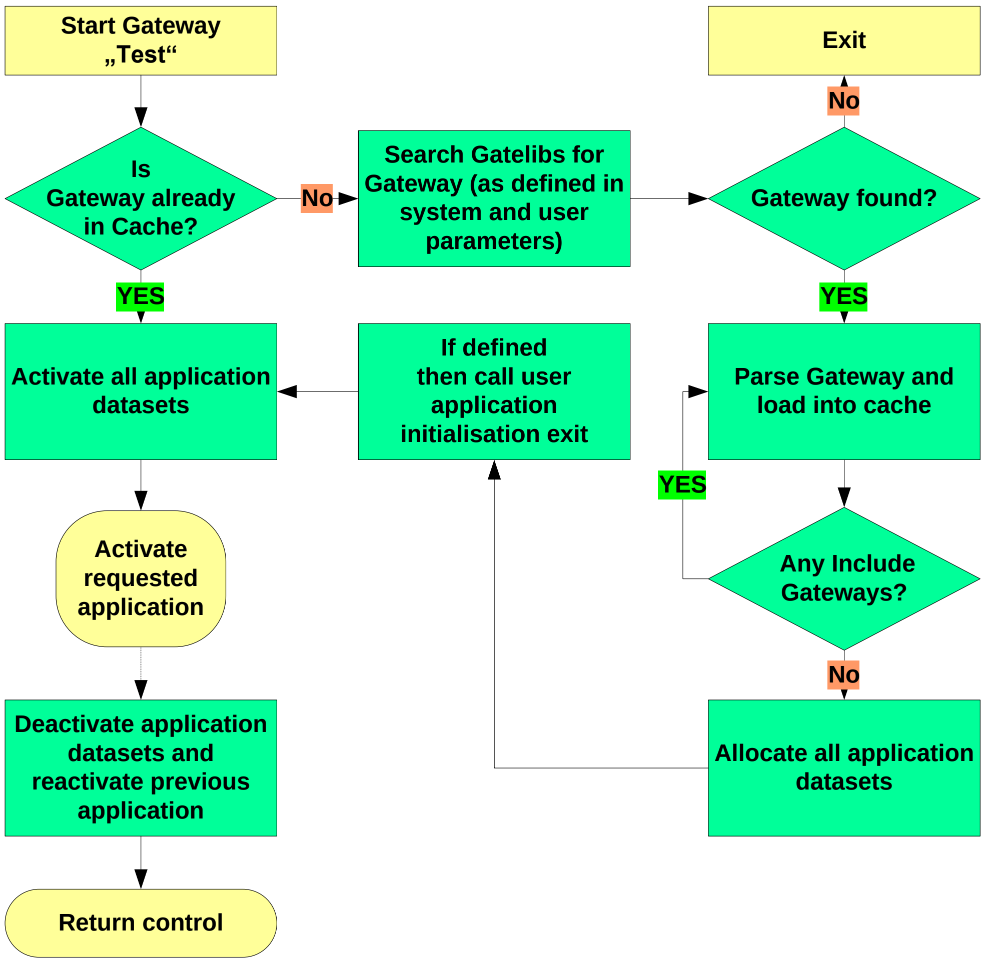
The runtime data and cache are two separate memory areas within main storage:

- Runtime Data contains application independent information
- The cache contains all application specific data

Both areas exist within the user's own address space. These areas can not be accessed or used by normal applications.

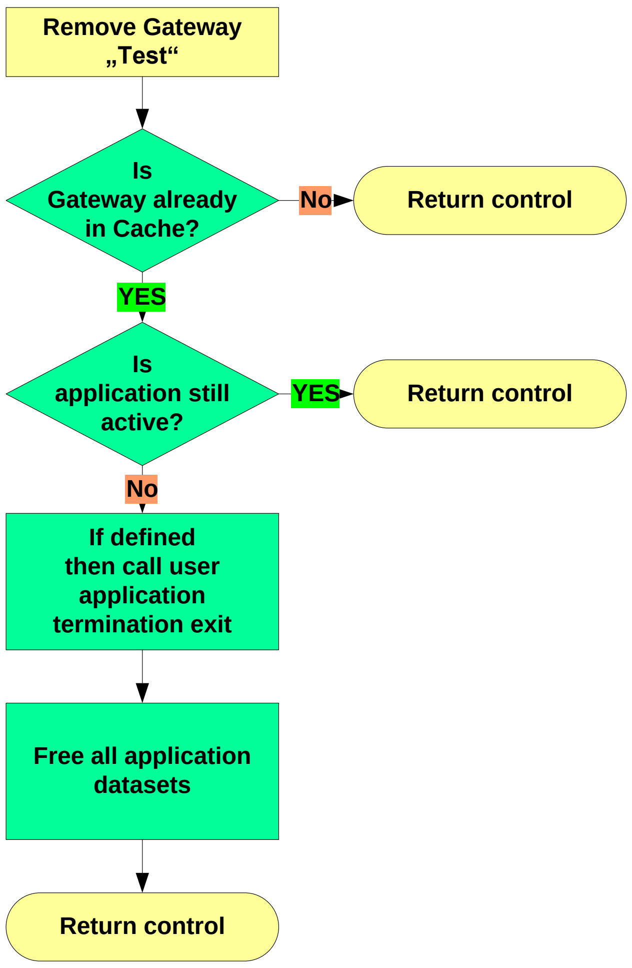
4.5. XPF Processing Flow

The following diagram illustrates the flow when a Gateway is started. This has been simplified to only show the major steps.


Start Gateway „Test“
Is Gateway already in Cache?
No
Search Gatelibs for Gateway (as defined in system and user parameters)
Gateway found?
No
Exit
YES
Activate all application datasets
If defined then call user application initialisation exit
YES
Parse Gateway and load into cache
YES
Any Include Gateways?
No
Allocate all application datasets
Activate requested application
Deactivate application datasets and reactivate previous application
Return control

Graphic 6: XPF Processing Flow

Now a simplified flow of a Gateway being removed from the XPF cache is shown.


Remove Gateway „Test“
Is Gateway already in Cache?
No
Return control
YES
Is application still active?
YES
Return control
No
If defined then call user application termination exit
Free all application datasets
Return control

Graphic 7: Simplified flow of a Gateway being removed from the XPF cache

5. Contact

For further information regarding the eXtended Productivity Facility please contact:

improvIT Software Innovations GmbH
Große Elbstraße 141 a
D-22767 Hamburg
Germany

Telephone:	+49 (0)40 540 90 29 - 7
Fax:	+49 (0)40 540 90 29 - 9
Email:	Contact@improvIT-Software-Innovations.de
Web:	www.improvIT-Software-Innovations.de

6. Index