The Universal BISOS: ByStar Internet Services Operating System BISOS

Model, Terminology, Implementation And Usage

A Framework For Cohesive Creation, Deployment and Management Of Internet Services

Document #PLPC-180047
Version 0.3
December 14, 2018

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

Chapter 1

An Overview Of BISOS And ByStar

1.1 Introduction

The Internet Services industry of today has three characteristics that greatly limit its capabilities and its usefulness.

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 internet Applications Services Provider (ASP) sits in the center and controls and owns almost every aspect of our (user) communications.

Second, 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.

Third, 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.

In the abstract the solution to this comprises three fundamental parts:

1. We need to require the Libre-Halaal manner-of-existence for internet services. In other words the entirety of our Internet Services should be in internally transparent. The entire software of our own internet services should be Libre-Halaal Software (FOSS, FLOSS, Open-Source, Free Software).

2. We need a "Universal Internet Services OS" to bring consistency and cohesion.

3. We need a "Unified Autonomy And Privacy Oriented Digital Ecosystem" that builds on the "Universal Internet Services OS" and provides us autonomous services – that belong to us and are controlled by us.

Here by "our" and "us" we are speaking of the society at large when it is represented and protected by the Internet Engineering Profession.

Thus far we have been describing the contours of the problem and the contours of solution in abstract terms. We now make them concrete based on a specific implementation.

The concrete solution that we offer 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-Halaal Services and By* models have enormous implications. The Libre-Halaal 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.

1.2 Concept Of The Universal Internet Services OS

Shortly after Internet started to impact the society (say in 1994 – there is no knee for exponential curve) and shortly after Linux became widespread, the idea of a server-side Internet Services OS appeared as "The LAMP Stack"

1.2.1 History Of The LAMP Stack

LAMP is an acronym that stands for "Linux, Apache, MySQL, Perl/PHP/Python." Packaged together, they create an application stack that is both free to use and open source which functions as a general purpose web server.

The concept of a LAMP stack (the free general purpose web server) had been possible from as early as 1994 when CERN httpd introduced the Common Gateway Interface, which allowed for the server-side execution of code to create dynamic webpages. Linux, the CERN httpd, and server-side programming languages such as Perl were available for free, but it wasn't until later that same year and the release of Postgre95 that it was possible to obtain a free database as well. Finally in 1996, MySQL was released online and a full LAMP stack was possible.

Validity of the LAMP stack as a server-side web services generic OS was established through its widespread use in the late 1990s. Many of the dot-con era firms ran their websites with LAMP.

1.2.2 Key Attributes Of A Universal Internet Services OS

Extending and improvingthe concept of LAMP leads the notion of "A Universal Internet Services OS".

Such an extentions involves two dimmensions:

1. An Internet Services OS should cover all internet services – not just web services
2. An Internet Services OS should fully cover all sides – clients, servers and things in the middle.

By "Universal" we are refering to this notion of "covering all sides" from phones and pads to mainframes and sever-clusters.

This idea of "Universal Services OS" builds on Debian's concept of "The Universal Software Operating System".

1.3 Concept Of A Unified Autonomous Digital Ecosystem

Equipped with a "Universal Internet Services OS" we would then want to build an entire digital ecosystem that purely runs our Universal Internet Services OS.

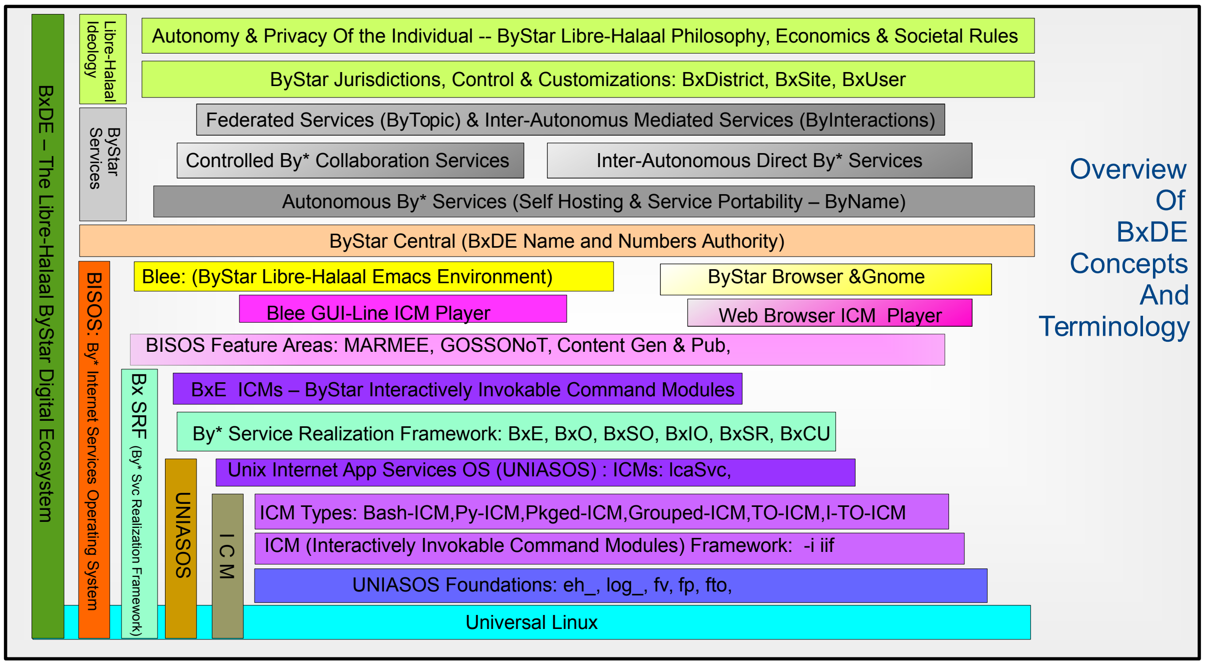

BxDE – The Libre-Halaal ByStar Digital Ecosystem
Libre-Halaal Ideology
Autonomy & Privacy Of the Individual -- ByStar Libre-Halaal Philosophy, Economics & Societal Rules
ByStar Jurisdictions, Control & Customizations: BxDistrict, BxSite, BxUser
ByStar Services
Federated Services (ByTopic) & Inter-Autonomus Mediated Services (ByInteractions)
Controlled By* Collaboration Services
Inter-Autonomous Direct By* Services
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)
ByStar Browser &Gnome
Blee GUI-Line ICM Player
Web Browser ICM Player
BISOS Feature Areas: MARMEE, GOSSONoT, Content Gen & Pub,
Bx SRF (By* Svc Realization Framework)
BxE ICMs – ByStar Interactively Invokable Command Modules
By* Service Realization Framework: BxE, BxO, BxSO, BxIO, BxSR, BxCU
UNIASOS
Unix Internet App Services OS (UNIASOS) : ICMs: IcaSvc,
I C M
ICM Types: Bash-ICM,Py-ICM,Pkged-ICM,Grouped-ICM,TO-ICM,I-TO-ICM
ICM (Interactively Invokable Command Modules) Framework: -i iif
UNIASOS Foundations: eh_, log_, fv, fp, fto,
Universal Linux
Overview Of BxDE Concepts And Terminology

Figure 1.1: ByStar Digital Ecosystem Model and Terminology

So all client side devices (phones, pad, laptops) and all services (mail, web) are built on top of the same software. That commonality and availability would then allow us to make the entire thing unified and autonomy oriented.

1.4 Overview Of BISOS and ByStar Digital Ecosystem

BISOS is the concrete form of the abstract “The Universal Internet Services OS”

ByStar is the concrete form of the abstract “Unified Autonomous Digital Ecosystem”.

1.4.1 Overview Of ByStar Digital Ecosystem

Figure xx depicts main 4 elements of ByStar Digital Ecosystem:

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

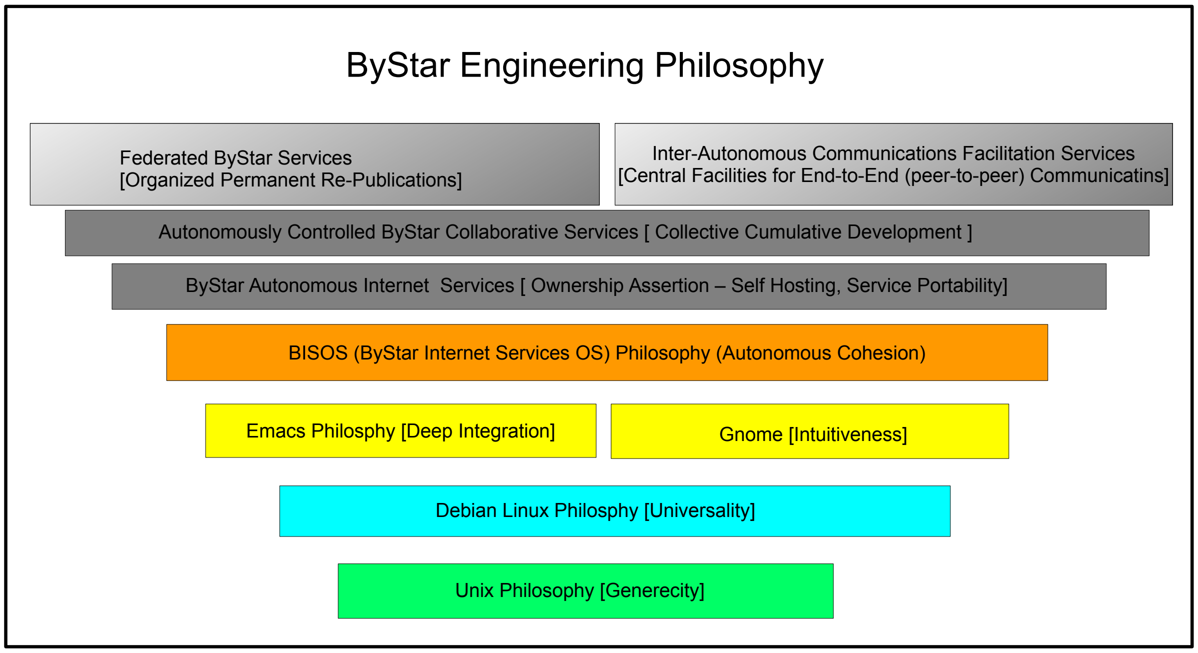

ByStar Engineering Philosophy
Federated ByStar Services [Organized Permanent Re-Publications]
Inter-Autonomous Communications Facilitation Services [Central Facilities for End-to-End (peer-to-peer) Communicatins]
Autonomously Controlled ByStar Collaborative Services [Collective Cumulative Development]
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 [Generecity]

Figure 1.2: ByStar Engineering Philosophy

Libre-Halaal Ideology Summary

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)

ByStar Central Summary

ByStar Internet Services OS (BISOS) Summary

1.4.2 ByStar Engineering Philosophy And Propositions

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.

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 1.1: ByStar Universal Software And Services

- 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.

Model Of Universalities (and Diversities) In ByStar Digital Ecosystem

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

Everything built around a singular Unix with full cohesion.

Towards full abstract-meta-homoiconicity.

Debian The Universal Operating System

Universal Software And Services

BxSe/BxISo The Universal Package Of Information and Service

Functionality Oriented Software And Services

The ByStar digital ecosystem consists of four elements:

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 1.2: ByStar Functionality Oriented Software And Services

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

1.5 Overview Of By* Internet Services OS (BISOS)

1.6 BISOS's "On" Unix Model

A framework for cohesive creation, deployment and management of Internet Services.

Unix Philosophy vs Linux Philosphy vs Business Philosphy.

Bringing the Cathedral to the Bazar, in the open.

With BISOS, there is no need for containers, ansible, chef, pupet.

1.7 Interactive Command Modules (ICM) and Players

Short summary plus reference to PLPC-180050.

Chapter 2

BISOS Model And Terminology

BxSe/BxISo The Universal Package Of Information and Service

2.1 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.

2.1.1 BISOS's On Unix Universal Model

BISOS is based on the "On" Unix model. Not an "In" Unix or "With" Unix model.

BISOS is based on the "Unix" model. Not the "Linux" model. We draw a distinct differentiation between "Unix Philosophy" vs "Linux Philosophy" vs "Business Philosophy". Unix Philosophy is a set of cultural norms and philosophical approaches to convivial software development and usage. Unix Philosophy has been well articulated by Ken Thompson, Doug McIlroy, Kernighan, Pike and others.

Linux Philosophy is a laissez faire adoptation of Unix Philosophy that results into software bloat.

Business Philosophy is the American model of economic creatures producing proprietary software or bastardizing Linux.

BISOS is firmly rooted in a Unix Philosophy which rejects the Business Philosophy and the Linux philosophy.

BISOS is based on the "Universal Unix" model. Not the differnt one Linux in the cloud and another Linux in the embedded device.

2.1.2 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.

2.1.3 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.

2.1.4 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*.

2.2 Model Of Information and Service Realization In ByStar Digital Ecosystem

2.2.1 BxDE – ByStar Digital Ecosystem

2.2.2 BxCollective Concept

Belief system, Country, Society, Laws.

2.2.3 BxDistrict Concept

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

2.2.4 BxSite and BxCluster Concepts

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

2.2.5 Definition of BxPlatform – Autonomy Assertion Platforms

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

2.2.7 Definition of BxServiceCapability (BxSc)

2.2.8 Definition of BxServiceRealization (BxSr)

2.3 ByStar Platform Taxonomy and Genesis Process

«Xref-BxPlatform»

2.3.1 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
```

2.3.2 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.

2.3.3 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

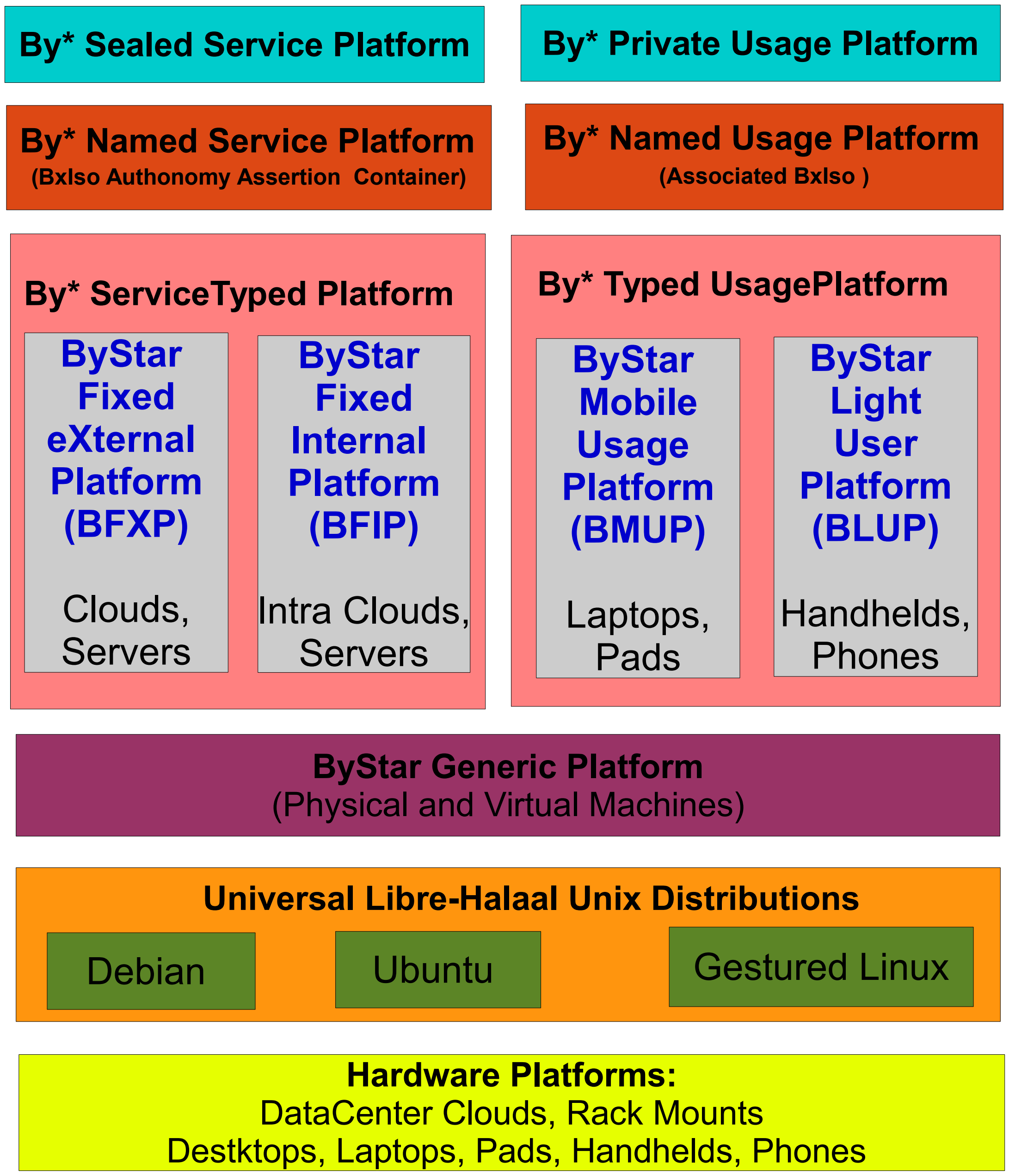

By* Sealed Service Platform
By* Private Usage Platform
By* Named Service Platform
(BxIso Authonomy Assertion Container)
By* Named Usage Platform
(Associated BxIso)
By* ServiceTyped Platform
ByStar Fixed eXternal Platform (BFXP)
Clouds, Servers
ByStar Fixed Internal Platform (BFIP)
Intra Clouds, Servers
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 2.1: Platform Genesis Steps And Layers

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.

2.3.4 ByStar Platform Connectivity Types

ByStar Fixed eXterior Connected (BFXC) – Ext-

Visible and reachable on Public Internet.

ByStar Fixed Interior Connected (BFIC) – Int-

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

ByStar Mobile Interior Connected (BMIC) – Mob-

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

2.3.5 ByStar Software Profile (Capability Types)

BxP-Service-SwP

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

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.

Bxp-User-SwP

For laptops and pads.
Previously BUE.

Bxp-Developer-SwP

For handhelds and phones.

BDPP – ByStar Domain (DNS) Publisher Platform

BDRP – ByStar Domain (DNS) Resolver Platform

ByStar Hosting Virtualization Platform

BCP – ByStar Custom Platform

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

Combination of:

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

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

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

2.3.7 BxP Software Platform Gensis Process

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

2.3.8 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
```

2.3.9 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.

2.4 BxE/BxISe Sovereignty, BxISe Registration and BxSe Types

2.4.1 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

2.4.2 Autonomous BxISe Types

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

2.4.3 Controlled BxISe Types

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.

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

Controlled Individual and Controlled Organization BxEs

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

Projects (Private and Collaborative) BxEs

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

2.4.4 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

2.4.5 Collaborative BxSe Types

1. ByLookUp

2.4.6 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

2.5 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.

2.6 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]

2.7 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) –.

2.7.1 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
```

2.7.2 Structure Of BxISo/iso

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
```

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
```

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              #
```

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
```

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
```

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
```

bxiso/iso/pkcs – Public/Private Keys

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

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                   #
```

2.8 BxISo Replications And Synchronizations – Cloning From Principle

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

2.9 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).

2.10 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
```

2.11 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.

2.11.1 BxSr Domains Bindings

2.11.2 ByStar Service Realization Modes (BxSrm)

BxSrm=live

BxSrm=shadow

BxSrm=here

BxSrm=inactive

2.12 LSIP Facilities Naming And Terminology

2.13 ByStar Content

Chapter 3

BISOS Architecture

3.1 Overview Of By* Internet Services OS (BISOS)

3.2 BISOS's "On" Unix Model

A framework for cohesive creation, deployment and management of Internet Services.

Unix Philosophy vs Linux Philosphy vs Business Philosphy.

Bringing the Cathedral to the Bazar, in the open.

With BISOS, there is no need for containers, ansible, chef, pupet.

3.3 Interactive Command Modules (ICM) and Players

Short summary plus reference to PLPC-180050.

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

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

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

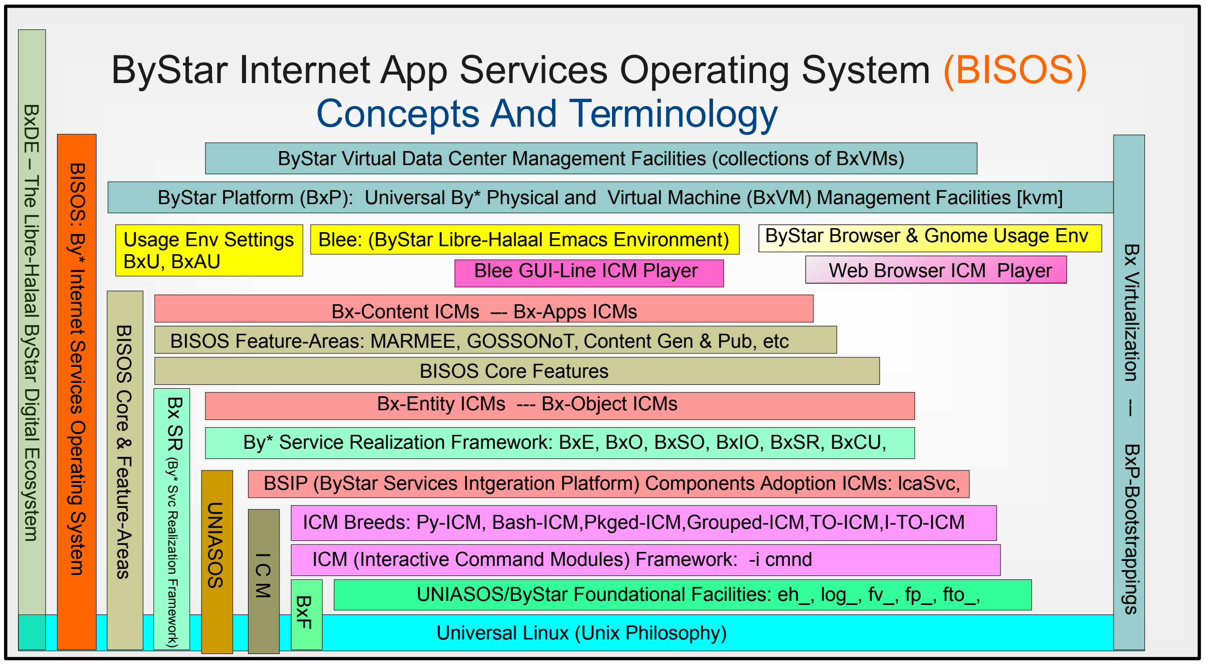

ByStar Internet App Services Operating System (BISOS)
Concepts And Terminology
ByStar Virtual Data Center Management Facilities (collections of BxVMs)
ByStar Platform (BxP): Universal By* Physical and Virtual Machine (BxVM) Management Facilities [kvm]
Usage Env Settings BxU, BxAU
Blee: (ByStar Libre-Halaal Emacs Environment)
ByStar Browser & Gnome Usage Env
Blee GUI-Line ICM Player
Web Browser ICM Player
Bx-Content ICMs — Bx-Apps ICMs
BISOS Feature-Areas: MARMEE, GOSSONoT, Content Gen & Pub, etc
BISOS Core Features
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
UNIASOS/ByStar Foundational Facilities: eh_, log_, fv_, fp_, fto_,
Universal Linux (Unix Philosophy)
BxDE – The Libre-Halaal ByStar Digital Ecosystem
BISOS: By* Internet Services Operating System
BISOS Core & Feature-Areas
Bx SR (By* Svc Realization Framework)
UNIASOS
I C M
BxF
Bx Virtualization — BxP-Bootstrappings

Figure 3.1: BISOS Model and Terminology

3.4.1 /hss Base

3.4.2 /dd Bases – Disk Drives

3.4.3 Privacy Scopes (privScopes)

3.4.4 /uniform Bases

3.4.5 /de Bases – Digital Ecosystems

3.4.6 BxISo Base Dirs

3.4.7 BxU Base Dirs

3.5 BISOS Bases Installation

The root of installation of BISOS is: $bisosBase/bisos

Often, $bisosBase/bisos is /bisos. When we refer to /bisos we are referring to $bisosBase/bisos.

If /bisos has not been installed, we need to prepare the /bisos bases.

3.5.1 Basic Preparation

There are two ways of setting up the BISOS bases.

The setup with Python 2.7 default installation is simpler.

It is also possible to setup BISOS bases without intruding on the existing default python environment.

BISOS Bases Setup – Without Default Python 2.7 – Non-Intrusive To Default Python Environment

If Python 2.7 is not installed on your system, install it. Python 2.7 need not become the default python.

When Python 2.7 is the default installation, this mode of installation does not impact the default python environment. Everything runs under the /bisos/venv/py2-bisos-3 virtual environment.

```
pip install virtualenv

sudo mkdir -p /bisos/venv/py2-bisos-3
sudo chown -R aUser:aGroup /bisos

virtualenv --no-site-packages --python=python2.7 /bisos/venv/py2-bisos-3

source /bisos/venv/py2-bisos-3/bin/activate
pip install --no-cache-dir --upgrade bisos.bx-bases

bx-bases -v 20 --baseDir=/bisos  -i pbdUpdate all
```

BISOS Bases Setup – With Python 2.7 Default

```
pip install virtualenv

sudo mkdir -p /bisos
sudo chown -R aUser:aGroup /bisos

pip install --no-cache-dir --upgrade bisos.bx-bases

bx-bases -v 20 --baseDir=/bisos  -i pbdUpdate all
```

/bisos/venv/py2-bisos-3 virtenv is then created.

3.5.2 BISOS CORE Pip Installs In Virtenv

With /bisos base in place, we can now add BISOS Core.

```
source /bisos/venv/py2-bisos-3/bin/activate
```

Then add a BISOS feature-area (BISOS package)

For example:

```
pip install --no-cache-dir --upgrade bisos.core   # NOTYET
```

3.6 /bisos Bases Directory Structure Overview

BISOS base directories are created with bx-bases.

In bxpBaseDir.py with

```
@ucf.runOnceOnlyReturnFirstInvokation
def pbdDict_bisosRoot(baseDir,):
```

the directory structure is created.

Assuming /bisos as root, this is a brief description of the directory hierarchy.

/bisos/dist: Current distributions are accessible from this base. A distribution could have been obtained from pip, or through AnonVC, or through AuthVC.

/bisos/dist/r3: Base for links to bisos3 release version

/bisos/dist/r3/core: Symlinks to: /bisos/venv/py2-bisos-3/lib/python2.7/site-packages/bisos/coreDist-root

/bisos/dist/r3/pkgs:

/bisos/dist/r3/pkgs/marmee:

/bisos/dist/r3/pkgs/marmee/Panel.org:

/bisos/dist/r3/bsif: Symlinks to: /bisos/venv/py2-bisos-3/lib/python2.7/site-packages/bsif/bashDist-root

/bisos/dist/r3/blee: Symlinks to: /bisos/venv/py2-bisos-3/lib/python2.7/site-packages/blee/elispDist-root

/bisos/dist/d3: Base for links to bisos3 development version Whose Collaboration Bx-Controlled Object is based at: /de/bx/nne/dev-py/pypi/pkgs

/bisos/dist/pip: The distribution obtained through pip. Structured through symlinks to /bisos/venev

/bisos/dist/pip/bisos: Links to the current bisos version

/bisos/dist/pip/blee:

/bisos/dist/pip/bsip:

/bisos/venv: A base for all virtual environments.

/bisos/venv/py2-bisos-3': Obtained with:

```
virtualenv --no-site-packages --python=python2 /bisos/venv/py2-bisos-3
```

/bisos/venv/py3-bisos-3': Obtained with:

```
virtualenv --no-site-packages --python=python2 /bisos/venv/py3-bisos-3
```

/bisos/vcAuth: A base for Git/CVS/etc authenticated obtained dists.

/bisos/vcAnon: A base for Git/CVS/etc anonymous obtained dists.

/bisos/control: Symlink to /de/run/NOTYET

/bisos/main: Base for symlinks to current selected (pip/vcAuth/vcAnon) bisosCore

/bisos/main/core: Base for symlinks to current selected (pip/vcAuth/vcAnon) bisosCore

/bisos/main/pkgs: Base for symlinks to current selected (pip/vcAuth/vcAnon) bisosPkgs

/bisos/bsip: Base for symlinks to current selected (pip/vcAuth/vcAnon) bsip

/bisos/blee: Base for symlinks to current selected (pip/vcAuth/vcAnon) blee

3.7 /bxo Service-Objects And Information-Objects Account Bases Directory Structure Overview

/bxo/so/: Service Objects

/bxo/so/oneBxSO: UserName=suid (uid is userAcctNu)

/bxo/io/: Information Objects

/bxo/io/oneBxIO: UserName=suid (uid is userAcctNu)

/bxo/fso/: Foreign Service Objects

/bxo/fso/foreignBxSO: Symlink to user/bxo

/bxo/fio/: Foreign Information Objects

/bxo/fio/oneBxIO: Symlink to user/bxo

3.7.1 BxSO Service Objects Bases Directory Structure Overview

oneBxo/so/run: Symlink to oneBxo/so/r3/run

oneBxo/so/r3/run: Symlink to /de/run/bxo/eachBxo/r3

3.7.2 Service Realization Bases – Controls And Configurations

oneBxso/so/sr: Symlink to oneBxo/so/r3/sr

oneBxso/so/r3/sr:

oneBxso/so/r3/sr/abxp: Adopted Bx Platform (debian apps)

oneBxso/so/r3/sr/abxp/SvcEgApache2:

oneBxso/so/r3/sr/abxp/SvcEgApache2/web:

oneBxso/so/r3/sr/core:

oneBxso/so/r3/sr/pkgs:

oneBxso/so/r3/sr/pkgs/marme: SvcCapName=marme

oneBxso/so/r3/sr/pkgs/marme/dsnProc: SvcCapName=marme/SvcName=dsnProc

oneBxso/so/r3/sr/pkgs/marme/dsnProc/control:

oneBxso/so/r3/sr/pkgs/marme/dsnProc/admin:

3.8 /de/run Bases Directory Structure Overview

/de/run/bisos is the base from which /bisos facilities are to be configured and executed. In the ByStar model, /bisos/venv,dist bases are considered Read-Only. All configurations and executions of /bisos is expected to take place in /de/run/bisos.

Assuming /de/run as root, this is a brief description of the directory hierarchy.

/de/run:

/de/run/bisos:

/de/run/bisos/r3:

/de/run/bisos/r3/curs: Current BISOS Wide Selections

/de/run/bisos/r3/curs/bxso: File Param Base

/de/run/bxso:

/de/run/bxso/oneBxso:

/de/run/bxso/oneBxso/r3:

/de/run/bxso/oneBxso/r3/tmp: core,pkgs,abxp Adopted Bx Platform (debian apps)

/de/run/bxso/oneBxso/r3/var: core,pkgs,abxp Adopted Bx Platform (debian apps)

/de/run/bxso/oneBxso/r3/log: core,pkgs,abxp Adopted Bx Platform (debian apps)

/de/run/bxso/foreign:

/de/run/bxso/foreign/r3:

/de/run/bxso/foreign/r3/tmp: core,pkgs,abxp Adopted Bx Platform (debian apps)

/de/run/bxso/foreign/r3/var: core,pkgs,abxp Adopted Bx Platform (debian apps)

/de/run/bxso/foreign/r3/log: core,pkgs,abxp Adopted Bx Platform (debian apps)

NOTYET, to be absorbed with above.

/bisos/run:

/bisos/run/r3:

/bisos/run/r3/var:

/bisos/run/r3/var/core:

/bisos/run/r3/var/pkgs:

/bisos/run/r3/pnl:

/bisos/run/r3/pnl/core:

/bisos/run/r3/pnl/pkgs:

/bisos/run/r3/tmp:

/bisos/run/r3/log:

/bisos/run/r3/log/core:

/bisos/run/r3/log/pkgs:

3.9 Optional Addition Of BISOS Feature-Areas (BISOS-Pkgs)

With /bisos base in place, you can now add your desired BISOS packages.

```
source /bisos/venv/py2-bisos-3/bin/activate
```

Then add a BISOS feature-area (BISOS package)

For example:

```
pip install --no-cache-dir --upgrade unisos.marme
pip install --no-cache-dir --upgrade bisos-gossonot
```

3.10 Software Of By* and By* As Software

Software Of By* and By* As Software

The Universal BISOS

BISOS: ByStar Internet application Services OS

- BISOS Core Features
- BISOS Feature-Areas And ByStar Capabilities

BISOS Software Feature-Areas Provide For By* Service Capabilities

By* Services and BISOS User-Environments Then Enable

ByStar Software-Service Continuums

Notes:

All of ByStar uses a common core Libre-Halaal software. We call this common core “The Universal BISOS”.

BISOS stand for ByStar Internet application Services Operating System.

Engineering philosphy of BISOS builds on the Unix philosophy in the applications and internet services domain.

BISOS facilities are structured in two parts. BISOS core features are those that are essential. BISOS Feature-Areas provide for open-ended realization of additional ByStar Capabilities.

BISOS along with ByStar services provide for ByStar Software-Service Continuums.

3.10.1 By* Platform Layers And Universalities Of BISOS

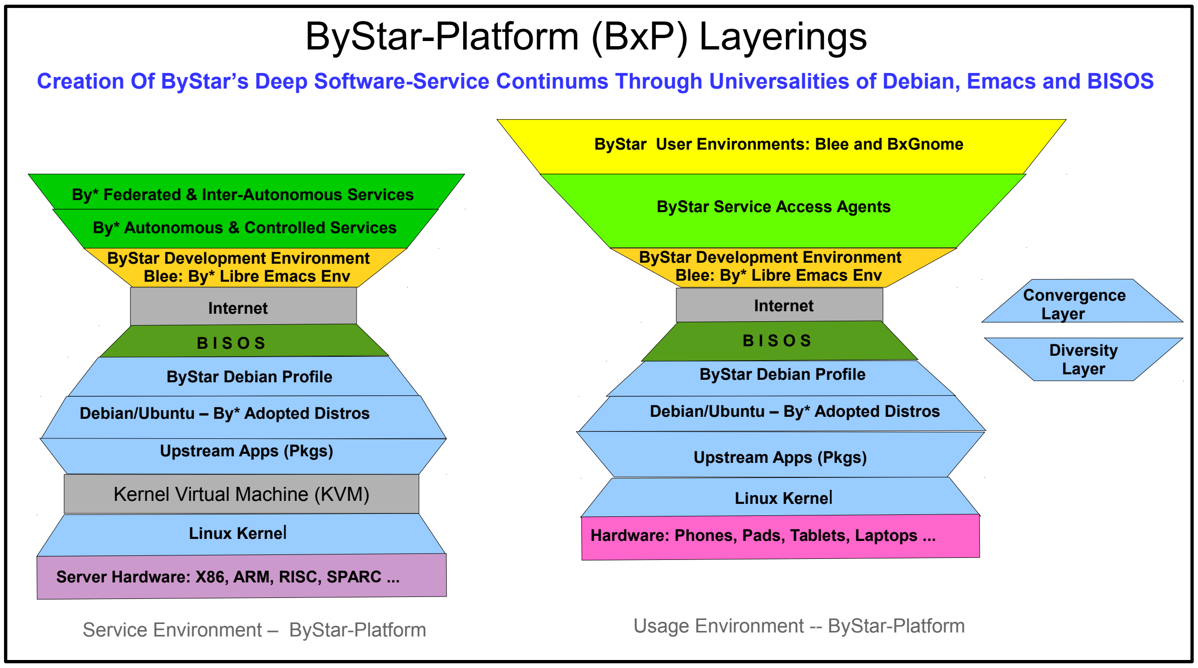

ByStar-Platform (BxP) Layerings
Creation Of ByStar's Deep Software-Service Continums Through Universalities of Debian, Emacs and BISOS
By* Federated & Inter-Autonomous Services
By* Autonomous & Controlled Services
ByStar Development Environment
Blee: By* Libre Emacs Env
Internet
B I S O S
ByStar Debian Profile
Debian/Ubuntu – By* Adopted Distros
Upstream Apps (Pkgs)
Kernel Virtual Machine (KVM)
Linux Kernel
Server Hardware: X86, ARM, RISC, SPARC ...
Service Environment – ByStar-Platform
ByStar User Environments: Blee and BxGnome
ByStar Service Access Agents
ByStar Development Environment
Blee: By* Libre Emacs Env
Internet
B I S O S
ByStar Debian Profile
Debian/Ubuntu – By* Adopted Distros
Upstream Apps (Pkgs)
Linux Kernel
Hardware: Phones, Pads, Tablets, Laptops ...
Usage Environment -- ByStar-Platform
Convergence Layer
Diversity Layer

Figure 3.2: ByStar Platform (BxP) Layerings

Notes:

In this figure, we are showing a certain perspetive for considering layerings in a ByStar-Platform (BxP).

This layering perspective, plus the goal of creating various universalities which would lead to deep software-service continuums is key to our design of the ByStar DE.

Universality of BISOS is inherited from being based on the Debian GNU/Linux. This universality permits all ByStarPalt-froms (BxP), be it handsets, devices, server, laptops and virtual machines – to function consistently and harmounously and collaboratively. This universality is rooted in all ByStar software to always be reproducible from its libre-halaal source.

Our choice of Emacs as a universal development and user environment brings an additional level cohesion to the BxDE.

3.10.2 BISOS Concepts And Terminology

Chapter 4

ByStar User Environments

4.1 COMEEGA: Collaborative Org-Mode Enhanced Emacs Generalized Authorship

4.1.1 COMEEGA For Human Oriented Machine Programming (HOMP)

4.1.2 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*.

4.2 Model Of Service Usage and Administration In ByStar Digital Ecosystem

4.2.1 ByStar User Environemnts

Blee

BxGnome

4.3 BxU and extasciitilde bxu Structure

4.4 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).

Chapter 5

Bootstarpping And Genesis

5.0.1 BxP Software Platform Gensis Process

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

5.0.2 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
```

5.0.3 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.

5.1 ByStar Bootstrappings

pip install bisos.bx-bases

bx-pip -i

VM BxP (ByStar Platform) Genesis ...

Chapter 6

BISOS Feature Areas And ByStar Capabilities

6.0.1 BISOS Core Features And Feature-Areas

BISOS Core Features	BISOS Feature-Areas
BLEE (By* Libre-Halaal Emacs Environment) and BxGnome	Autonomous Content Authorship, Generation And Publication
COMEEGA (Collaborative Org-Mode Enhanced Emacs Generalized Authorship	Inter Personal Communications (MARMEE)
ICM (Interactive Command Modules) and ICM Players	Multi-Media Organization and Consumption
ByStar Bootstrapings (Hosts, VMs and Web Factories)	Genealogy (Geneweb)
BxE/BxO Credentials and Synchronization (PKCS, Git-s)	GOSSONoT (Generalized Open-Source SON of Things)

Notes:

In the left column we provide an overview of the basic key common features of BISOS. These are facilities that permit basic interactions with ByStarCentral and which permit realization of additional capabilities for BxOs.

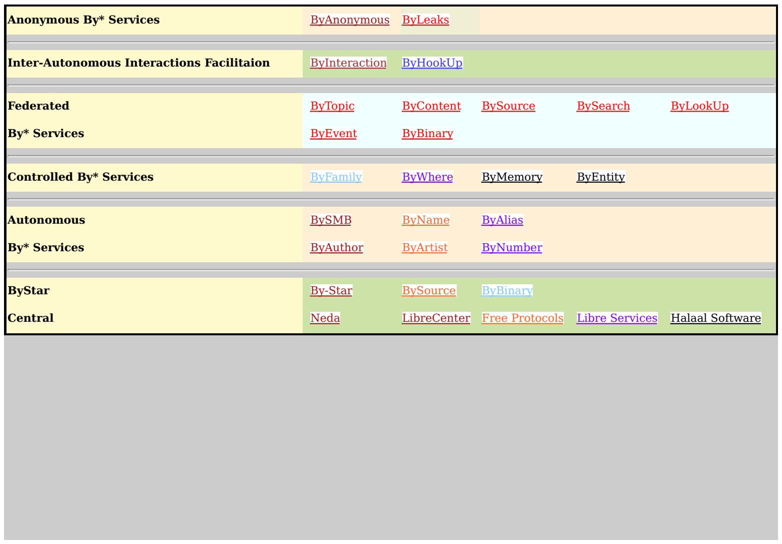
In the right column we provide an overview of some of the key feature-areas. feature-areas represent capabilities that By*Objects can inherit.

6.1 ByStar Internet Services – The By* Web Collective

Notes: In this table we provide a list of By domains that we have aquired and some place holder services that we have implemented.

These are categorized based on the type of entities associated with them.

6.2 ByStar Software And Service Catgorizations



Anonymous By* Services	ByAnonymous	ByLeaks			
Inter-Autonomous Interactions Facilitaion	ByInteraction	ByHookUp			
Federated	ByTopic	ByContent	BySource	BySearch	ByLookUp
By* Services	ByEvent	ByBinary			
Controlled By* Services	ByFamily	ByWhere	ByMemory	ByEntity	
Autonomous	BySMB	ByName	ByAlias		
By* Services	ByAuthor	ByArtist	ByNumber		
ByStar	By-Star	BySource	ByBinary		
Central	Neda	LibreCenter	Free Protocols	Libre Services	Halaal Software

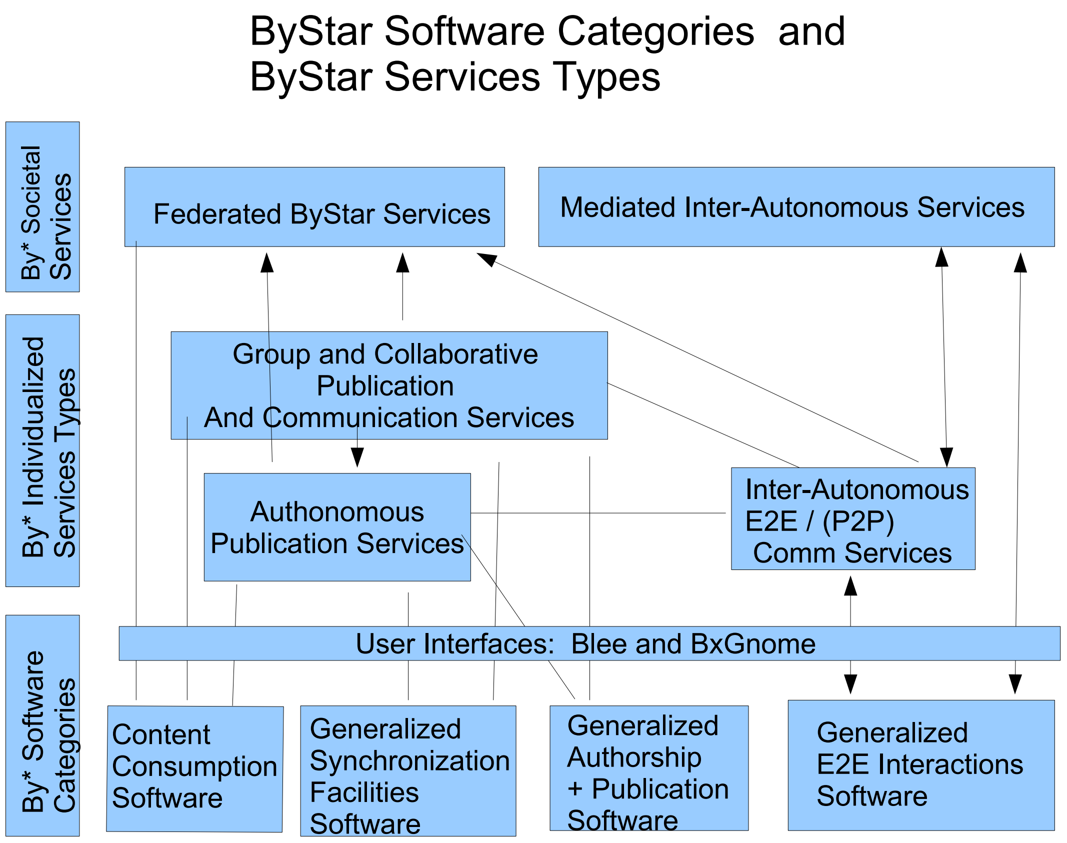

ByStar Software Categories and
ByStar Services Types
By* Societal Services
By* Individualized Services Types
By* Software Categories
Federated ByStar Services
Mediated Inter-Autonomous Services
Group and Collaborative Publication And Communication Services
Authonomous Publication Services
Inter-Autonomous 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 6.1: ByStar Software And Services Types

Chapter 7

BISOS Central

7.1 An Overview Of ByStar Central

- The Libre-Halaal Foundation – non-profit, non-proprietary
- ByStar Name and Number Assignment Authority: EntityId Assignments,
- Software Distribution: BySource.org, ByBinary.org, Debian, PyPi, Emacs-Repos

Implementation Infrastructure:

- Neda Communications, Inc. – for-profit, non-proprietary
- LibreCenter.net – Data Center, Private Clouds
- Hosted ByStar Services
- Asserted/Possessed ByStar Services

Notes:

Proper operation of BxDE requires a number of CentralServices.

In order to address this we have created ByStar-Central.

The Libre-Halaal Foundation – as a non-profit organization – has been formed to function as an organization reponsible for maintaining central integrity of ByStar.

ByStar-Central primary functions are:

- That of being a Name and Number Assignment Authority

- That of Being a Software Distribution center

LH Foundation and Neda Comm Inc have been intertwined in the early phases of the evolution of ByStar DE.

Chapter 8

Direction Statements

8.1 ByStar Crypto Cash

Bibliography