ByStar Internet Services Operating System (BISOS) Model, Terminology, Implementation And Usage

Document #PLPC-180047
Version 0.2
July 28, 2017

This Document is Available on-line at:
http://www.by-star.net/PLPC/180047

Neda Communications, Inc.
Email: http://www.by-star.net/contact

Contents

List of Figures

1 Introduction

The Internet Services industry of today has two characteristics that greatly limit its capabilities.

First, virtually all existing Internet Services are based on the traditional proprietary software model. As yet the free software movement has no formal presence within the services domain. The solution to this is the Libre Services model, a completely non-proprietary model for delivery of Internet Services. For more information see the article titled *Libre Services: A non-proprietary model for delivery of Internet Services* [?], or visit the Libre Services information center at: http://www.LibreServices.org

Second, the Internet Services industry has arisen in a highly disorganized, unstructured way, driven by a multitude of uncoordinated commercial initiatives. The various industry capabilities have been built in an *ad hoc* manner, based on immediate business expedience, rather than by any sort of overarching engineering design. The result is the Internet Services industry as it exists today: chaotic, uncoordinated, and falling far short of its true potential.

Furtheremore, the current proprietary central model of American Internet services has taken us to live in a world where our use of the network is mediated by organizations that often do not have our best interests at heart.

The solution to this is the By* family of Libre Services. By* (pronounced "by-star") is a coherent, scalable, generalized Internet Services model. By building software that does not rely on a central service, we can regain control and privacy. By keeping our data under our own real control either by maintaining possession of our data in our homes or through remote trusted possession of our information, we gain useful legal protections over it. By giving back power to the users over their networks and machines, we are returning the Internet to its intended end-to-end and peer-to-peer architecture.

Together, the Libre Services and By* models have enormous implications. The Libre Services development model, and the By* unified services model, can transform the Internet completely, from the proprietary and *ad hoc* model of today into something far more powerful.

The realization of this potential is large, complex and ambitious. It is far too large in scope to be accomplished by any one company acting alone, but instead can only be accomplished as a coordinated industry-wide effort. But the Libre Services model enables precisely the necessary large-scale, distributed, cooperative effort. Libre Services brings the tremendous collaborative power of free software to the Internet Services domain.

1.1 Overview Of ByStar Digital Ecosystem

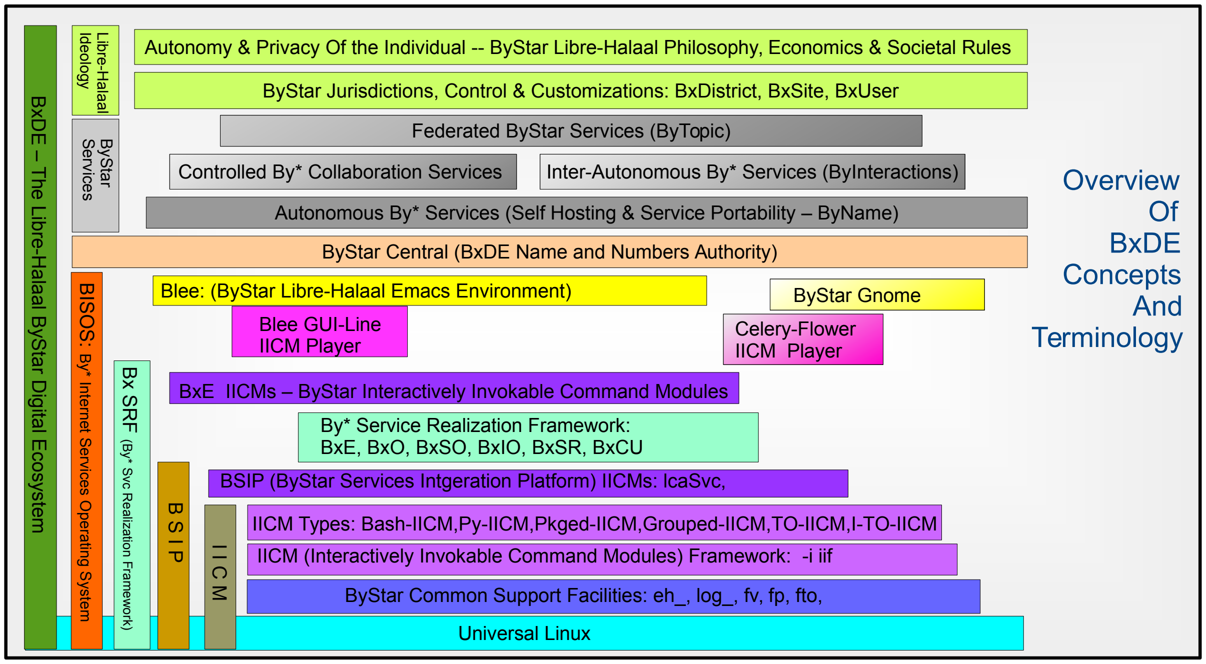
BxDE – The Libre-Halaal ByStar Digital Ecosystem
Libre-Halaal Ideology
Autonomy & Privacy Of the Individual -- ByStar Libre-Halaal Philosophy, Economics & Societal Rules
ByStar Jurisdictions, Control & Customizations: BxDistrict, BxSite, BxUser
ByStar Services
Federated ByStar Services (ByTopic)
Controlled By* Collaboration Services
Inter-Autonomous By* Services (ByInteractions)
Autonomous By* Services (Self Hosting & Service Portability – ByName)
ByStar Central (BxDE Name and Numbers Authority)
BISOS: By* Internet Services Operating System
Blee: (ByStar Libre-Halaal Emacs Environment)
Blee GUI-Line IICM Player
ByStar Gnome
Celery-Flower IICM Player
Bx SRF (By* Svc Realization Framework)
BxE IICMs – ByStar Interactively Invokable Command Modules
By* Service Realization Framework: BxE, BxO, BxSO, BxIO, BxSR, BxCU
B S I P
BSIP (ByStar Services Intgeration Platform) IICMs: IcaSvc,
I I C M
IICM Types: Bash-IICM,Py-IICM,Pkged-IICM,Grouped-IICM,TO-IICM,I-TO-IICM
IICM (Interactively Invokable Command Modules) Framework: -i iif
ByStar Common Support Facilities: eh_, log_, fv, fp, fto,
Universal Linux
Overview Of BxDE Concepts And Terminology

Figure 1: ByStar Digital Ecosystem Model and Terminology

Figure xx depicts main 4 elements of ByStar Digital Ecosystem:

- Libre-Halaal Ideology
- ByStar Services
- ByStar Central
- ByStar Internet Services OS (BISOS)

1.1.1 Libre-Halaal Ideology Summary

1.1.2 ByStar Services Summary

- ByStar Autonomous Private Cloud (Private Server)
- ByStar Collaborative Services
- ByStar Inter-Autonomous Direct Interactions – Email, chat, talk
- ByStar Inter-Autonomous Mediated Interactions – (ByInteractions, ByHookup)
- ByStar Federated Services – (ByContent)

1.1.3 ByStar Central Summary

1.1.4 ByStar Internet Services OS (BISOS) Summary

1.2 ByStar Engineering Philosophy

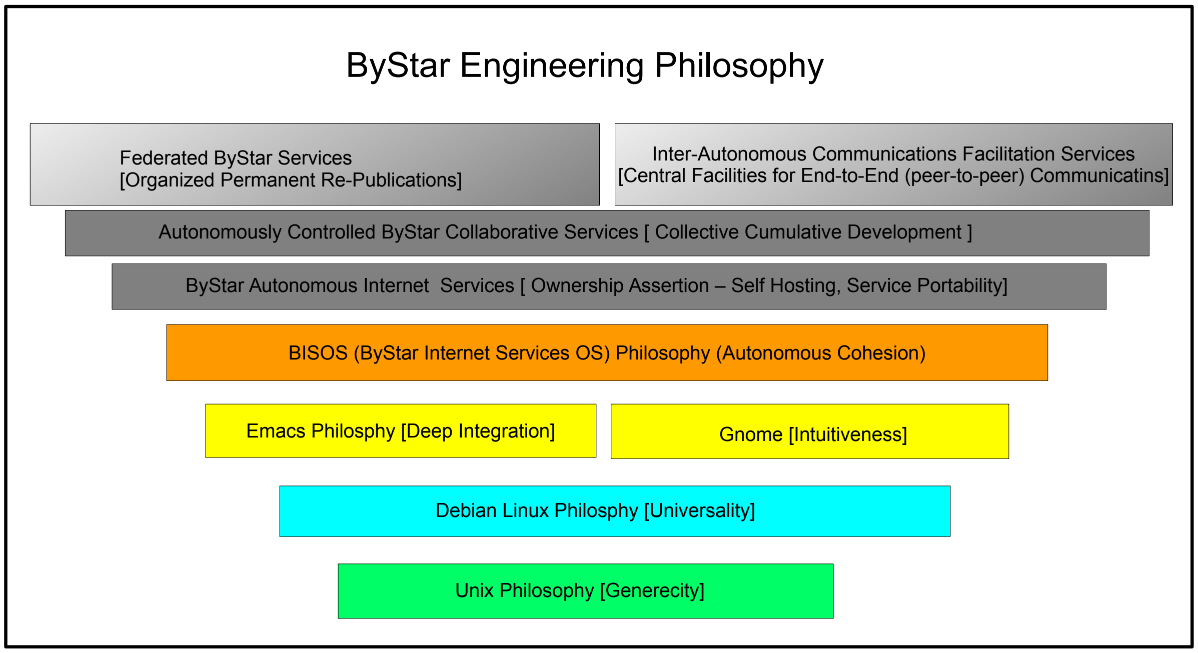
ByStar Engineering Philosophy
Federated ByStar Services [Organized Permanent Re-Publications]
Inter-Autonomous Communications Facilitation Services [Central Facilities for End-to-End (peer-to-peer) Communicatins]
Autonomously Controlled ByStar Collaborative Services [Collective Cumulative Development]
ByStar Autonomous Internet Services [Ownership Assertion – Self Hosting, Service Portability]
BISOS (ByStar Internet Services OS) Philosophy (Autonomous Cohesion)
Emacs Philosphy [Deep Integration]
Gnome [Intuitiveness]
Debian Linux Philosphy [Universality]
Unix Philosophy [Genereciy]

Figure 2: ByStar Engineering Philosophy

1.2.1 Value Propositions

The By* services provide a number of critical value propositions both to end users and to service providers.

- By virtue of being Libre Services, the protection of a number of critical freedoms and civil liberties, including privacy, freedom of speech, and freedom of information.

- Greatly enhanced user functionality based on the completeness and close integration among the By* family services.
- Greatly enhanced user functionality based on the deep integration between the By* services and the By* user environments.
- Constantly increasing richness of features and functionality via the free software development model.
- The By* services are designed in every way for scaling, and to be a long-term, enduring value proposition for all users. This does not refer just to the ability to accommodate large numbers of users, it refers to the naming architecture allowing unlimited spreading of the franchise model. This gives service providers a long-term rationale for buy-in.
- The By* structure provides the basis to address certain problems currently limiting and/or degrading the Internet, resulting from the unstructured manner in which it has arisen. A good example is public key encryption. Though of enormous value, this has not yet entered mainstream usage. The reason for this is that public key encryption requires large-scale uniformity and consensus of usage. In the existing Internet landscape consisting of multiple *ad hoc* non-interoperating islands of functionality, this is not possible. But By* provides precisely the large-scale uniformity of usage required to address problems like this.

1.3 ByStar Platform Universality Leading To Deep Software-Service Continuums

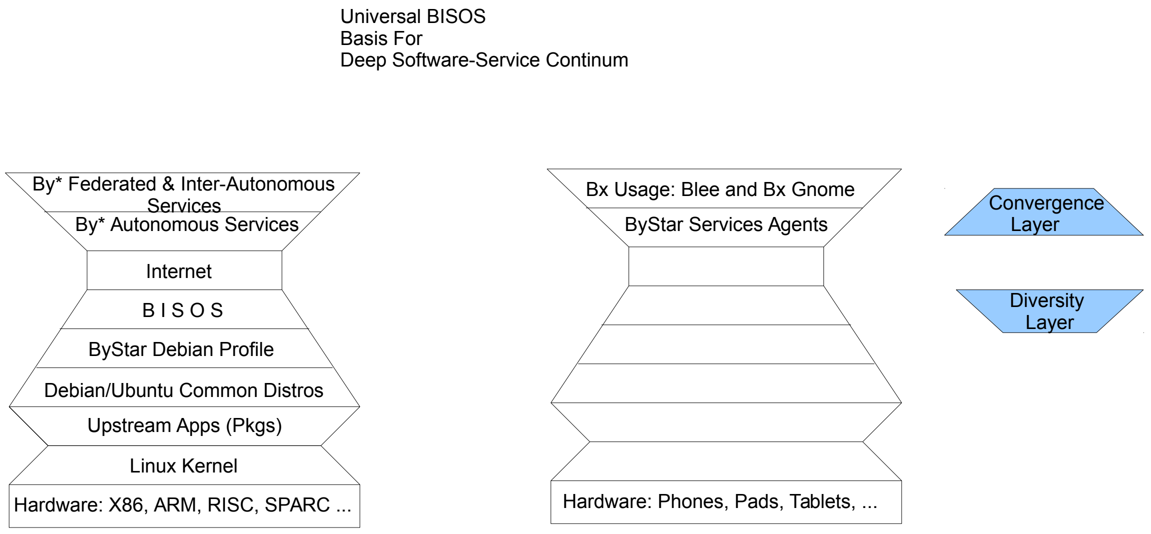
Universal BISOS
Basis For
Deep Software-Service Continum
By* Federated & Inter-Autonomous Services
By* Autonomous Services
Internet
B I S O S
ByStar Debian Profile
Debian/Ubuntu Common Distros
Upstream Apps (Pkgs)
Linux Kernel
Hardware: X86, ARM, RISC, SPARC ...
Bx Usage: Blee and Bx Gnome
ByStar Services Agents
Hardware: Phones, Pads, Tablets, ...
Convergence Layer
Diversity Layer

Figure 3: ByStar Platforms Universality Leading To Deep Software-Service Continuums

1.3.1 Model Of Universalities (and Diversities) In ByStar Digital Ecosystem

NOTYET, bring in the hour glass picture from libre services paper.

Capability	ByStar
Universal OS	Ubuntu/Debian/Linux/Gnu
Universal Desktop	Gnome Everywhere
Universal Browser	Firefox Everywhere
Unified Editor Centered User Env	Blee/Emacs
Universal Organization Tools	Org-Mode
Universal Synchronization And Version Control	Git and BxGit
Universal Personal Clouds	Autonomous Bx Services
Universal Content Access	Bx Federated Services
Universal Identity, Authentication	Autonomous Bx Services

Table 1: ByStar Universal Software And Services

Everything built around a singular Unix with full cohesion.

Towards full abstract-meta-homoiconicity.

1.3.2 Debian The Universal Operating System

1.3.3 Universal Software And Services

1.3.4 BxSe/BxISo The Universal Package Of Information and Service

1.4 ByStar Software And Service Catgorizations

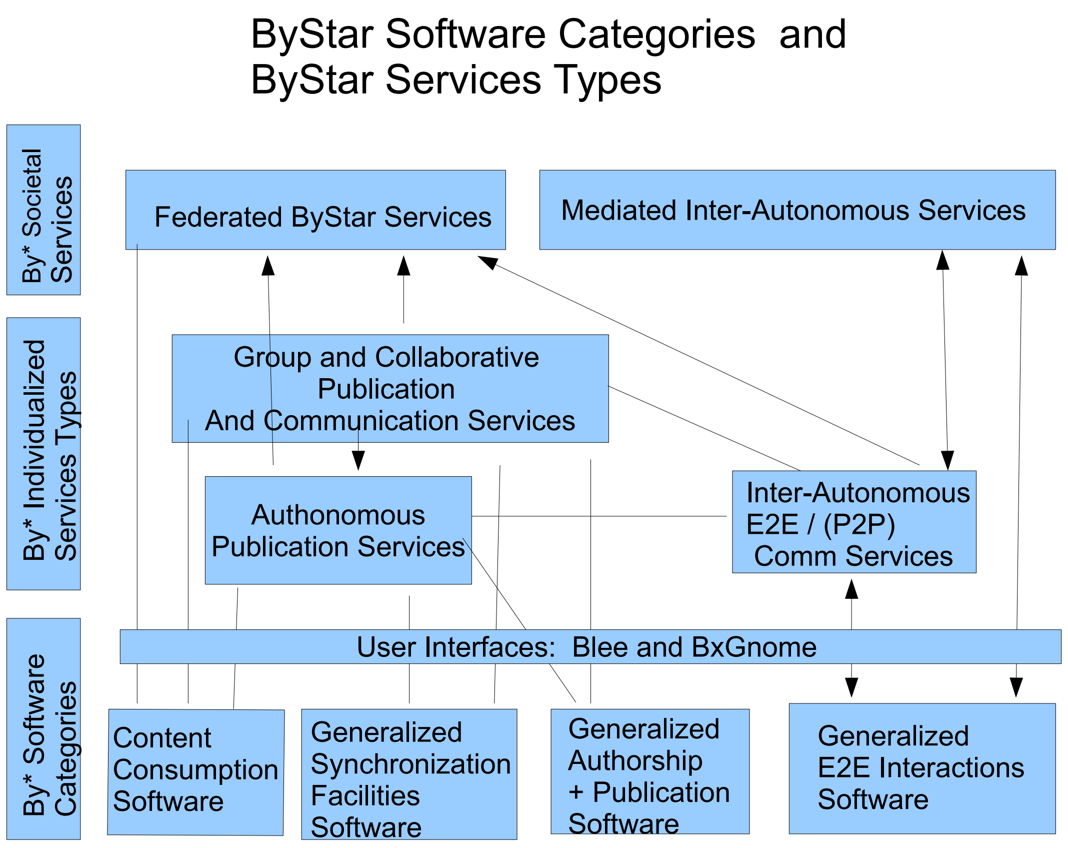
ByStar Software Categories and ByStar Services Types
By* Societal Services
By* Individualized Services Types
By* Software Categories
Federated ByStar Services
Mediated Inter-Autonomous Services
Group and Collaborative Publication And Communication Services
Authonomous Publication Services
Inter-Autonomous E2E / (P2P) Comm Services
User Interfaces: Blee and BxGnome
Content Consumption Software
Generalized Synchronization Facilities Software
Generalized Authorship + Publication Software
Generalized E2E Interactions Software

Figure 4: ByStar Software And Services Types

1.4.1 Functionality Oriented Software And Services

The ByStar digital ecosystem consists of four elements:

- ByStar User Environments
- ByStar Autonomous Private Cloud (Private Server)
- ByStar Inter-Autonomous Direct Interactions – Email, chat, talk

Functional Group	ByStar
Content Authoring	XeLaTeX, Havea, libreoffice-Draw, vlc
Content Consumption	Kodi, vlc
Communication And Collaboration Tools	Gnus
Personal And Team Productivity Tools	Blee-Org-Mode
Software Development	python, elisp, c, javascript, html
Things Management	EMPNA
Professionally Oriented Apps	R

Table 2: ByStar Functionality Oriented Software And Services

- ByStar Inter-Autonomous Mediated Interactions – (Federated Services)

These work in harmony through the unified model of ByStar-Objects (ByStar Entities).

ByStar provides online communication tools respecting your autonomy, privacy and data possession and ownership.

ByStar is a Libre-Halaal software stack, a subset of the Debian universal operating system.

2 Overview Of By* Internet Services OS (BISOS)

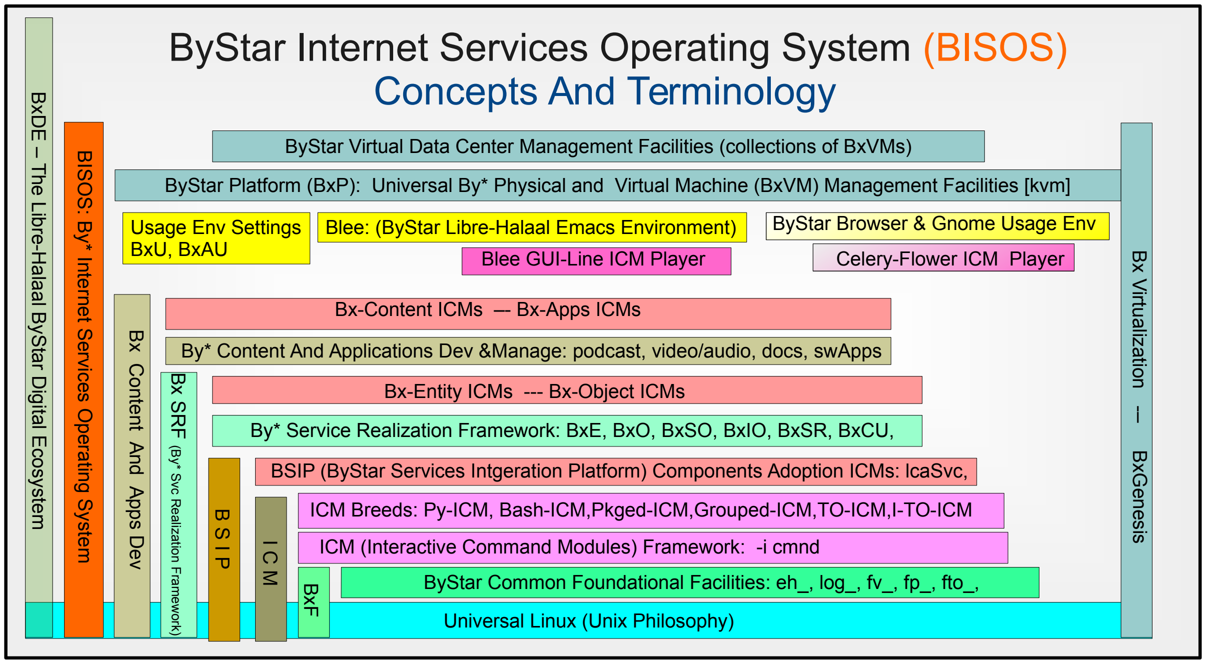
ByStar Internet Services Operating System (BISOS)
Concepts And Terminology
ByStar Virtual Data Center Management Facilities (collections of BxVMs)
ByStar Platform (BxP): Universal By* Physical and Virtual Machine (BxVM) Management Facilities [kvm]
Usage Env Settings BxU, BxAU
Blee: (ByStar Libre-Halaal Emacs Environment)
ByStar Browser & Gnome Usage Env
Blee GUI-Line ICM Player
Celery-Flower ICM Player
Bx-Content ICMs — Bx-Apps ICMs
By* Content And Applications Dev &Manage: podcast, video/audio, docs, swApps
Bx-Entity ICMs --- Bx-Object ICMs
By* Service Realization Framework: BxE, BxO, BxSO, BxIO, BxSR, BxCU,
BSIP (ByStar Services Intgeration Platform) Components Adoption ICMs: IcaSvc,
ICM Breeds: Py-ICM, Bash-ICM,Pkged-ICM,Grouped-ICM,TO-ICM,I-TO-ICM
ICM (Interactive Command Modules) Framework: -i cmnd
ByStar Common Foundational Facilities: eh_, log_, fv_, fp_, fto_,
Universal Linux (Unix Philosophy)
BxDE – The Libre-Halaal ByStar Digital Ecosystem
BISOS: By* Internet Services Operating System
Bx Content And Apps Dev
Bx SRF (By* Svc Realization Framework)
B S I P
I C M
BxF
Bx Virtualization — BxGenesis

Figure 5: BISOS Model and Terminology

3 Interactive Command Modules (ICM) and Players

Short summary plus reference to PLPC-180050.

4 Overview Of By* Software And Services

5 Model Of ByStarEntity And ByStarObjects

In terms of functionality and capability, By* is a unified services model. It is a coherent framework for enabling complex interactions among people, businesses and information. The By* framework is based on a formal engineering design approach. The architectural and design considerations are based on proper engineering discipline, rather than short-term marketing and business considerations. In creating By* we have considered the following sorts of questions:

- In creating such a framework, what are the key types of entity (individuals, businesses, etc.) that must be represented?
- For each type of entity, what is required to represent the entity in a highly generalized, abstract form?
- What structures and conventions are required so that these entities can be instantiated and named consistently, at a scale of 6 billion?
- What general classes of services are required to enable complex interactions among these entities?

None of these questions was asked during the explosive, organic growth of the Internet that brought us to where we are today. And this is what makes By* different. By* is a formal model for bringing structure and order to the Internet, at the scale of the entire planet.

5.1 The ByStarEntity Concept

By* is based on a set of key abstractions, representing the major real-world entities that must be represented within a generalized web structure. These entities include such things as individual persons, businesses, physical locations, and events. For each such entity we have defined the structures and conventions required to represent, instantiate and name that entity in a unified consistent way, and at a very large scale. We have then defined the major classes of services required to manage these entities, and to allow highly generalized interactions within and among each other.

In the ByStar applied model, a real-world entity type (for example individuals or a physical locations) maps on to a `ByStarEntityType`. A real-world entity instance maps on to a `ByStarEntity` All ByStar services are anchored in `ByStarEntity`.

ByStarEntityTypes are structured hierarchically in a tree.
`ByStarEntityType` is either a `ByStarAutonomousEntityType`
or a `ByStarControlledEntityType`.

`ByStarAutonomousEntityType` and `ByStarControlledEntityType` are either Classified or UnClassified.

Each `ByStarEntityType` is identified by a `ByStarEntityTypeId`.

In this structure, persons identified by their name, are represented as:

```
ByStarEntityTypeId=ByStarEntityType.ByStarAutonomousEntityType.Classified.Person.ByName
```

Each `ByStarEntity` (an instance) is identified by `ByStarEntityId`.

A `ByStarEntityId` is structured as:

```
  ByStarEntityId=RegistrarId+ByStarEntityTypeId+InstanceId
```

All ByStarEntityIds are unique. The `InstanceId` is assigned by the `RegistrarId`.

Each `ByStarEntity` can be activated within a `ByStarAutonomyAssertionVirtualMachine`.
The representation of a `ByStarEntity` in a `ByStarAutonomyAssertionVirtualMachine`
is called a `ByStarServiceObject`.
A `ByStarServiceObject` maps to a Unix account and a user-id.
The `ByStarServiceObject` can have any `ByStarServiceCapability`
that `ByStarAutonomyAssertionVirtualMachine` offers.

Any `ByStarServiceCapability` can be bound to and exposed through a registered domain name.

Currently, ByStarServiceCapability is one of the capabilities enumerated in figure **??**.

Based on the above structures, ByStar services can consistently grow and interact with other ByStar services to provide a rich and healthy environment.

5.2 Naming principles

A consistent naming convention is essential in order to instantiate entities such as individual persons at extremely large scales. All object instantiations throughout By* are based on consistent naming principles. For example the ByName service provides each user with a personal domain based on the user's own name, using the naming schema:

homer.simpson.1.ByName.net

This naming schema allows an unlimited number of named instances. ByName users are required to provide their real names for this purpose; pseudonyms and aliases are not permitted. This implies a certain standard of authenticity and integrity on the part of both the ByName service and the ByName user.

5.3 User Environments

Internet services work by communication over the Internet between a client application running in the user's computing environment, and a corresponding server application running within the service.

In the proprietary model the service is typically accessed via a web browser. It may also possibly be accessed via a dedicated client application provided by the service provider.

In the Libre Services model, however, there are no proprietary limitations placed on integration between the user's computing environment and the service. Since the service is completely transparent, any user environment can be integrated with any Libre Service.

Furthermore, a much deeper level of integration is now possible. In particular, free user environments (i.e. user environments based on free software) can be integrated with Libre Services. And since both the client and server sides of the service are now completely transparent, this permits a highly complex level of integration between the two. This allows the development of Internet services with a power and versatility that far exceeds what exists today.

The By* services can thus be greatly enhanced by providing the user with a "matched" environment—a user environment that is closely integrated with the service. This will provide the user with features and capabilities that go far beyond what is possible using the traditional generic browser access. The role of matched user environments is described in greater detail in *Libre Services: A non-proprietary model for delivery of Internet Services* [**?**].

At the appropriate point in our continued development of By* we will develop these matched user environments to enhance the utility of the By* services. For example see the Project Document titled “Libre Emacs Office Environment (EOE)” in the article *Libre Services: Projects for bootstrapping* [?]. The goal of this project is to establish Emacs as a complete user environment for interaction with the initial set of starting-point Libre Service Engines. These are the Libre Engines upon which the By* services are based, therefore the resulting user environment will be immediately applicable to By*.

6 Model Of Information and Service Realization In ByStar Digital Ecosystem

6.1 BxDE – ByStar Digital Ecosystem

6.2 BxCollective Concept

Belief system, Country, Society, Laws.

6.3 BxDistrict Concept

A Public of Private Service Provider Cloud. A Service Provider with its own policies and AUP.

6.4 BxSite and BxCluster Concepts

A collection of colaborative BxPlatforms. BxCluster is technology oriented. BxSite is policy and value oriented.

6.5 Definition of BxPlatform – Autonomy Assertion Platforms

6.6 Definition of BxInfoAndServiceObject (BxISo) and BxInfoAndServiceEntity (BxISe)

6.7 Definition of BxServiceCapability (BxSc)

6.8 Definition of BxServiceRealization (BxSr)

7 ByStar Platform Taxonomy and Genesis Process

«Xref-BxPlatform»

7.1 Physical Platform (PP)

Physical Platform (PP) is the bare computer that is being used. Without any of the optional peripherals that can be connected to it.

Each PP is uniquely identified by a PP-ID (obtained from /sys/class/dmi/id/product_uuid).

In the contex of ByStar, A PP-ID can be mapped to a PP-Name (Formerly BoxName).

Physical Platform (PP) facilities are abstracted in:

```
/opt/public/osmt/bin/ppMachineInfo.sh
```

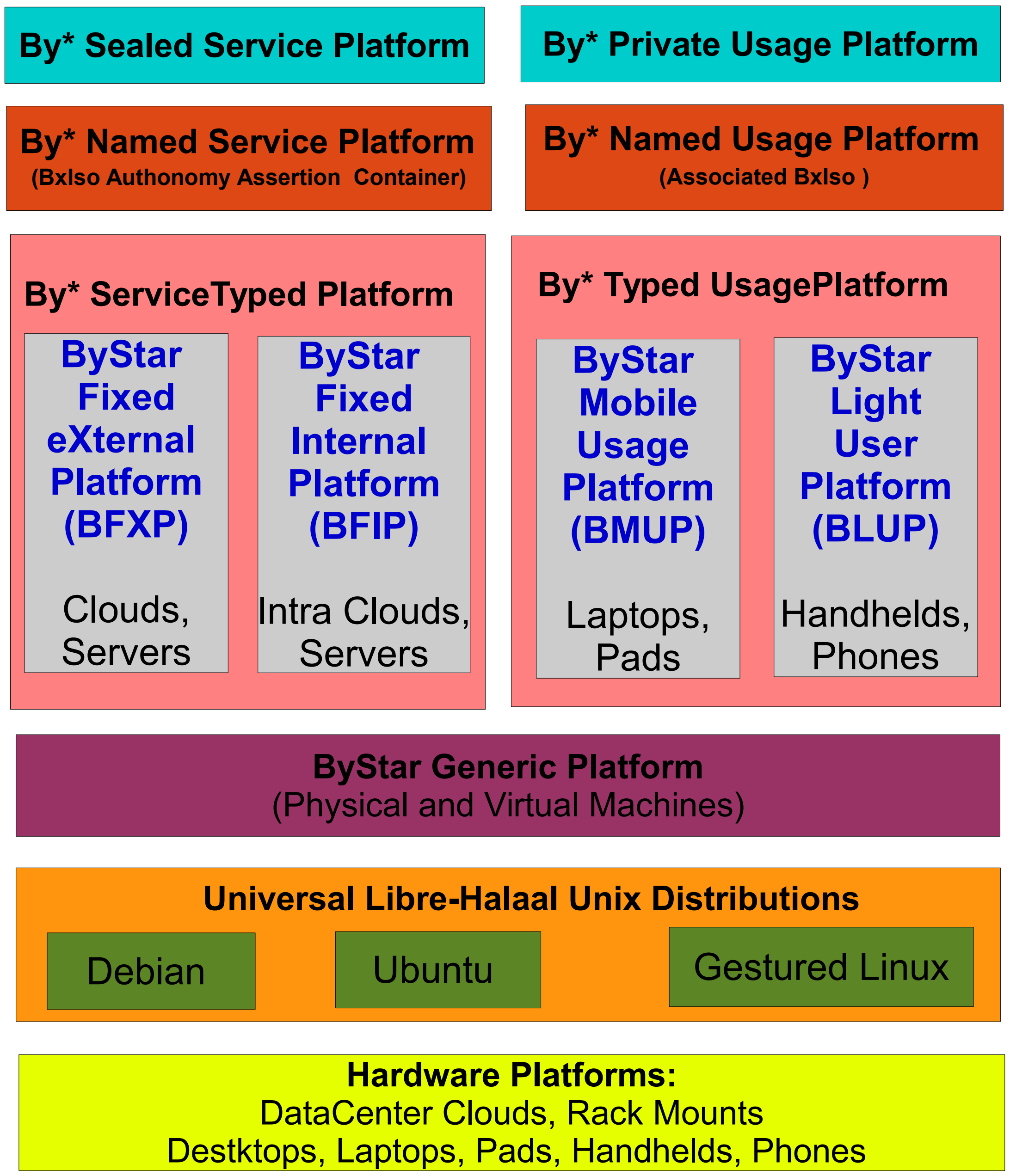
By* Sealed Service Platform
By* Private Usage Platform
By* Named Service Platform
(Bxlso Authonomy Assertion Container)
By* Named Usage Platform
(Associated Bxlso)
By* ServiceTyped Platform
ByStar Fixed eXternal Platform (BFXP)
Clouds, Servers
ByStar Fixed Internal Platform (BFIP)
Intra Clouds, Servers
By* Typed UsagePlatform
ByStar Mobile Usage Platform (BMUP)
Laptops, Pads
ByStar Light User Platform (BLUP)
Handhelds, Phones
ByStar Generic Platform
(Physical and Virtual Machines)
Universal Libre-Halaal Unix Distributions
Debian
Ubuntu
Gestured Linux
Hardware Platforms:
DataCenter Clouds, Rack Mounts
Destktops, Laptops, Pads, Handhelds, Phones

Figure 6: Platform Genesis Steps And Layers

7.2 Effetive Physical Platform (EPP)

Effective Physical Environment (EPP) – The Physical Environment in Effect. If a laptop is now connected to a second large monitor. EPP now include that second large monitor as well.

7.3 ByStar Platform (BxP)

BxP is the base distro augmented with ByStar abstractions and software that can be an element within BXDE.

BxPs are universal. They are common platforms that can contain BxIo-s and BxISo-s.

Each BxP will include the following tags:

1. BxP Generation
   CVS Pre 2016 generation is BxP-Gen-1
   CVS 2016 generation is BxP-Gen-2
   Git 2016 generation is BxP-Gen-3

   Each BxP-Gen uses different distros at different distro releases. For example BxP-Gen-3 can be built on top of Ubuntu 1404 or Ubuntu 1604.

2. BxP Software Profiles

   - BxP-Base-SwP – Minimal BxIo Container (Information Object Container) which is common across all other SwP-s. Includes Blee.
   - BxP-User-SwP – BxIo + User Env
   - BxP-ExternalService-SwP – BxSIo + User Env + External-Services
   - BxP-UserService-SwP (User and Internal-Service) – BxSIo + User Env + Internal-Services
   - BxP-Developer-SwP
   - BxP-VirtHost-SwP (A Host For KVM, Can be Combined with BxP Developer)

3. BxP Connectivity Profiles

   - ByStar Mobile Interior Connected (BMIC) – Mob-
   - ByStar Fixed eXterior Connected (BFXC) – Ext-
   - ByStar Fixed Interior Connected (BFIC) – Int-

4. BxP Service Profiles – BxP-Generic-Characters (To Be Assigned Based On BxP Connectivity Profiles) The BxP-Generic-Character is a BxISo type. It does not include any assignments and it does not include any service data.

   - BLUP: Bx Light User Platform (BxP-User-SwP + BMIC)
     For Handhelds, Phones
   - BMUP: Bx Mobile Usage Platform (BxP-UserService-SwP + BMIC)
     For Laptops, Pads
   - BDIP: Bx Developer Internal Platform (BxP-Developer-SwP + BFIC or BMIC)
     Desktops and Laptops
   - BUSP: Bx User Service Platform (BxP-UserService-SwP + BFIC)
     Intra Cloud Servers

    - BXSP: Bx External Service Platform (BxP-ExternalService-SwP + BFXC)
      Internet Cloud Servers

5. BxP-Specific-Characters The BxP-Specific-Character is a BxISo type. It includes all assignments and some or all service data. Examples:
    - Reproducible and Portable Specific DNS Server with all its data
    - Reproducible and Portable Server for a set of customers

Various of these are available as common VMs named based on the above taxonomy.

7.4 ByStar Platform Connectivity Types

7.4.1 ByStar Fixed eXterior Connected (BFXC) – Ext-

Visible and reachable on Public Internet.

7.4.2 ByStar Fixed Interior Connected (BFIC) – Int-

Fixed-IP Address behind a particular firewall inside a particular Intranet.

7.4.3 ByStar Mobile Interior Connected (BMIC) – Mob-

HDCPed or fixed IP Address within possibly a number of Intranets. Always firewalled.

7.5 ByStar Software Profile (Capability Types)

7.5.1 BxP-Service-SwP

Service Software Profile. BxIso Containers, where the Iso can be realized – become a service.

Previously BACS.

7.5.2 BxP-UserService-SwP

BxIso Containers, where the Iso can be realized – become a service.

Different from BXFP in terms of services. For example BFIP can run NFS/Samba but not BFXP.

Previously BISP.

7.5.3 Bxp-User-SwP

For laptops and pads.

Previously BUE.

7.5.4 Bxp-Developer-SwP

For handhelds and phones.

7.5.5 BDPP – ByStar Domain (DNS) Publisher Platform

7.5.6 BDRP – ByStar Domain (DNS) Resolver Platform

7.5.7 ByStar Hosting Virtualization Platform

7.5.8 BCP – ByStar Custom Platform

7.6 ByStar Character – BxP-Character – (Character Types)

Combination of:

- Software Profile
- Connectivity Type
- Character Assignments (Name, IP-Addrs, Service Capabilities)

7.6.1 BxP-Generic-UnAssigned Character

BLUP: Bx Light User Platform (BxP-User-SwP + BMIC) Mobile Connectivity + Developer Profile For Handhelds, Phones

BMUP: Bx Mobile Usage Platform (BxP-UserService-SwP + BMIC) For Laptops, Pads

BDIP: Bx Developer Internal Platform (BxP-Developer-SwP + BFIC or BMIC) Desktops and Laptops

BUSP: Bx User Service Platform (BxP-UserService-SwP + BFIC) Intra Cloud Servers

BXSP: Bx External Service Platform (BxP-Service-SwP + BFXC) Internet Cloud Servers

7.6.2 BxP-Specific Character – Examples

The BxP-Specific-Character is a BxISo type. It includes all assignments and some or all service data. Examples:

- Reproducible and Portable Specific DNS Server with all its data
- Reproducible and Portable Server for a set of customers

BXSP-0024: Bx External Service Platform

7.7 BxP Software Platform Gensis Process

IP Addrs and Name remain generic as we move through phases.

7.8 BxP Platform Gensis Process

```
**    Stages       ::  Order Of Steps
***    Distro           ::  Distro (Ubuntu/Debian) Installation From Media
***   BxP-Generic    :: Common Packages -- Initial SW -- Everything Retrieved As dist/anon (Common-VM)
***   BxP-SwTyped     ::  BUE0, BISP0, BACS0 -- Software Profile -- But Un-Named (Common-VM)
***    BxP-Named        ::  Assigned -- Named -- Can Be Live
     Retreive bxAdmin BxIso
     dist/anon is replaced by dist/auth
     Site BxIso will be retreived -- New Name Can Be Assigned
     Named BxP-Iso May Be Created to reproduce this BxP-Iso
***    BxP-Provisioned  ::  Prepared/Provisioned To Provide Particular Sets Of Services
***    BxP-Sealed       ::  Customized and Specialized
     Retrieve Additional BxIso-s
```

7.9 BxP Character Realization Process

BxP Character Realization Process Takes as its input the following:

- BxP-Character
  - BxP-Generic-UnAssigned-Character
  - BxP-Specific-Character
- BxP-Software Profile (or a common VM image)

Relevant district, site BxISo-s are retrieved.

When needed, assignments are made to produce a BxP-Generic-Assigned-Character based on BxP-Generic-UnAssigned-Character.

The following setps are then realized.

- /etc/network/interfaces is created.
- The platform is re-named.
- All applicable services are re-configured based on new identity.
- All inapplicable services are disabled.
- Platform is rebooted with new identity

Once the BxP has its chracter, it can start receiving BxISo-s.

7.10 ByStar Platform On Top Of Debian/Ubuntu

7.11 Obtaining ByStar Platform Virtual Machine Images

8 BxE/BxISe Sovereignty, BxISe Registration and BxSe Types

8.1 BxSSe Sovereignty, BxSe Registration

1. ByStar Autonomous Entity Types

2. ByStar Controlled Entity Types
3. ByStar Federation Entity Types
4. ByStar Collaborative Entity Types
5. ByStar Central

8.2 Autonomous BxISe Types

1. Bxe-Identified-Individual (ByName)
2. Bxe-Organization (BySmb)

8.3 Controlled BxISe Types

8.3.1 BxDe Platform, Site, District – BxEs

1. BxDe-PlatformCharacter (pse) Read/Write. Example, BACS0019: Contains necessary information to turn any box into a specified profile. Work with charXXX as part of LSIP.
2. BxDe-Site (sse) Read/Write. Example, NedaPlus: BxSite defines IP Address Assignments, File System Mounts, ... Work with siteXXX as part of LSIP.
3. BxDe-District (dse) Read/Write. Example, LibreCenter: BxSite defines IP Address Assignments, File System Mounts, ... Work with siteXXX as part of LSIP.

8.3.2 Website BxEs

1. Bxe-WebSite
2. Bxe-WebSite-ByMemory
3. Bxe-WebSite-ByAuthor
4. Bxe-WebSite-ByArtist
5. Bxe-WebSite-ByWhere
6. Bxe-WebSite-ByFamily
7. Bxe-WebSite-ByAlias
8. Bxe-Identified-Individual
9. Bxe-Organization
10. Bxe-Project

8.3.3 Controlled Individual and Controlled Organization BxEs

1. Bxe-Identified-Individual
2. Bxe-Organization

8.3.4 Projects (Private and Collaborative) BxEs

1. Bxe-Project-Standalone
2. Bxe-Project-Collaborative

8.4 Federation BxISe Types

1. Content Republication
   (a) ByContent
   (b) ByTopic
   (c) BySearch
   (d) BySource
   (e) ByBinary
   (f) ByEvent
2. Anonymity
   (a) ByLeaks
3. End-To-End Interaction Facilitation
   (a) ByInteraction
   (b) ByHookUp

8.5 Collaborative BxSe Types

1. ByLookUp

8.6 ByStar Central

1. Philosophy, Morality, Ethics
   (a) Free Protocols
   (b) Neda
   (c) By-Star.net
   (d) HalaalSoftware
   (e) LibreServices
2. Names And Number Authority – by-star.net
3. BxDistricts – LibreCenter

9 BxISe (BxSIAE) in Potential Form and BxISe in Realized Form

A ByStar (Bx) Services Information Abstract Entity (SIAE) (BxSIAE)

A BxSe is either in Potential Form (Potential-BxSe) or in Realized Form (Realized-BxSe).

A Potential-BxSe represents a registration and no services or information.

A Realized-BxSe represents a registration and services and/or information.

In realized form, a BxSe is hosted on a BxPlatform as a BxISo. Each Realized-BxSe is in the form of a Principle-BxISo through which other BxISo-s on other BxPlatforms can be cloned. Those Cloned-BxISo-s, and the Principle-BxISo together are realization of a BxSe.

10 Principle-BxISo (BxSIo Services Info Obj) Creation Based On BxISe (BxSIAE)

```
BARC  -- ByStar Acct Request Container
||
VV
RBAE -- Registered Bx Acct Entry (Same As BxSe)
||
VV
BxPlatform
||
VV
Autonomous-Principle-BxISo  (Temp-Passwd)
||
VV
Cloned-BxISo
||
VV
Controlled-BxISo[0-N]
```

Autonomous-BxSe –> Autonomous-Principle-BxISo – Temp Passwd Communicated To Autonomous Entity.

Autonomous-Realized-BxSe –> Controlled-BxSe[1-N]

11 Structure Of BxISo Base Dir

/opt/public/osmt/bin/bxsoBaseDirs.sh is based on the seed described in – ByStar Base Directories (/hss, /dd, /de, /uniform) –.

11.1 BxISo Base Top Dirs

```
~bxiso/iseOid               # Mirror Of BxIse-Oid
~bxiso/iso                  # Information And Service Object (Synced)
~bxiso/sync                 # Checked out Git areas of iso's general sync area
~bxiso/control              # Non-Sync-ed
~bxiso/var                  # Non-Sync-ed
~bxiso/tmp
```

11.2 Structure Of BxISo/iso

11.2.1 bxiso/iso/entity – Registeration and Access Info For This BxISo

Perhaps biso/iso/bxe instead of entity.

```
~bxiso/iso/entity/bxId             # BxSe-Id
~bxiso/iso/entity/passwd           # Temporary Password
```

11.2.2 bxiso/iso/gits – Gits Admin/Control Information

These base dirs are tied to BxU's User Environment assumptions.

Previously LUE.

```
~bxiso/iso/gits/admin
~bxiso/iso/gits/admin/repos/pub/sync/0
```

11.2.3 bxiso/iso/cap – Capabilities List

Based on specification of biso/iso/cap, biso/iso/sr can be auto generated.

```
~biso/iso/cap                           #
~biso/iso/cap/mail/full                 #
~biso/iso/cap/plone3/basic              #
```

11.2.4 bxiso/iso/sr – Service Realization

The structure is: "/sr" Service Realization. Followed by: "/sr/capability".

```
~BxISo/iso/sr/apache2/git                 # Web Authenticated Git Access
~BxISo/iso/sr/apache2/gitweb              # Unauthenticated Git Access
~BxISo/iso/sr/apache2/web                 # Ordinary plain html web server

~BxISo/iso/sr/plone3/bxMain               # BxISo's primary Plone3 Site
```

11.2.5 bxiso/iso/svcPars – Common Service Parameters

```
~BxISo/iso/svcPars/git/name   # This should really go under gits admin
~BxISo/iso/svcPars/git/ipAddr   # This should really go under gits admin)
~BxISo/iso/svcPars/domains
~BxISo/iso/svcPars/domains/1
```

11.2.6 bxiso/iso/rel – Relations

```
~bxiso/iso/rel/master            # Autonomous bxso of the master for this bxso
~bxiso/iso/rel/controlled        # bxsos that this controls
~bxiso/iso/rel/members           # bxsos that are granted access to this bxso
~bxiso/iso/rel/groups            # bxsos providing access to this bxso
~bxiso/iso/rel/bundles           # Other BxISo-s that go together with this BxISo
```

11.2.7 bxiso/iso/pkcs – Public/Private Keys

```
~bxiso/iso/pkcs                 #
```

11.2.8 bxiso/iso/ue – User Environment

These base dirs are tied to BxU's User Environment assumptions.

Previously LUE.

```
~bxiso/iso/ue/emacs              #
~bxiso/iso/ue/elisp              #
~bxiso/iso/ue/bin                #
```

12 BxISo Replications And Synchronizations – Cloning From Principle

See /libre/ByStar/InitialTemplates/activeDocs/bxServices/versionControl/git/fullUsagePanel-en.org Section "ByStar Git: – Structures And Policies"

13 Model Of Service Usage and Administration In ByStar Digital Ecosystem

13.1 ByStar User Environemnts

13.1.1 Blee

13.1.2 BxGnome

14 BxU and extasciitilde bxu Structure

15 BxU – Ue-BxISo Association (BxU<–>BxAUE)

A Ue-BxISo is a class of BxISo-s that includes support for User Envirnoments.

At any given time, zero or one Ue-BxISo is bound (associated) with BxU. That Ue-BxISo is called: BxAUE

The totality of user environment that BxU, ByStar Platform, ByStar Services and BxAUE provide is called BUE (the ByStar User Environment).

16 ByStar Service Capabilities (BxSc) of BxISo

A “Service Capability” (BxSc) is a capability provided by the BxPlatform. For example apache2, plone3, qmail, geneweb, etc.

Within a BxISo, a BxSc is housed in:

```
~BxISo/sr/BxSc/specificRealization
```

As examples, consider:

```
~BxISo/sr/apache2/git          # Web Authenticated Git Access
~BxISo/sr/apache2/gitweb       # Unauthenticated Git Access
~BxISo/sr/apache2/web          # Ordinary plain html web server

~BxISo/sr/plone3/bxMain        # BxISo's primary Plone3 Site
```

17 ByStar Service Realization (BxSr and BxSrm) – Modes and DomainBindings

Based on a BxPlatform’s Service Capabilities (BxSc) specific services can be realized. Each specific realization of service capabilities is called a “Service Realization” BxSr.

Within a BxISo, a BxSr is housed in:

```
~BxISo/sr/BxSc/BxSr
```

As an example, consider:

```
~BxISo/sr/apache2/git          # Web Authenticated Git Access
```

Each BxSr has information and facilities to one or more domains (BxSr-Dom). Further a BxSr-Dom can be made to resolve to a local BxSr based on “Service Realization Mode” – BxSrm.

17.1 BxSr Domains Bindings

17.2 ByStar Service Realization Modes (BxSrm)

17.2.1 BxSrm=live

17.2.2 BxSrm=shadow

17.2.3 BxSrm=here

17.2.4 BxSrm=inactive

18 ByStar Base Directories (/hss, /dd, /uniform, privScope, /de)

/opt/public/osmt/bin/seedPlatformBaseDirs.sh

/opt/public/osmt/bin/bystarPlatformBaseDirs.sh

18.1 /hss Base

18.2 /dd Bases – Disk Drives

18.3 Privacy Scopes (privScopes)

18.4 /uniform Bases

18.5 /de Bases – Digital Ecosystems

18.6 BxISo Base Dirs

18.7 BxU Base Dirs

19 LSIP Facilities Naming And Terminology

20 ByStar Content

References