



## **Deus+ SDK**

### **Whitepaper**

Update March 15, 2005

## **Contents**

---

---

### **General**

---

Management Summary

About Ex Machina

About Deus+

### **For Game Developers**

---

Why license Deus+ ?

Availability

Licensing Model

### **More on the Technology**

---

Design Considerations

Architecture Overview

Specifications – Server, Client, Communications, Modules and Tools

### **Appendix**

---

Deus+ Network Core

Deus+ Event System

Roadmap

More on the history of Deus+

Getting Started – In a Nutshell



## Management Summary

The advantages of developing multiplayer games powered by Deus+:

- **Reliable, proven technology:** Deus+ has been working for five years and is deployed on live servers, serving over a million games played.
- **Transparent communication:** Any phone, Any Java version (J2ME, MIDP1.0, MIDP2.0, DoJa, J2SE, Series 40/60/90), any protocol (HTTP / TCP/IP).
- **Community tools:** Generic Lobby & Matching, Instant Messaging, Chatting and High Scores
- **Example game:** A working example is provided to simplify development with Deus+. Additionally, much (if not all) of the code can be applied in other applications.
- **Optimal use of the available network connection:** if the network gets quicker, Deus+ based applications will be quicker without any modification.
- **Continuous updates:** Deus+ is constantly revised as new mobile phones and relevant application platforms become available. This ensures your application's communication will work and keep working correctly, while being able to mix different kinds of phones with a single game and server.
- **Small footprint:** The Deus+ client core components have a memory footprint of 9 KB.
- **Easy integration and web support.**
- **Modularity:** Deus+ is an extension to your current environment and will not limit your possibilities or creative mind twists.

## About Ex Machina

Ex Machina is an Amsterdam-based innovative development studio. We're experienced guys who build tools and middleware for multiplayer games. Our multiplayer engine Deus+ runs on mobile phones, handhelds, PDA's and web browsers. We really like gaming, and will do anything to take multiplayer gaming to the next level – and the levels after that.

Besides providing the most compatible technology on the market for the most common phones, Ex Machina differentiates itself from other multiplayer platform vendors by focusing on just that: the multiplayer platform. Ex Machina believes a slim, pure and efficient core makes for a solid foundation, while offering a module-model for both adding one-off game-specific features as well as cross-game reusable functionality.



## About Deus+

As a proven multiplayer platform, consisting of a multiplayer server and clients for compatible networks and platforms, Deus+ is the best multiplayer engine available for current handsets. Deus+ has been available as a fully-fledged SDK since October 2003.

Its characteristics:

- Any phone (Nokia s40, s60, Sony Ericsson, Siemens, Sharp...)
- Any Java version (J2ME, MIDP1.0, 2.0, DoJa, J2SE)
- Any protocol (HTTP, TCP/IP)
- Any gaming concept
- Future Proof
- Perfectly suitable for MMOG's and location based games
- Continuous updates, professional support
- 9 kB footprint
- Designed to integrate and perform
- Reliable, proven technology
- Deployed for five years, millions of games played

Deus+ key USPs are its modularity, its reliability and proven track record, its proven compatibility with billing platforms, and its handset compatibility. By supporting open standards, integration with external systems such as billing platforms, content management systems and statistics packages is easy to implement. This makes it an ideal product for game developers who want to develop quality titles for a guaranteed market, without any technical hassles.

Deus+ will handle thousands of players simultaneously on a single server and is scalable to any capacity necessary. Our platform supports bandwidth as low as 9600 bps and deals with high network latency. What's more: All technology runs on proven code. We use, test and stress test the technology on our own products, which are running 24 hours a day providing millions of players daily gaming pleasure.

Deus+ was originally developed to power PC-based web games, but now also powers multiplayer gaming on GPRS-enabled mobile phones, as well as PDA's, pocket-PCs and interactive television. A 2.5G network like GPRS is already sufficient for true multiplayer games, while next generation networks will allow for an even richer experience. The server can handle any client platform that supports HTTP or TCP/IP sockets such as Java (J2ME and J2SE), Flash, and native C/C++ based platforms. This means support not only for smartphones but for common phones too - from every major vendor.

Deus+ opens up various new business models for publishers and operators and is available in various licenses. It comes included with full service and support.



## **Game Developers - Why license Deus+?**

We all know how fast technology evolves. Unless you have a really big R&D budget, it is impossible to keep pace. If you would develop your own multiplayer mobile gaming services, your technology is obsolete before it's able to function as needed.

So, it's better not to reinvent the wheel. Using our time-tested code gives you advanced technology from day one. You can play with it, you can tweak it, and you can focus on doing what you do best: Creating games. Our engine is proven and runs in various games and environments. We've experienced and tackled all the unforeseen complexities of multiplayer gaming.

Also: you know you need a fast development cycle. Licensing technology is always cheaper than releasing your title 6 months later than required. Start developing games today and focus on real game play, instead of technical problems and platform limitations.

## **Availability**

Our technology has been converted to a SDK system, and as such, has been available since October 15, 2003.

You can obtain the Deus+ SDK by contacting [info@exmachina.nl](mailto:info@exmachina.nl), or by calling +31 (0) 20 419 31 60. We'll send you an Non-Disclosure Agreement, after which we'll give you access to all development tools, including the support forum.

Please understand we are more than happy to incorporate your feedback in new versions!

## **Licensing Model**

We license the Deus+ SDK on a per studio basis. Multiplayer games allow for a great variety of business models, so please contact us to discuss the details.

The license includes the full SDK with support via the internet. On-site support, hosting, sales support, training and a telephone hotline can be included for extra costs.



## Design considerations

The Deus+ system is designed as a tool, enabling the application programmer to make more in less time. Deus+ is an extension to your current environment and will not limit your possibilities or creative mind twists.

To provide the most flexible solution to your requirements we have chosen to divide the SDK system in various core units. Of these units only the network core and event system are required for the SDK to function properly.

The SDK is completely event driven; just what you would expect from a JAVA based system. To further extend the possibilities and handling of events, one of the core building blocks in our development kit is the event system. This event system handles, schedules and forwards all events to various blocks within the complete system. The event system is explained further on.

The heart of the Deus+ SDK consists of two major components: the **network core** and the **event system**. The network core implements all network specific functionality needed for multi-user applications and provides a completely generic interface to this functionality. In other words, the protocol and network solutions used underneath do not influence the way the application developer writes his code.

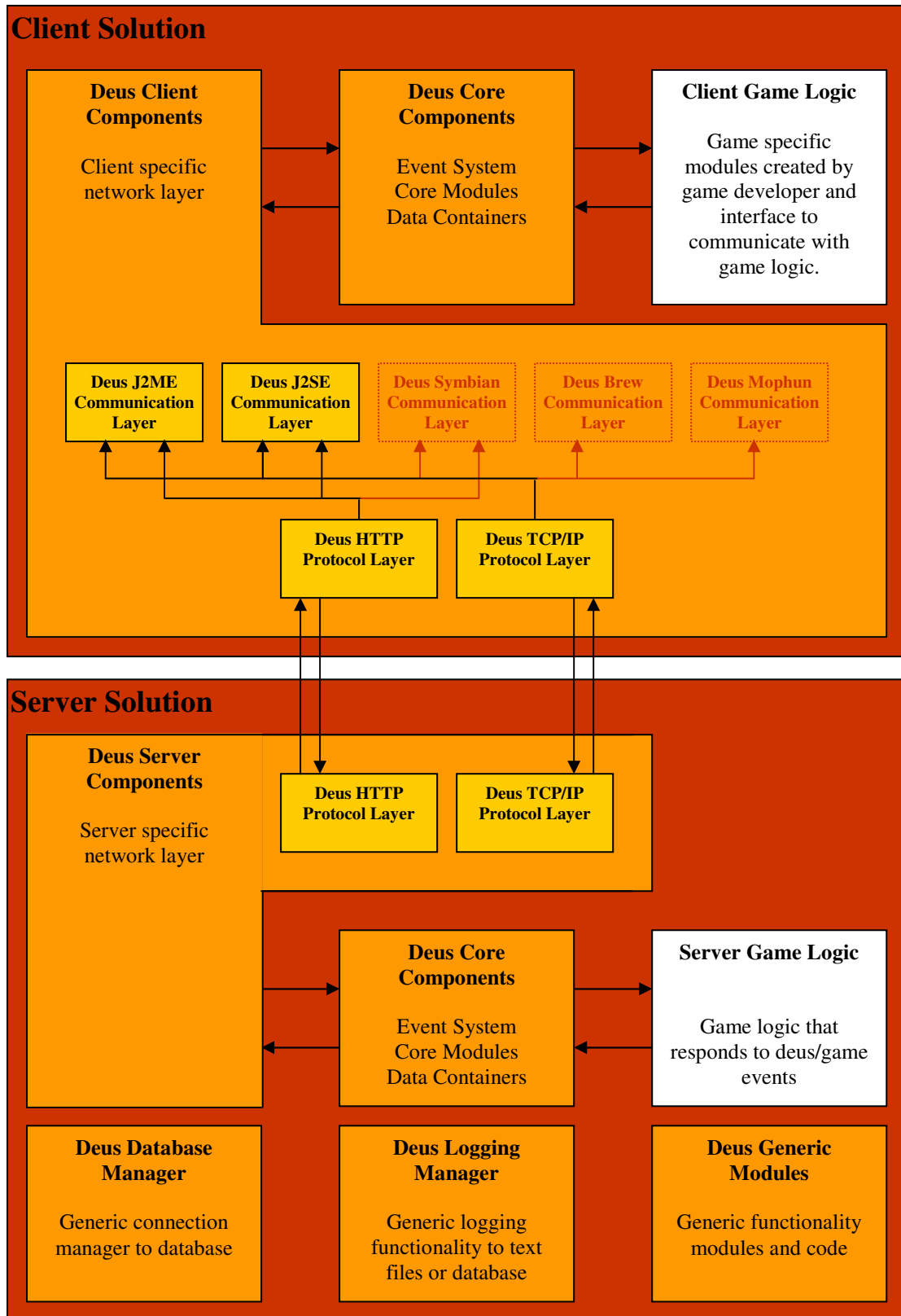
The event system is a standard Model-View-Control based system as seen in almost all Java APIs. Within this system there are two types of events; network events and module/internal events.

Furthermore, the core system can be customized to the developer's demands.

(See Appendix 1 and 2 for more on Network Core and Event System.)



## Architecture Overview



□ Blank boxes are non-Deus+ components created by the game/application developer.



## **Specifications - Server**

Deus+ is written in Java. Modules that have to run within the Deus+ server are therefore also required to be written in Java. Deus+ server can be run as a Servlet within a Servlet Runner (eg. Tomcat) or as a stand-alone server. Deus+ supplies a module based server architecture. A Deus+ Server has one or more modules for which it handles internal (Module to Module) and external (Client to Module, Module to Client) event communication. Deus+ supplies standard components for handling logging and database access.

### *Configuration*

Deus+ configuration files are in written in xml, this allows for easy understanding and changing the way Deus+ is configured.

## **Specifications – Client**

Deus+ is distributed with a client side written in Java. The client is free to be written in other languages and platforms (such as BREW, Flash, Symbian and Mophun), and future versions of Deus+ will be distributed with client side core system in other languages.

Deus+ Client core systems are distributed with both a J2ME and a J2SE version. This ensures that both MIDLets, applets and applications can use Deus+. The Deus+ client core components have a memory footprint of 9 kb.



## Specifications – Communication

Deus+ supplies Java-applications with a communication layer that standardizes communications between clients and servers. Features are:

### *Standard communication*

Deus+ uses a simple event based system. Events are easily passed from clients and received by the server and vice versa. This is implemented in a consistent and easy to understand manner.

### *Compatibility*

Deus+ works on all Java-enabled phones (Series 40/60/80).

### *Deployability*

Deus+ supplies application developers with a transparent communication layer. Deus+ will function on any phone regardless of the phone manufacturers Java VM implementation. This should of course not be necessary, since an application that works on one phone should work on all phones. Unfortunately, it is a wide-known issue that different Java VM's behave differently. Yet Deus+ solves these differences – in other words, the applications using Deus+ will not notice these Java VM Differences!

### *Protocols*

Deus+ can be set to work with any network protocol (HTTP, TCP/IP, etc). This is done transparently for applications using Deus+, who will not notice any difference in using one protocol or another. New protocols are added when they become commonly available on mobile phones.



## Specifications – Example Modules

### *User Module*

The User Module provides generic functionality for user management such as the login/logout process, account management and storing of properties per user. This data is either stored in text-files or a database. If stored in a database this module uses the Deus+ standard database manager (described later). The module is build to allow other (custom) ways of storing the data, perhaps needed to use an existing store of user data. This module can actually be used as is, since it supplies all basic functionality in user management.

### *Simple Module*

This module is the result of the implementation guide. It shows basic structure of Deus+ server and client modules and the general flow of events within Deus+.

## Specification – Production Modules

### *User Module*

Although it has the same name and basic functionality as the example user module, this module is quite different. It has not been built as an example module but as a production module, and is therefore designed to be as fast as possible. This comes at the expense of flexibility.

The module allows other modules to add user specific data, as it utilizes DeusDatabaseInitializer).

### *Highscore Module*

Module for keeping highscore lists. This module is designed to be as fast as possible. All actions are timed between 0 and 2 ms on test system. To achieve this speed this module preloads all highscore data on startup.

The module keeps track of highscores for 4 intervals: all-time, day, week and month. Data is synchronized with a configurable data source (usually a database). The data source can be accessed to retrieve top scores, for example to present them on web pages.

### *Location Module*

This module provides functionality to request a client's real world location from within Deus+. This module can be used to enable game concepts that revolve around where the player is in real world coordinates. Do note that this functionality is only available for countries where one or more operators support location based systems.



## Specification - Tools

### *DeusLog*

Deus+ features an extended log system. This system is used by Deus internally, but is also available to other apps within Deus. DeusLog was created because of the need for flexible logging. Both flexible in how much logging you need and where it has to go. Its features are:

- Easy configuring in an XML document;
- Periodic runtime reloading of settings, allowing you to change logging settings without the pausing of other processes;
- Several standard ways of output and the possibility to design and use an output method of your own choice.

### *DeusDatabaseManager*

Deus+ uses this database manager internally. But it can be used by others within Deus. Its features are:

- Multiple connections to a database. This allows queries to run parallel;
- The manager opens connections when many are needed and closes them if they are not;
- If a reconnect is called, all prepared statements previously set are immediately set on the reconnected connection;
- Thread-safety - only one thread can use one connection at a time.

### *DeusDatabaseInitializer*

Deus+ supplies this system to simplify transferring from one database to another (for example, when the servers themselves are being moved to another physical location). Any module can define what database tables it requires, even adding fields to tables of other modules. If the initializer is started and some or all defined tables do not exist, the initializer will try and create them according to the specified definitions.



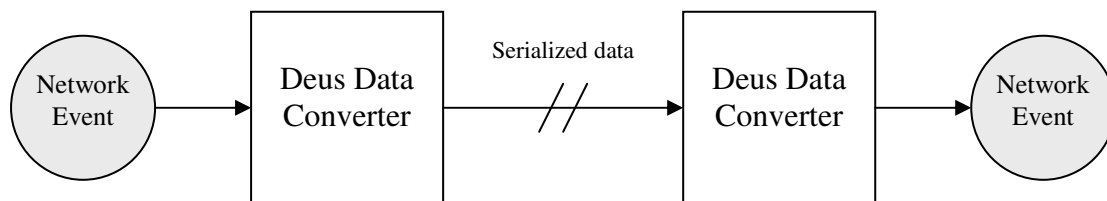
## Appendix 1: Deus+ Network Core

The networking core has been designed to overcome networking differences and to standardize these. This core unit provides a transparent way of communication, regardless of what underlying network architecture is available. It supports standard HTTP requests, HTTP via sockets and pure socket connections. The latter is currently still in beta.

The network core is explained further on as well.

When the Deus+ server is set up, various modules can be loaded by the Server. These components can be standard modules supplied with Deus+ or custom modules developed by our licensees. We have readily available modules for handling archiving high score tables or chat sessions and generic game operations.

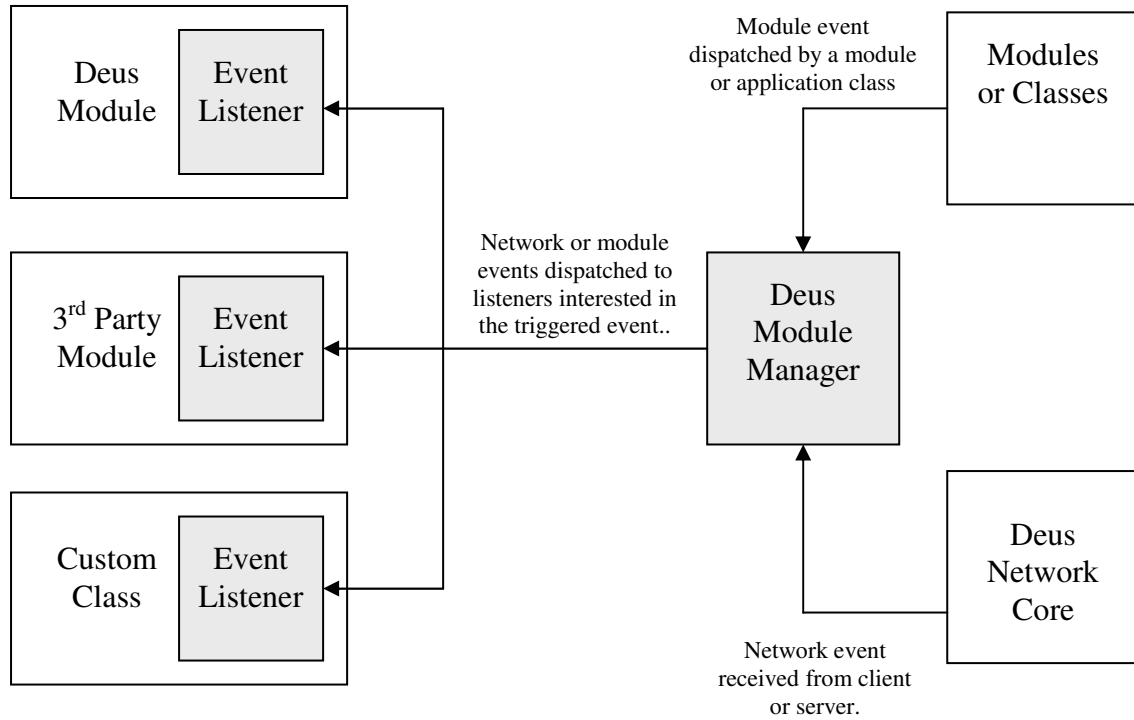
The interface to the network core, as presented to developers, is completely independent of the platform and network protocols used internally. As such, Deus+ supports protocols and platforms ranging from a high latency HTTP based servlet solution to a raw socket implementation and anything in between. This ensures that functionality is only limited by the restraints of the chosen platform.



The core takes care of connections between the Deus+ server and its clients. It manages disconnections and generates events to notify any piece of custom code that is interested. This allows you to neglect communications and only react on events passed on to your own modules.



## Appendix 2: Deus+ Event System





## **Appendix 3: Road Map**

Note: the road map may be subject to change.

### **Matching/Lobby Module**

ETA: Early April, 2005

### **Chat Module**

ETA: Mid-April, 2005

### **Instant Messaging Module**

ETA: Early May, 2005

### **Socket Server Communication**

ETA: May, 2005

### **BREW/Symbian Support**

ETA: June, 2005

### **Load Balancing**

ETA: June, 2005

### **Web-based Configuration and Statistics**

ETA: July, 2005



## Appendix 4: More on the History of Deus+

Ever since we started working on Deus+ for the web in the late 90s, we focused on making multiplayer games as accessible as possible. This meant dealing with narrowband connections with high latency and low bandwidth and allowing people to play games through even the most restricting firewalls. It meant not needing to install and configure desktop software or to have bleeding edge processing and graphics technology. Our philosophy proved to be successful with all of our web games, and helped draw not just hardcore gamers but whole crowds of casual gamers, women, kids, seniors and other mainstream audiences that liked to share an exciting online experience with friends or total strangers. In fact, valuing the sharing more than the game itself, it seemed.

After an initial stint with Flash-generating software from Macromedia that wouldn't perform at all, we decided to write our own multiplayer server engine from scratch, based on a J2EE framework. Combined with industry leading standards like Apache, Linux and PostgreSQL this turned out to be an efficient, scalable and robust basis for even the most popular multiplayer web games ever made in The Netherlands.

Coming from this background, our decision to enter the mobile games market when downloadable mobile applications became reality is easy to understand. The first true mass deployed platform for mobile applications, DoJa in Japan, offered the protocols required for basic online features in games. Another J2ME specification, MIDP 1.0, followed suit and established Java as the leading platform for mobile applications development. Other platforms started emerging as well, but have so far not achieved the popularity of J2ME with vendors, developers, publishers and consumers. Our server was written in Java, and it was designed to support the ubiquitous HTTP standard – which is the only protocol available to an application requiring a connection to a server in any MIDP 1.0 or DoJa phone.

Our experience and technology shows that there's a whole world of multiplayer opportunity, sitting comfortably in the middle between the bleeding edge technology required by real-time game concepts and the often yawn-inducing delays inflicted by turn-based installments. The key is to 'hide' the network synchronization (the part where the coordination between all the players takes place) in the game, and we have developed various mobile games already to prove that this point works, ranging from racing to parlor to action to puzzle to edutainment. Of course, turn-based can be cool, and online features like ghost-image exchanges and highscores will prove to be a great driver for mobile gaming and operator's data revenues. In fact, those features are very easy to add to any game - once you have a platform for live multiplayer games up and running that is.

In spite of all limitations discussed here, Ex Machina is first and foremost dedicated to provide a true multiplayer experience to the most common handsets on the market today. J2ME and GPRS capable phones like Nokia Series 30 and 40 (3510i, 5100, 6100, 6610, 7210, 7250), Sony Ericsson T610, T630 and Z600, Sharp GX10 and GX20, Siemens M55 and S55 and the Samsung SGH-E700 handsets have become very popular in 2003 and will remain best sellers in 2005. Of course, phones based on



Symbian, Palm and Microsoft operating systems offer more power, storage and networking protocols and are therefore easier to develop multiplayer games for. But as we like to enable multiplayer games for the masses, we have started with the most common denominator.

.



## Appendix 5: How to get started - in a nutshell

First of all, you must have a good gaming concept that allows the game to be played by multiple contestants simultaneously. Also, don't forget the network latency for GPRS and other mobile networks is still too high to allow stuff like *Quake*.

The trick is to create gaming concepts that hide the latency – or better yet, use it as a strength. A familiar concept is the turn-based game – such as strategy games, trading games, board and quiz games. But don't limit your creativity to turn-based ideas! Take, for example, a game where players *do* play simultaneously, but are unable to interfere with each other. For example: everybody is in the hunt for the same thing, and the goal is to get somewhere (or capture something) as soon as possible. Such a game can be as fast-paced as you like, with a great sense of multiplayer competition!

Please note that the Deus+ SDK is in no way limiting your options or adding to the latency. It has been designed to take optimal advantage of the available network connection. As soon as the network latency is reduced (either by providers reducing GPRS latency, or by the adaptation of UMTS) Deus+ based applications can be used without modification of the application code.

Again: the key is to use the latency as strength, not as a weakness. If you're creative, you can even "hide" it so that the people playing won't even notice.

*A checklist for developing your game:*

- Study structure of the SDK (Read help and further technical documentation)
- Design your own GUI models and client/server game logics (models) in a way they can interact with the SDK
- Develop application-specific custom modules
  - o Try using available modules as much as possible
- Make test suit for checking connections
- Determine latency on various devices and connections and alter your concept if needed
- Implement as much game logic on the server as possible

A full log (example / walkthrough) is available for registered developers.